T-1564

SOLVING THE GEOMETRIC PROGRAMMING PROBLEM USING CONDENSING MONOMIALS TO APPROXIMATE POSYNOMIAL DENOMINATORS

By

Danny Lee Taylor

T-1564

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 (Mathematics)

Signed: Danny L Gaylor
Student

Golden, Colorado

Date: 3 May, 1973

Approved: Robert H Wolfe
Thesis Advisor

Joseph R Lee
Head of Department

Golden, Colorado

Date: 3 May, 1973

ABSTRACT

Real World engineering optimization problems are often non-linear. Recent developments using approximating monomials for groups of posynomial terms will allow the solution of many of these problems using Geometric Programming. A time-sharing computer program is developed utilizing these techniques. Sample problems and run costs are provided.

TABLE OF CONTENTS

ACKNOWLEDGMENTS

I would like to thank my thesis advisor, Dr. R. E. D. Woolsey, for originally suggesting the need for the program developed herin. I also thank Professor C. R. Baer for the use of his matrix inversion routine, and Paul Treece for his assistance in modifying this program for the Colorado School of Mines PDP-10. Finally, I wish to thank A. C. Williams for telling me what I should do with "minus-one" degree of difficulty problems.

INTRODUCTION

In the Real World an engineer is often requested to optimize a given process with respect to cost. More often than not the equations defining the cost of the process are non-linear, thus ruling out the use of Linear Programming techniques. The engineer's usual (and textbook recommended) approach is a straight forward application of differential Calculus to determine the minimum of the cost equation.

Since this is the most straight forward and engineeringly obvious attack, it is difficult to fault this method. However, the cost equation can frequently be classified as "hideously" non-linear in a large number of variables. In this case the application of the Calculus can result in a large system of still hideously non-linear simultaneous equations. The engineer then has the choice of calling in an over-paid consulting mathematician or spending a great deal of computer time (and money) using various search techniques in a large-number dimension space. Neither of these alternatives is particularly appealing to either the engineer or his supervisor, who both

have more important things to do than to wait around for an answer that was needed yesterday.

Recently there has appeared on the engineering scene an alternative to the Calculus: Geometric Programming (G.P.). It turns out, that if a system can be defined in terms of posynomials (polynomials with only positive coefficients) then a great deal of the hideousness can be made to go away. The problem becomes one of inverting matricies rather than multi-dimensional searches.

However, the engineers have not been entirely cooperative with the mathematicians in this alternative approach. They seem to insist that some process equations contain terms with negative coefficients, thus violating the canons of G.P. Engineers maintain that these heretical negative terms are needed to model the Real World. The negatives are not within the framework of Geometric Programming Theory. There seems to be an impasse.

However, in 1971, a paper by Avriel and Williams described a method by which groups of terms can be approximated by a single monomial. This can not only remove the negative term problem, but may actually reduce the degree of difficulty of the problem (a measure of the Geometric hideousness). This may be the approach which will bridge the impasse between Theory and Real World engineering.

The purpose of this study has been to develop a usable, interactive, time-sharing computer program for the

solving of non-linear minimization problems using G.P. and utilizing the method of Avriel and Williams as needed. The scope of the program has been limited to unconstrained, zero-degree of difficulty problems expressable as polynomials or ratios of polynomials. Even with this limitation, the program still allows the engineer to sit down at a teletype, enter his cost equation in standard FORTRAN format and receive his answer back immediately. (In most cases the teletype speed was found to be the limiting factor in returning an answer.)

DESCRIPTION OF THE PROBLEM

The Primal Geometric Programming Problem can be described as:

Minimize $g_o(t)$ (1)

subject to: $g_k(t) \leq 1$, $k=1,2,\ldots,p$ (2)

where $g_k(t)$ is a posynomial for $k=0,1,2,\ldots,p$ having the form:

$$g_k(t) = \sum_i c_{ki} \prod_j t_j^{a_{kji}} \quad (3)$$

where $c_{kj} > 0$

$t_j > 0$

and a_{kji} are arbitrary real constants.

Taking the degree of difficulty as defined by Duffin, Peterson, and Zener (1967) to be:

D.D. = number of terms - number of variables - 1

we find that the zero degree of difficulty problem can be solved merely by inverting the matrix representing the normality and orthoganality conditions of the Dual Problem.

If the equation $g_o(t)$ is not in the form of (3) then new variables may be introduced to make it conform. For example if:

$$g_o(t) = (t_1^{-1} + t_2) \cdot (t_1 + t_2^{-2})^e$$

then new variables t_3 and t_4 can be defined so that the problem is now:

Minimize $g_o(t) = t_3 t_4^e$

subject to: $t_3 \geq t_1^{-1} + t_2$

$t_4 \geq t_1 + t_2^{-2}$.

If the constraint equations are each divided by their left-hand sides, they are then consistent with (2). (The preceding is all due to and explained further in Chapter 1 of Duffin, Peterson and Zener (1967).)

If, however, the original equation read:

Minimize $g_o(t) = (t_1^{-1} + t_2)^{-1} \cdot (t_1 + t_2^{-2})^e$

then our first constraint would be:

$$t_3 \leq t_1^{-1} + t_2 \qquad (4).$$

In order to make this constraint of the form $g(t) \leq 1$, there are two methods of approach. The first is to subtract one term (for example t_2) and then divide by the other, thus:

$$t_1 t_3 - t_1 t_2 \leq 1.$$

However, note that one of those heretical negatives has crept into the equation.

The other approach would be to divide (4) by its right-hand side:

$$\frac{t_3}{t_1^{-1} + t_2} \leq 1 \qquad (5).$$

This, however, does not yield $g_1(t)$ as a posynomial but rather as the ratio of two posynomials.

At this point we will fall back on the basis of all G.P.: The Geometric Inequality. Given a general posynomial $Q(t)$ where

$$Q(t) = \sum_{i=1}^{N} q_i(t)$$

if we choose a normalizing vector u such that $\sum_{i=1}^{N} u_i = 1$, then by the Geometric Inequality:

$$Q(t) = \sum_{i=1}^{N} q_i(t) \geq \prod_{i=1}^{N} \left(\frac{q_i(t)}{u_i}\right)^{u_i}.$$

Avriel and Williams (1971) have shown that if we choose a positive vector $\bar{t}$ we can define $Q(t,\bar{t})$ as

$$Q(t,\bar{t}) = \prod_{i=1}^{N} \left(\frac{Q(\bar{t})}{q_i(\bar{t})} \cdot q_i(t)\right)^{q_i(\bar{t})/Q(\bar{t})}.$$

Note that $Q(t,\bar{t})$ is a single positive monomial term and, moreover, that:

$$Q(t) \geq Q(t,\bar{t}) \qquad (6).$$

Returning to the original problem, we can allow the terms of the constraints to have the form of either posynomials or ratios of posynomials. In the latter case the constraint is given as:

$$g_k(t) = \frac{P_k(t)}{Q_k(t)} \leq 1 \qquad (7).$$

Using (6) and substituting $Q(t,\bar{t})$ for $Q(t)$ we have:

$$\frac{P_k(t)}{Q_k(t,\bar{t})} \leq \frac{P_k(t)}{Q_k(t)} \leq 1.$$

Note that when the division in $\frac{P(t)}{Q(t,\bar{t})}$ is carried out the result is the posynomial $P(t)$ with a factor of $Q(t,\bar{t})^{-1}$ in each of the terms of $P(t)$. Thus the original ratio of posynomials has been approximated by a posynomial with a number of terms strictly less than the original constraint. In this case Avriel and Williams redefine degree of difficulty as:

D.D. = no. of terms in numerators-no. of variables-1.

The degree to which $Q(t,\bar{t})$ approximates $Q(t)$ depends on $\bar{t}$ and t. When $\bar{t} = t$ then $Q(t,\bar{t}) = Q(t)$. This seems to indicate an iterative procedure for the solution of the original problem. The procedure suggested by Avriel and Williams is to start with an arbitrary positive vector $\bar{t}$, and then use this to determine a solution vector t_1 from the system:

Minimize $g_o(t_1)$

subject to $\frac{P_k(t_1)}{Q_k(t_1,\bar{t})} \leq 1 \qquad k=1,2,\ldots,p.$

The solution vector t_1 is then used in place of $\bar{t}$ to determine t_2 which is then used to find t_3, etc. This is done until two successive solutions for $g_o(t)$ are within some predetermined limit.

The rigorous proof that this series of solutions does in fact converge was demonstrated by Avriel and Williams in 1970 (p. 136-140). This proof is quite detailed and will not be presented here. The basis of the proof lies in the fact that the approximation algorithm involves a point-set mapping of a point in the feasible region (x_n) into a subset of the feasible region (the approximating constraints). Avriel and Williams showed that a sequence generated by this type of mapping under the proper conditions will converge to a quasi-minimum point. This quasi-minimum can then be shown to correspond to a local minimum for the Complementary Geometric Programming Problem, and under not too exclusive conditions to a global minimum for the Primal Geometric Programming Problem.

DEVELOPMENT OF THE PROGRAM

The development of the computer program for the project was divided into two separate but interrelated phases. First was a list-processor code which would accept the FORTRAN coded objective function as input, then "crack" the statement and generate the matricies needed for the solution. The second part of the program is the actual solution of a zero-degree of difficulty problem using condensed posynomials if necessary.

In the writing of the program three guidelines were established. First, although the program is written for the Colorado School of Mines PDP-10, the code should be written in standard FORTRAN-IV in such a way that conversion to any other time-sharing system can be accomplished with a minimum of pain and effort by any competent FORTRAN programmer.

Secondly, although the code is only for processing unconstrained polynomial objective functions, ample "hooks" should be left in the program on which to hang code for handling transcendental functions such as e^x and Log x, as well as code for processing user supplied constraints. (Surprisingly, the Real World sometimes imposes physical

limitations on cost equations.)

Finally, although the program is only meant to process zero-degree of difficulty problems by matrix inversion, the code should be designed so that more powerful techniques for handling higher degrees of difficulty could be employed.

In other words, this program was not designed as a finished product, but as a flexible framework on which a comprehensive and sophisticated Geometric Programming package can be built. To this same end, a large number of comments are included in the code in order to aid in future program development.

List-processor

The first part of the program is the list-processor which "cracks" the user supplied objective function. Simplicity of use rather than ease of programming is the rationale for this list-processor. It was felt that a user-oriented program which, because of its ease of use, would be employed regularly by the working engineer, has a higher utility than a program which minimizes core and run-time at the expense of simple data-entry and, therefore, sits unused in the program library until someone has the sense to delete it. Therefore, a minimum level of programming expertise is assumed for the user. He must be able to Log-on to the system and call the program. And he must be

able to cast his objective function in the form of a FORTRAN replacement statement, i.e.:

VARIABLE = FORTRAN statement

where VARIABLE is any user-supplied name for the quantity to be minimized, and the FORTRAN statement is the FORTRAN representation of the polynomial objective function.

Due to the limited scope of this program, parentheses may be nested only two-deep; although, up to nine parenthetical enclosures are allowed.* Each such enclosure will be replaced by a Program Generated Variable(PGV) and the appropriate constraint (condensed or not). The total number of variables, user-supplied and PGV, may not exceed 20.

Any variable names acceptable in standard FORTRAN may be used with no limit on length. However, for the PDP-10 only the first five characters will be used by the program and must be unique for different variables.

Constants may be either integer or decimal notation; no scientific or E format is allowed. If present a constant coefficient must be the first factor in a term and only one coefficient is allowed per term.

---

*Note that all references to program size limitations are for the code supplied in this thesis. Increase, or decrease, can be accomplished by adjusting DIMENSION statements and the initialization routine.

Formatting of the statement is again standard FORTRAN. The statement must be contained between columns 7 and 72. If additional space is required continuation "cards" may be entered by placing any non-blank character in column 6 of the continuing statement. All leading, imbedded, and trailing blanks are ignored.

The operating method for the list-processor is based on a subroutine called WHATSIT. Input to this routine is a starting column for analysis. It returns the type of parameter found starting in the given column (variable, constant, or operator) and the width of the field containing this parameter.

The analysis proceeds from left to right, one term at a time, utilizing repeated calls to WHATSIT. As a term is analysed, an exponent matrix A and a coefficient matrix C are constructed. These matricies are consistent with the notation of Kochenburger (1968).

As parentheses are encountered they are scanned to determine type (posynomial or general polynomial) and number of terms. When a close-parentheses is determined, flags are set for proper constraint generation and a PGV is assigned.

After each term in the objective function is completed, any outstanding constraints are generated. If condensation is not needed, entries for the constraint go directly into the A and C matricies. If, however, the constraint is in the

form of a ratio, the entries go into matricies P and PC for the numerator or Q and QC for the denominator.

Once the last term in the objective function has been analysed, and the last constraint has been generated, the program prints the constructed matricies so that the user can see how the program is considering his problem with regard to PGVs and term numbers (for delta analysis). At this point, degree of difficulty is determined. If the degree of difficulty is zero, the problem is passed to the next section of code for solution.

Problem Solution

The actual solution to the zero-degree of difficulty problem is through the solution of the dual problem to determine the dual variables (deltas) and then solve for the primal variables. If the problem requires no condensation this process is simply to build from the exponent matrix A a square matrix CDEL which represents the normality and orthogonality conditions of the dual problem. This matrix is inverted to solve for the deltas. These deltas are then used to determine the value of the objective function and, through another matrix inversion, to determine the primal variables. This method is adequately described in Chapter 1 of Duffin, Peterson, and Zener.

If, however, the problem contains constraints which require condensation then the approximation A matrix is

dependent on the assumed values of the primal variables. Therefore, an arbitrary positive value for the primal variables is assumed and the coefficient and exponents of the denominator-approximating monomial are determined. These exponent values are then subtracted from each term in the numerator to yield a new A matrix. The problem is then solved as in the preceding paragraph.

After this solution has been determined, the value of the objective function is compared to its previous value (initially set to machine infinity). If the ratio of the two values is within a preset limit, the solution is declared final and the program asks if a rerun is desired. If the ratio is not within the limit of convergence, the current values of the primal variables are used to determine a new approximation and the process is repeated until the limit is reached.

TEST PROBLEMS

Test problems for the code were formulated from Real World engineering situations. They represent a set of typical cost equations with which an engineer might have to work.

The first four problems are from F. C. Jelen (1970) and serve to demonstrate code capabilities for solving zero-degree of difficulty problems without condensation. The first two are minimization problems and the latter two show maximization techniques. The remaining problems are from a paper by I. W. Kabak (1969) and use the approximating method of condensed posynomials.

Uncondensed Problems

The following are from Chapter 12 of Jelen (1970).

Problem 1 -- This problem is to determine the drilling cycle-time for an oil-well drilling rig to minimize cost per month. The cost equation is given as:

$$C_m = 4{,}250 \cdot N \cdot T_d + 10{,}500 \cdot N$$

where

$$N = 6.667 \cdot T_d^{-0.5}$$

C_m is cost in \$/month

and T_d is cycle-time in days.

This was entered into the program as:

```
COST = 28333*TD**.5 + 70000*TD**-.5
```

The program determined the deltas as $\delta_1 = \delta_2 = \frac{1}{2}$, cost per month as $.8907 \times 10^5$ dollars, and cycle-time as 2.471 days.

Problem 2 -- This problem has two variables and determines the batch size and power requirements for a plastic-production batch mixer. Let S be the size of the batch in pounds and P be the electrical power required in kilowatts. By assuming a yearly required production of 1,000,000 lb and a power cost of 1.0 ¢/kwh, the cost per year can be written:

$$C_y = 316.2S^{0.5} + 10^8/(PS^{0.5}) + 34.3P.$$

This equation was entered as:

```
COST = 326.2*S**.5 + 100000000*S**-.5/P + 34.4*P
```

The program determined the deltas as $\delta_1 = \delta_2 = \delta_3 = 1/3$, COST as $.3082 \times 10^5$ \$/year, and the size as 10,560 lb requiring 300 kw to operate.

Problem 3 -- This is a maximization problem to determine the load factor for an electrical power generation station rated at 20,000 kw, so that the maximum profit is realized. Let X be the load factor in Mwatts. Then the cost per hour is given as:

$$12 + .2X + .27X^2$$

and the income per hour is:

$$16X - .2X^2.$$

The net revenue per hour is therefore:

$$R_x = (16X - .2X^2) - (12 + .2X + .27X^2)$$

or $$R_x = -12 + 15.8X - .47X^2.$$

If we define R_x' as $R_x + 12$ then the problem becomes:

Maximize $R_x' = 15.8X - .47X^2$ or

Minimize $1/R_x' = \dfrac{1}{15.8X - .47X^2}.$

The problem was entered as:

```
MIN = 1/(15.8*X - .47*X**2)
```

One PGV was assigned as:

```
PGV1 = 15.8*X - .47*X**2.
```

This gives three terms and two variables equaling zero degrees of difficulty. Deltas were determined as $\delta_1 = \delta_2 = \delta_3 = 1$, and MIN = .007531, PGV1 = 132.8, X = 16.8.

The load factor for maximum revenue is, therefore, 16.8 Mwatts, QED. Note, however, that more information than just the load factor has been generated; the formula for PGV1 is, in fact, that of R_x'. Therefore, the revenue generated at optimality is PGV1 - 12 or $120.80 per hour.

Problem 4 -- This again is a drilling problem, except that this time maximum revenue is to be determined as a function of drilling cycle-time. The revenue per month is given as:

$$R_m = \frac{9{,}000 T_d^{0.5}}{(.75 + T_d)}$$

Since the program is set to minimize, the reciprical of the revenue is entered as:

MIN = 1/ (9000*TD**.5/(TD+.75))

In that two levels of parentheses are present, the two PGVs were assigned from the inside out as:

PGV1 = TD + .75

PGV2 = 9000*TD**.5/PGV1.

With these substitutions made the program determined the deltas as $\delta_1 = 1$, $\delta_2 = \delta_3 = \frac{1}{2}$, $\delta_4 = 1$; and

MIN = .0001925

PGV1 = 1.5

PGV2 = 5,196

TD = .75

Therefore the cycle-time for optimum revenue is .75 days. Again the revenue generated can be determined from the substitutions by noting that the formula for PGV2 is that of the desired revenue which is therefore equal to $5,196.

The answers found for the preceding four problems correspond exactly to those which Jelen calculated using the Calculus.

Condensed Problems

The following problems are from a study by Kabak (1969) on optimum design of steady-state availability of a production system. Letting the mean time to failure equal U and the mean downtime equal D, then the total cost is the sum of three components: the cost associated with U, the

cost associated with D, and the cost associated with the proportion of downtime, $\frac{D}{(D+U)}$. The cost equation can be written as:

$$C = K_1U^a + K_2D^b + K_3 \frac{D}{D+U} \qquad (8).$$

Where K_1, K_2, K_3, and a are positive constants and b is a negative constant.

Kabak decided that the problem was not directly acceptable to G.P., due to the presence of a heretical negative in a forced constraint and a degree of difficulty of one. However, he noted that as long as $DU^{-1} << 1$, then the final term can be approximated as K_3DU^{-1}; which reduces the degree of difficulty to zero. Kabak goes on in the paper to solve six sample problems using G.P. and the equation:

$$C = K_1U^a + K_2D^b + K_3DU^{-1}.$$

However in the last three examples worked, the assumption that $DU^{-1} << 1$ was not valid and Kabak noted that the G.P. "solutions" were invalid and, therefore, differential Calculus should probably be employed rather than G.P.

For the purpose of this thesis, the original equation (8) was entered into the condensing code with the result that the problem was analysed to be:

Minimize $C = K_1U^a + K_2D^b + K_3D/PGV1$

subject to $\frac{PGV1}{D+U} \leq 1.$

For specific values of K_1, K_2, K_3, a and b the following runs were made. Tables summerize results of each iteration using a criteria of 1.0 per cent improvement to determine the final solution.

Problem 5 -- $K_1 = 1$, $K_2 = 1{,}000$, $K_3 = 100{,}000$, a = 1, and b = -1; the problem was entered as:

```
CI = U + 1000/D + 100000*D/(D+U)
```

STEP	δ_1	δ_2	δ_3	δ_4	U	D	PGV1	CI
0					3	9	27	
1	.1667	.1667	.6667	.6667	1858.	.5381	7.238	11,150
2	.3333	.3333	.3334	.3334	463.9	2.155	464.5	1,392
3	.3328	.3328	.3344	.3344	462.7	2.161	464.9	1,390
Kabak	1/3	1/3	1/3		464.	2.15		1,392

Problem 6 -- K_1, K_2, and K_3 as in Problem 5, and with a = ½, b = -½; the problem was entered as:

```
CII = U**.5 + 1000*D**-.5 + 100000*D/(D+U)
```

STEP	δ_1	δ_2	δ_3	δ_4	U	D	PGV1	CII
0					3	9	27	
1	.2500	.2500	.5000	.5000	9×10^6	1.11	58.46	379.7
2	.4000	.4000	.2000	.2000	33140	30.17	33140	455.1
3	.3999	.3999	.2001	.2001	33120	30.19	33150	455.1
Kabak	2/5	2/5	1/5		33144	30.2		455.

Problem 7 -- K_1, K_2, and K_3 as above and with a = 2, b = -2; the problem was entered as:

```
CIII = U**2 + 1000*D**-2 + 100000*D/(D+U)
```

STEP	δ_1	δ_2	δ_3	δ_4	U	D	PGV1	CIII
0					3	9	27	
1	.1000	.1000	.8000	.8000	49.10	.6441	3.33	24,110

STEP	δ_1	δ_2	δ_3	δ_4	U	D	PGV1	CIII
2	.2484	.2484	.5033	.5033	35.15	.8996	35.92	4,976
3	.2468	.2468	.5063	.5063	35.01	.9032	35.92	4,966
Kabak	1/4	1/4	1/2		35.4	.89		5,020

The three preceding problems yield very similar answers to the results Kabak obtained. Significant variation can be noticed in Problem 7 which seems to be due to the break-down of the assumption that $DU^{-1} << 1$.

Problem 8 -- $K_1 = 100$, $K_2 = 10{,}000$, $K_3 = 1{,}000$, a and b as in Problem 5; the problem was entered as:

CIV = 100*U + 10000/D + 1000*D/(D+U)

STEP	δ_1	δ_2	δ_3	δ_4	U	D	PGV1	CIV
0					3	9	27	
1	.1667	.1667	.6667	.6667	2.728	36.66	33.60	1,637
2	.0608	.0608	.8783	.8783	.7683	130.2	117.3	1,623
3	.0058	.0058	.9884	.9884	.0601	16640	16620	1,000
4	.00004	.00004	.9999	.9999	.0004	3×10^5	3×10^5	1,000
5	0	0	1.0	1.0	0	∞	∞	1,000
Program halt due to zero values of deltas.								
Kabak	1/3	1/3	1/3		10.0	10.0		3,000

What has happened in this problem is that the coefficients have been chosen so that the values of U and D fall outside the feasible region. That is, the first term is driven to zero by setting U to zero, the second term goes to zero by letting D increase without bound, and then the ratio D/(D+U) approaches one, thus letting the total cost be equal to K_3 or 1,000. This situation was not encountered

in Problems 5, 6, or 7 since the value of K_3 (=100,000) was greater than the attained minimum.

This seems to contradict Geometric Programming Theory which states that for a zero-degree of difficulty problem the dual variables (deltas) are invariant with respect to the primal coefficients. That is, therefore, the deltas for Problem 8 should be the same as those for Problem 5, since a and b are the same in both problems.

What has actually occurred is that the problem is <u>not</u> zero-degree of difficulty. It is a one-degree of difficulty problem which we have approximated with a zero-degree of difficulty problem. Therefore, the dependence of the dual variables upon the primal coefficients is not at all surprising.

This brings us to a very important discovery, which shall be stated here as Taylor's Axiom for Reality in Operations Research:

> By assuming different views of a problem, we may be able to apply different mathematical concepts as tools to solve the original problem. However, neither the altered view-points nor the mathematics used can effect a change in the physical reality of the original situation.

This Axiom was stated more succinctly by Alfred Korzybski as: "A map is not the territory it represents."

CONCLUSIONS

Since this program was developed to be used in the solution of Real World problems, the primary criteria for its utility must be cost to use. This cost is measured in both computer costs and in engineering man-hours needed to enter data and receive results.

Runs for the preceding test problems were made on the Colorado School of Mines PDP-10 computer to determine these two cost factors. The elapse time and total cost figures determined represent time and cost for the user to Log-on to the system, request execution of the compiled program, enter data, receive results, and Log-off. The dollar costs are from the CSM computer resources accounting system. It is assumed that private sector cost, either internal or external, would be similar for runs made on similar equipment.

The following table summerizes the time and cost figures obtained for the PDP-10.

Problem	CPU time (sec.)	Elapse time (min.)	Total cost ($)
1	.67	4	0.38
2	.82	3	.37
3	.82	3	.38
4	.98	3	.38

Problem	CPU time (sec.)	Elapse time (min)	Total cost ($)
5	1.52	5	0.45
6	1.52	5	.44
7	1.60	6	.45
8	1.65	7	.48

From this table the time and cost for these runs ranged from 3 to 7 minutes, $0.37 to $0.48; with an average of 4.5 minutes and $0.42. For the uncondensed problems the average was 3.0 minutes and $0.38; with 5.6 minutes and $0.45 for the problems requiring condensation. These times and dollar costs are well within the usual economic limits set on engineering time-sharing applications, and the program, therefore, could easily be economically useful in a Real World situation.

The condensation principle proposed by Avriel and Williams and employed in this program appears to be a quite useful tool in this type of problem solving. The user should be careful, however, to relate the program results back to the original problem so that false assumptions are not made in generalizing from approximation solutions.

APPENDIX

The following section contains the computer listings relevant to this thesis. The first section is a listing of the actual code. The second is a print-out of the sample problems previously described.

```
C*      * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C       THIS PROGRAM SOLVES ZERO DEGREE DIFFICULTY GEOMETRIC PROGS,
C       THE INPUT EQUATIONS ARE IN STANDARD FORTRAN FORMAT
C       FULL DESCRIPTION IS FOUND IN MASTERS THESIS T-1564
C       WRITEN BY DANNY L TAYLOR SPRING 1973.
C*      * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      INTEGER V
      INTEGER COMNT,BLANK
      INTEGER VNAME,TECN,SHFT
      COMMON /NOT/ NOTTM,NOTEQ
      COMMON /LIM/ IL
      COMMON NVAR,NVARPG,IPFLG(20)
      COMMON KQT(10),KPT(10),KT(10)
      COMMON A(20,20,10),C(20,10),P(10,10,10),Q(10,10,10),PC(10,10),
     + QC(10,10)
      COMMON CARD(16),KARD(198)
      COMMON VNAME(20)
      COMMON WAV(20),SAV(20)
      COMMON LEVEL(10),LL2(10),IPRNO(10),IPRNC(10),KTYPE(10),LOCATE(10),
     + KNSTVAR(10),L2TAB(10)
      DIMENSION X(20),LCTBACK(10)
      DIMENSION DELTA(20),SUMDEL(10)
      DIMENSION CDEL(20,20)
      DIMENSION SQ(10)
      DIMENSION NVEC(20)
      DIMENSION B(20)
      DIMENSION IKARD(66,6)
      EQUIVALENCE (KARD,IKARD)
      REAL LIMIT
       NPROB=0
      LIMIT=1.01
C
C     SET UNIT NUMBERS FOR OUTPUT
C
      V=4
      ITTY=4
      WRITE(ITTY,1006)
 1006 FORMAT(' ENTER OUTPUT UNIT NUMBER-- TTY=4, LPT=6'/)
      READ(ITTY,1007) IOT
 1007 FORMAT(I)
      IE=4HE
      COMNT=5HC
      BLANK=5H
C*      * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     INITIALIZE ARRAYS
C
C* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
  6   CONTINUE
```

```
      DO 5 I=1,10
      KQT(I)=0
      KPT(I)=0
       KT(I)=0
      DO 2 J=1,20
      DO 1 K=1,20
 1    A(J,K,I)=0.
 2    C(K,I)=0.
      DO 4 J=1,10
      DO 3 K=1,10
      P(K,J,I)=0.
 3    Q(K,J,I)=0.
      PC(J,I)=0.
 4    QC(J,I)=0.
      LEVEL(I)=0
       LL2(I)=0
       IPRNO(I)=0
       IPRNC(I)=0
      KTYPE(I)=0
      LOCATE(I)=0
      KNSTVAR(I)=0
      L2TAB(I)=0
      LCTBACK(I)=0
 5    CONTINUE
      NVAR=0
      NVARPG=0
      LCN=0
C*     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     READ IN EQUATION AND DECODE INTO BUFFERS
C
      NPROB= NPROB + 1
      WRITE(IOT,1004) NPROB
 1004 FORMAT(1H1,///5X,' PROBLEM NO.',I3//)
      NCAR=0
 10   CONTINUE
      WRITE(ITTY,1002)
 1002 FORMAT(' ENTER CARD'/)
      READ(V,1000) CARD
 1000 FORMAT(16A5)
      DECODE(6,1001,CARD) KCOL1,KCOL2
 1001 FORMAT(A1,4X,A1)
 1003 FORMAT(5O13)
      IF(KCOL1.EQ.IE) GO TO 21
      WRITE(IOT,1005) CARD
 1005 FORMAT(1X,16A5)
      IF(IOT.EQ.ITTY) GO TO 11
      WRITE(ITTY,1005) CARD
 11   CONTINUE
      IF(KCOL1.EQ.COMNT) GO TO 10
      IF(KCOL2.EQ.BLANK) GO TO 20
```

```
      IF(NCAR.GE.1) GO TO 12
      ISTOP=1
      GO TO 9001
 12   NCAR=NCAR+1
      DECODE(72,1012,CARD) (IKARD(I,NCAR),I=1,66)
 1012 FORMAT(6X,66A1)
      DO 13 I=1,66
 13   IKARD(I,NCAR)=SHFT(IKARD(I,NCAR),-29)
      GO TO 10
 20   IF(NCAR) 99,12,21
 99   ISTOP=99
      GO TO 9001
 21   CONTINUE
      IL=NCAR*66
      KEQC=0
      KEQ=1
C*     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     CODE TO ANALYSIS STATEMENT
C
      CALL GETTEC(KCOL,TECN)
C     RETURNS WITH KCOL EQ TO COL CONTAINING =
C
C     BEGIN TERM BY TERM ANALYSIS
C
      NTERM=0
      CALL BUMPCOL(KCOL)
 24   CONTINUE
      DO 241 NV=1,NVAR
 241  WAV(NV)=0.
      IDEL=0
      ISH=1
      WC=1.
      NTERM=NTERM+1
 25   CALL WHATSIT(KCOL,ITYP,IWD)
      GO TO (34,29,27,26,26,900) ITYP
 900  ISTOP=900
      GO TO 9002
 901  ISTOP=901
      GO TO 9002
C*     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     PLUS OR MINUS
 26   IF(IDEL.NE.0) GO TO 901
      IDEL=1
      IF(ITYP.EQ.5) IDEL=-1
      CALL BUMPCOL(KCOL)
      IF(KCOL.GT.0) GO TO 25
      ISTOP=610
      GO TO 9005
C*     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
C       OPEN PARENS TYPE 3
 27     CONTINUE
C
C       ANALYSIS OF LEVEL 1 PARENS
C
        LAST=7
        IAD=1
        ISUB=Ø
        IFLG2=Ø
        KCOL1=KCOL
        CALL BUMPCOL(KCOL1)
        IF(KCOL1.LE.Ø) GO TO 927
        CALL WHATSIT(KCOL1,ITYP,IWD)
        IF(ITYP.NE.5) GO TO 27Ø3
        IAD=Ø
        GO TO 27Ø6
 27Ø1   KCOL1=KCOL1 + IWD - 1
 27Ø2   CALL BUMPCOL(KCOL1)
        IF(KCOL1.LE.Ø) GO TO 927
        CALL WHATSIT(KCOL1,ITYP,IWD)
 27Ø3   GO TO(27Ø1,27Ø4,271Ø,27Ø5,27Ø6,27Ø7,27Ø7,27Ø7,274Ø,928) ITYP
 27Ø4   IF(LAST.EQ.8) LAST=7
        GO TO 27Ø1
 27Ø5   IF(LAST.EQ.8) GO TO 27Ø2
        IAD=IAD + 1
        GO TO 27Ø7
 27Ø6   IF(LAST.EQ.8) GO TO 27Ø2
        ISUB=ISUB + 1
 27Ø7   LAST=ITYP
        IF(LAST.LE.7) 27Ø2,27Ø1
C*       * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C       BEGIN SCAN OF LEVEL 2 PARENS
C
 271Ø   CONTINUE
        KCOL2=KCOL1
        IAD1=1
        ISUB1=Ø
        ISN=1
        IF(LAST.EQ.6) ISN=-1
        LAST=7
        CALL BUMPCOL(KCOL2)
        IF(KCOL2.LE.Ø) GO TO 927
        CALL WHATSIT(KCOL2,ITYP,IWD)
        IF(ITYP.NE.5) GO TO 2713
        IAD1=Ø
        GO TO 2716
 2711   KCOL2=KCOL2+IWD-1
 2712   CALL BUMPCOL(KCOL2)
        IF(KCOL2.LE.Ø) GO TO 927
        CALL WHATSIT(KCOL2,ITYP,IWD)
```

```
 2713 GO TO (2711,2714,998,2715,2716,2717,2717,2717,2720,928) ITYP
 927  ISTOP=927
      GO TO 9003
 928  ISTOP=928
      KCOL=KCOL2
      GO TO 9002
 2714 IF(LAST.EQ.8) LAST=7
      GO TO 2711
 2715 IF(LAST.EQ.8) GO TO 2712
      IAD1=IAD1 + 1
      GO TO 2717
 2716 IF(LAST.EQ.8) GO TO 2712
      ISUB1=ISUB1 + 1
 2717 LAST=ITYP
      IF(LAST-7) 2712,2712,2711
C*     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     SET PGV VALUES FOR LEVEL 2
C
 2720 CONTINUE
      CALL GETPGV(KCOL1,KCOL2)
      IF(IAD1.EQ.1.AND.ISUB1.EQ.0) GO TO 2722
      IF(IAD1) 910,2721,2724
 910  ISTOP=910
      KCOL=KCOL1
      GO TO 9002
 2721 IF(ISUB1) 910,2722,2723
 2722 KTYPE(NVARPG)=1
      GO TO 2730
 2723 K=4
      IF(ISUB1.GT.2) K=6
      GO TO 2729
 2724 IF(ISUB1) 910,2725,2726
 2725 K=2
      IF(IAD1.GT.2) K=5
      GO TO 2729
 2726 K=7
      IF(IAD1.EQ.1.AND.ISUB1.EQ.1) K=3
 2729 KTYPE(NVARPG)=K
 2730 LEVEL(NVARPG)=2
      KNSTVAR(NVARPG)=NVAR
      KCOL1=KCOL2
      CALL BUMPCOL(KCOL2)
      IF(KCOL2.LE.0) GO TO 927
      CALL WHATSIT(KCOL2,ITYP,IWD)
      IF(ITYP.NE.8) GO TO 2731
      KCOL2=KCOL2+IWD-1
      CALL BUMPCOL(KCOL2)
      IF(KCOL2.LE.0) GO TO 927
      CALL WHATSIT(KCOL2,ITYP,IWD)
      IF(ITYP.NE.5) GO TO 2731
```

```
      ISN=-ISN
C
C     SET TEMPORARY FLAG IN WAV FOR SIGN OF A(PGV)
C
 2731 WAV(NVAR)=ISN
      IFLG2=IFLG2 + 1
      L2TAB(IFLG2)=NVARPG
C
C     RETURN TO LEVEL 1 SCAN
      GO TO 2702
C*     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     SET PGV VALUES FOR LEVEL1 PARENS
C
 2740 CALL GETPGV(KCOL,KCOL1)
      IF(IAD.EQ.1.AND.ISUB.EQ.Ø) GO TO 2742
      IF(IAD) 91Ø,2741,2744
 2741 IF(ISUB-1) 91Ø,2742,2743
 2742 KTYPE(NVARPG)=1
      GO TO 275Ø
 2743 K=4
      IF(ISUB.GT.2) K=6
      GO TO 2749
 2744 IF(ISUB) 91Ø,2745,2746
 2745 K=2
      IF(IAD.GT.2) K=5
      GO TO 2749
 2746 K=7
      IF(IAD.EQ.1.AND.ISUB.EQ.1) K=3
 2749 KTYPE(NVARPG)=K
 275Ø LEVEL(NVARPG)=1
      KNSTVAR(NVARPG)=NVAR
      IVAR=NVAR
      KCOL=KCOL1
C
C     SET LINK INDEX (LL2) FOR LEVEL 2 PGV
C
 2751 IF(IFLG2.LE.Ø) GO TO 35
      K=L2TAB(IFLG2)
      LL2(K)=NVAR
      IFLG2=IFLG2-1
      GO TO 2751
C*     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     CONSTANT -- TYPE 2
C
 29   CALL UNCODE(KCOL,WC,IWD)
      IF(IDEL.LT.Ø) WC=-WC
C*     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     LOOK FOR NEXT OPERATOR AFTER COEF OR EXPONENT PROCESSING
```

```
C
 30    KCOL=KCOL+IWD-1
       CALL BUMPCOL(KCOL)
       IF(KCOL.LE.0) GO TO 500
       CALL WHATSIT(KCOL,ITYP,IWD)
       ISH=1
       K=ITYP-2
       GO TO (903,500,500,31,32,903) K
 903   ISTOP=903
       GO TO 9002
C*     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C      SLASH -- TYPE 6
 31    ISH=-1
C
C      STAR  -- TYPE 7
 32    KCOL=KCOL+IWD-1
C
C      LOOK FOR NEXT VARIABLE
C
       CALL BUMPCOL(KCOL)
       IF(KCOL.LE.0) GO TO 999
       CALL WHATSIT(KCOL,ITYP,IWD)
       IF(ITYP.EQ.3) GO TO 27
       IF(ITYP.NE.1) GO TO 904
C
C      CHECK FOR VARIABLE IN SYMBOL TABLE
C
 34    CALL ISITYET(KCOL,IVAR,IWD)
       KCOL=KCOL + IWD -1
C
C      LOOK FOR NEXT OPERATOR AFTER VARIABLE IDENTIFICATION
C
 35    WAV(IVAR)=ISH
       CALL BUMPCOL(KCOL)
       IF(KCOL.LE.0) GO TO 500
       CALL WHATSIT(KCOL,ITYP,IWD)
       K=ITYP-2
       GO TO (905,500,500,36,36,40,905) K
 905   ISTOP=905
       GO TO 9002
 904   ISTOP=904
       GO TO 9002
 36    CONTINUE
       ISH=1
       IF(K.EQ.4) 31,32
C*     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C      PROCESS EXPONENT
C
 40    KCOL=KCOL+IWD-1
```

```
 41   CALL BUMPCOL(KCOL)
      IF(KCOL.LE.0) GO TO 999
      CALL WHATSIT(KCOL,ITYP,IWD)
      GO TO (906,45,906,41,42,906) ITYP
 906  ISTOP=906
      GO TO 9002
 42   ISH=-ISH
      GO TO 41
C
C     DECODE EXPONENT AND RETURN TO LOOK FOR NEXT OPERATOR
C
 45   CALL UNCODE(KCOL,VAREXP,IWD)
      SH=ISH
      WAV(IVAR)=VAREXP*SH
      GO TO 30
C*     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     CODE TO END TERM -- BEGIN AGAIN
C
C
C
C     MOVE WORKING VECTOR WAV INTO ARRAY A
C
 500  CONTINUE
      DO 501 I=1,NVAR
      A(I,NTERM,1)=WAV(I)
 501  CONTINUE
      C(NTERM,1)=WC
C
C     START CONSTRAINT GENERATION AT STATEMENT 5023
C
      GO TO 5021
 502  KT(K)=KTERM
 5021 IF(LCN.GE.NVARPG) 5022,5023
 5022 IF(KCOL.LE.0) 700,24
 5023 LCN=LCN + 1
      KVAR=KNSTVAR(LCN)
C*     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     SET ISN EQUAL TO SIGN OF EXPONENT OF PGV
C     GET PGV TYPE AND JUMP TO PROCESS
C
      SN=A(KVAR,NTERM,1)
      IF(LEVEL(LCN).LT.2) GO TO 503
      K=LL2(LCN)
      XDEL=A(K,NTERM,1)
      SN=SN*XDEL
      A(KVAR,NTERM,1)=0.
 503  XDEL=1.
      ISN=SIGN(XDEL,SN)
      KCOL1=IPRNO(LCN)
```

```
      LCOL=IPRNC(LCN)
      K=LCN + 1
      KTERM=0
      IRTN=1
      GO TO (510,520,530,540,520,540,570,9006)  KTYPE(LCN)
C*     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C
C     THIS BEGINS THE TERM ANALYSIS ROUTINE - INLINE SUBROUTINE
C
 510  KTERM=KTERM+1
      DO 5101 I=1,NVAR
 5101 WAV(I)=0.
      ISH=1
      WC=1.
      DEL=1.
 5105 CONTINUE
      CALL BUMPCOL(KCOL1)
      IF(KCOL1.GE.LCOL) GO TO 5150
      CALL WHATSIT(KCOL1,ITYP,IWD)
      GO TO (5123,5115,5123,5105,5110,913) ITYP
 913  ISTOP=913
      KCOL=KCOL1
      GO TO 9002
 5110 DEL=-1.
      GO TO 5105
 5115 CALL UNCODE(KCOL1,WC,IWD)
 5116 KCOL1=KCOL1+IWD-1
      CALL BUMPCOL(KCOL1)
      IF(KCOL1.GE.LCOL) GO TO 5150
      CALL WHATSIT(KCOL1,ITYP,IWD)
      ISH=1
      GO TO(913,5150,5149,5120,5121,913),ITYP-2
      GO TO 913
 5120 ISH=-1
 5121 CALL BUMPCOL(KCOL1)
      IF(KCOL1.GT.LCOL) GO TO 919
      CALL WHATSIT(KCOL1,ITYP,IWD)
      GO TO (5123,914,5123,914)  ITYP
 914  ISTOP=914
      KCOL=KCOL1
      GO TO 9002
 5123 CALL ISITYET(KCOL1,IVAR,IWD)
      WAV(IVAR)=ISH
      KCOL1=KCOL1 + IWD - 1
      CALL BUMPCOL(KCOL1)
      IF(KCOL1.GE.LCOL) GO TO 5150
      CALL WHATSIT(KCOL1,ITYP,IWD)
      GO TO (915,5150,5149,5120,5124,5125,916) ITYP-2
 915  ISTOP=915
      GO TO 9002
```

```
 916  ISTOP=916
      GO TO 9002
 5124 ISH=1
      GO TO 5121
 5125 KCOL1=KCOL1+IWD-1
      XDEL=1.
 5126 CALL BUMPCOL(KCOL1)
      IF(KCOL1.GE.LCOL) GO TO 917
      CALL WHATSIT(KCOL1,ITYP,IWD)
      IF(ITYP.EQ.2) GO TO 5130
      IF(ITYP.EQ.4) GO TO 5126
      IF(ITYP.NE.5) GO TO 918
      XDEL=-1.
      GO TO 5126
 917  ISTOP=917
      GO TO 9005
 918  ISTOP=918
      GO TO 9002
 919  ISTOP=919
      GO TO 9005
 5130 CALL UNCODE(KCOL1,VAREXP,IWD)
      WAV(IVAR)=WAV(IVAR)*XDEL*VAREXP
      GO TO 5116
 5149 KCOL1=KCOL1-1
 5150 CONTINUE
      WC=WC*DEL
      GO TO (5155,5205,5305,5705,5355,5375,5365) IRTN
C
C     END OF TERM ANALYSIS ROUTINE
C*     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C*     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     TYPE 1 -- SIMPLE MONOMIAL
 5155 WAV(KVAR)=-1.
      SN=ISN
      DO 5156 I=1,NVAR
 5156 A(I,KTERM,K)=WAV(I) * SN
      C(KTERM,K)=WC**ISN
      GO TO 502
C*     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     TYPE 2 -- BINOMIAL + +
C     TYPE 5 -- POSYNOMIAL
C     IF ISN NEGATIVE LOOP THUR TYPE SEVEN LOGIC
C
 520  IF(ISN.LT.0) GO TO 5701
      IRTN=2
      GO TO 510
 5205 CONTINUE
C
```

```
C      MOVE WAV TO A -- NOTE NO CONDENSATION NEEDED
C
       WAV(KVAR)=-1.
       DO 5210 I=1,NVAR
 5210  A(I,KTERM,K)=WAV(I)
       C(KTERM,K)=WC
       IF(KCOL1.GE.LCOL) 502,510
C*     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C      TYPE 3 -- MIXED BINOMIAL
C      IF ISN NEGATIVE BRANCH TO 5350 -- NO CONDENSATION NEEDED
C
 530   IF(ISN.LT.0) GO TO 5350
 5301  IRTN=3
       KQTERM=0
       KPTERM=0
       KEQC=KEQC+1
       PC(1,KEQC)=1.
       GO TO 510
 5305  CONTINUE
C
C      CHECK COEF TO DETERMINE IF TERM IS Q OR P
C
       IF(WC) 5320,920,5310
 920   ISTOP=720
       GO TO 9008
 5310  CONTINUE
       KPTERM=KPTERM+1
       DO 5311 I=1,NVAR
 5311  P(I,KPTERM,KEQC)=WAV(I)
       PC(KPTERM,KEQC)=WC
       GO TO 5330
 5320  KQTERM=KQTERM+1
       DO 5321 I=1,NVAR
 5321  Q(I,KQTERM,KEQC)=WAV(I)
       QC(KQTERM,KEQC)=-WC
 5330  IF(KCOL1.LT.LCOL) GO TO 510
       KQTERM=KQTERM+1
       Q(KVAR,KQTERM,KEQC)=1.
       QC(KQTERM,KEQC)=1.
       GO TO 580
C*     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C      LOGIC TO PROCESS MIXED BINOMIAL WITH NEGATIVE ISN
C
 5350  IRTN=5
       GO TO 510
 5355  CONTINUE
       IF(WC) 5360,921,5370
 921   ISTOP=921
       GO TO 9008
```

```
 5360 C(KTERM,K)=-WC
      DO 5361 I=1,NVAR
 5361 A(I,KTERM,K)=WAV(I)
      IRTN=7
      GO TO 510
 5365 C(KTERM,K)=WC
      DO 5366 I=1,NVAR
 5366 SAV(I)=-WAV(I)
      WAV(KVAR)=-1.
      DO 5367 I=1,NVAR
      A(I,KTERM,K)=-WAV(I)
 5367 A(I,KTERM-1,K)=SAV(I)+A(I,KTERM-1,K)
      C(KTERM-1,K)=C(KTERM-1,K) /WC
      GO TO 502
 5370 C(KTERM,K)=1./WC
      DO 5371 I=1,NVAR
 5371 SAV(I)=-WAV(I)
      SAVC=WC
      WAV(KVAR)=-1.
      DO 5372 I=1,NVAR
 5372 A(I,KTERM,K)=-WAV(I)
      IRTN=6
      GO TO 510
 5375 CONTINUE
      C(KTERM,K)=-WC/SAVC
      DO 5376 I=1,NVAR
 5376 A(I,KTERM,K)=SAV(I) + WAV(I)
      GO TO 502
C* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     TYPE 4 -- BINOMIAL - -
C     TYPE 6 -- POLYNOMIAL ALL -
C     IF ISN NEGATIVE PROBLEM IS ILL-DEFINED OTHERWISE BRANCH TO TYPE 3
  540 IF(ISN) 925,925,5301
  925 ISTOP=925
      GO TO 9007
C*     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     TYPE 7 -- MIXED POLYNOMIAL
C     IF ISN POSITIVE BRANCH TO TYPE 3
C
  570 IF(ISN) 5701,925,5301
C
C     ISN NEGATIVE
C
 5701 IRTN=4
      KPTERM=0
      KQTERM=0
      KEQC=KEQC + 1
      PC(1,KEQC)=1.
      GO TO 510
```

```
 5705 CONTINUE
C
C      CHECK COEF TO DETERMINE IF TERM IS P OR Q
C
       IF(WC) 5710,931,5720
 931   ISTOP=931
       GO TO 9008
 5710  KPTERM=KPTERM+1
       DO 5711 I=1,NVAR
 5711  P(I,KPTERM,KEQC)=WAV(I)
       PC(KPTERM,KEQC)=-WC
       GO TO 5730
 5720  KQTERM=KQTERM+1
       DO 5721 I=1,NVAR
 5721  Q(I,KQTERM,KEQC)=WAV(I)
       QC(KQTERM,KEQC)=WC
 5730  IF(KCOL1.LT.LCOL) GO TO 510
       KPTERM=KPTERM+1
       P(KVAR,KPTERM,KEQC)=1.
       PC(KPTERM,KEQC)=1.
C
C      END OF CONSTRAINT LOGIC
C
C
C      SET LOCATE AND BACK-LOCATE INDECIES FOR CONDENSED CONSTRAINTS
C
 580   CONTINUE
       LOCATE(KEQC)=LCN
       LCTBACK(LCN)=KEQC
       IF(KPTERM.EQ.0) KPTERM=1
       KQT(KEQC)=KQTERM
       KPT(KEQC)=KPTERM
       KTERM=0
             GO TO 502
C
C*      * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C*      * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C      ANALYSIS COMPLETE -- PRINT RESULTS
C
 700   CONTINUE
       KT(1)=NTERM
       NTERM=0
       NEQ=LCN+1
       IDSH=4H----
       WRITE (IOT, 1699) (IDSH,NV=1,NVAR)
 1699  FORMAT(1X,4H---- ,10(A4,8H-------- ))
       WRITE (IOT, 1700) (VNAME(I),I=1,NVAR)
 1700  FORMAT(/5X,10(4XA8))
       DO 750 I=1,NEQ
```

```
      WRITE (IOT, 1699) (IDSH,NV=1,NVAR)
      WRITE (IOT, 17Ø1) I
 17Ø1 FORMAT(3HØEQ ,1X , I2)
      K=KT(I)
      IF(K.EQ.Ø) GO TO 73Ø
      NTERM=NTERM+K
      DO 72Ø J=1,K
      WRITE (IOT, 17Ø5) C(J,I)
 17Ø5 FORMAT(/2XE12.4)
      WRITE (IOT, 17Ø2) (A(NV,J,I),NV=1,NVAR)
 17Ø2 FORMAT(5X,(1ØE12.4))
 72Ø  CONTINUE
      GO TO 75Ø
 73Ø  KEQC=LCTBACK(I-1)
      K=KPT(KEQC)
      KT(I)=K
      NTERM=NTERM+K
      WRITE (IOT, 17Ø3)
 17Ø3 FORMAT(3HØP- )
      DO 735 J=1,K
      WRITE (IOT, 17Ø5) PC(J,KEQC)
      WRITE (IOT, 17Ø2) (P(NV,J,KEQC),NV=1,NVAR)
 735  CONTINUE
      WRITE (IOT, 17Ø4)
 17Ø4 FORMAT(3HØQ- )
      K=KQT(KEQC)
      DO 736 J=1,K
      WRITE (IOT, 17Ø5) QC(J,KEQC)
      WRITE (IOT, 17Ø2) (Q(NV,J,KEQC),NV=1,NVAR)
 736  CONTINUE
 75Ø  CONTINUE
C
C     DETERMINE DEGREE OF DIFICULTY
C
C     IF D.D. LESS THAN ZERO SOLVE PROBLEM
C
      IDD=NTERM-(NVAR+1)
      WRITE (IOT, 176Ø) IDD
 176Ø FORMAT(/1ØX,2ØHDEGREE OF DIFICULTY=    ,I3)
      IF(IDD) 799,8ØØ,76Ø
 76Ø  CONTINUE
      WRITE (IOT, 1761)
 1761 FORMAT(1ØX18HPROBLEM TERMINATED     )
      GO TO 81ØØ
 799  CONTINUE
      WRITE (IOT, 1799)
 1799 FORMAT(1ØX19HNOTE NEGATIVE D. D.        /)
C*     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     BEGIN SOLUTION OF PROBLEM
C
```

```
 800  CONTINUE
      IF(LOCATE(1).EQ.0) GO TO 8015
C
C     INITIALIZE SOLUTION VECTOR -- X(I)
C
      XDEL=NVAR
      DO 8001 I=1,NVAR
 8001 X(I)=XDEL**(I)
      TECL=10.**30
C*     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     DETERMINE A FROM P,Q, AND X FOR EACH CONDENSED CONSTRAINT KEQC
C
 8002 KEQC=0
 8003 KEQC=KEQC+1
      LCN=LOCATE(KEQC)
      IF(LCN.LE.0) GO TO 8015
      K=LCN+1
      KQ=KQT(KEQC)
      KTERM=KPT(KEQC)
      TQ=0.
      DO 8004 I=1,NVAR
 8004 WAV(I)=0.
      DO 8007 I=1,KQ
      DAC=1.
      DO 8005 NV=1,NVAR
      DAC=DAC*(X(NV)**Q(NV,I,KEQC))
 8005 CONTINUE
      DAC=DAC*QC(I,KEQC)
      SQ(I)=DAC
 8007 TQ=TQ+DAC
      WC=1.
      DO 8010 I=1,KQ
      DAC=SQ(I)/TQ
      WC=WC*(QC(I,KEQC)/DAC)**DAC
      DO 8009 NV=1,NVAR
 8009 WAV(NV)=WAV(NV) + (Q(NV,I,KEQC)*DAC)
 8010 CONTINUE
      DO 8012 NT=1,KTERM
      C(NT,K)=PC(NT,KEQC)/WC
      DO 8012 NV=1,NVAR
 8012 A(NV,NT,K)=P(NV,NT,KEQC) - WAV(NV)
      GO TO 8003
C*     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     A DETERMINED NOW BUILD SQUARE MATRIX CDEL FROM A
C
 8015 CONTINUE
      CALL CALC1(CDEL,NTERM,KT,NVEC,A,NVAR,NEQ)
C*     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
```

```
C      INVERT CDEL TO DETERMINE DELTAS AS FIRST ROW IN INVERSE
C
       CALL INVERT(CDEL,NTERM,NTERM,DET)
       IF(DET.EQ.0.) GO TO 9004
       CALL CALDELT(CDEL,NTERM,DELTA)
       WRITE (IOT, 1699) (IDSH,I=1,NTERM)
C
C      PRINT DELTAS
C
       WRITE (IOT, 8029) (DELTA(NT),NT=1,NTERM)
 8029 FORMAT(/10X,20HDELTAS DETERMINED AS  /(5X,10E12.4))
       IF(CDEL(1,1).GT.0.) GO TO 8031
       WRITE(IOT,8030)
 8030 FORMAT(10X,'NOTE DELTAS LESS/EQ ZERO'/10X,'JOB ABORTED')
       GO TO 8100
 8031 CONTINUE
C*      * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C      DETERMINE A LINEAR INDEPENDENT SET FROM A TO SOLVE FOR X
C
       CALL CALC2(CDEL,NVAR,KT,NEQ,NVEC,WAV,A)
C*      * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C      INVERT NEW CDEL TO DETERMINE X
C
       CALL INVERT(CDEL,NVAR,NVAR,DET)
       IF(DET.EQ.0.) GO TO 9009
C
C      DETERMINE TEC FROM DELTAS AND COEF VECTOR
C
       TEC=1.
       K=KT(1)
       DO 8040 NT=1,K
       DAC=DELTA(NT)
       IF(DAC.EQ.0.) GO TO 8040
       TEC=TEC*(C(NT,1)/DAC)**DAC
 8040 CONTINUE
       IF(NEQ.EQ.1) GO TO 8045
       IDEL=K
       DO 8044 KEQ=2,NEQ
       K=KT(KEQ)
       XDEL=0.
       DO 8042 NT=1,K
       IDEL=IDEL+1
       DAC=DELTA(IDEL)
       IF(DAC.EQ.0.) GO TO 8042
       TEC=TEC*(C(NT,KEQ)/DAC)**DAC
       XDEL=XDEL+DAC
 8042 CONTINUE
       IF(XDEL.EQ.0) GO TO 8044
       TEC=TEC*XDEL**XDEL
```

```
 8044 SUMDEL(KEQ)=XDEL
 8045 CONTINUE
C*     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C      RECOMPUTE  B(J) MATRIX
C
      K=KT(1)
      J=0
      DO 8050 NT=1,K
      IF(NOTEQ.EQ.1.AND.NOTTM.EQ.NT) GO TO 8050
      J=J+1
      B(J)=ALOG(TEC*DELTA(NT)/C(NT,1))
 8050 CONTINUE
      IF(NEQ.EQ.1) GO TO 8053
      I=K
      DO 8052 KEQ=2,NEQ
      XDEL=SUMDEL(KEQ)
      K=KT(KEQ)
      DO 8052 NT=1,K
      I=I+1
      IF(NOTEQ.EQ.KEQ.AND.NOTTM.EQ.NT) GO TO 8052
      J=J+1
      B(J)=0.
      IF(XDEL.EQ.0.) GO TO 8052
      B(J)=ALOG(DELTA(I)/(XDEL*C(NT,KEQ)))
 8052 CONTINUE
 8053 CONTINUE
C*     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C      SOLVE FOR X FROM CDEL INVERSE AND RECOMPUTED B(J)
C
      CALL CALCX(CDEL,NVAR,B,X)
C
C      PRINT TRIAL SOLUTION
C
      WRITE (IOT, 1699) (IDSH,NV=1,NVAR)
      WRITE (IOT, 8057) TECN,TEC
 8057 FORMAT(/10X,20HTRIAL SOLUTION FOUND ,10X,A8,5X,E12.4)
      WRITE (IOT, 1700) (VNAME(NV),NV=1,NVAR)
      WRITE (IOT, 1699) (IDSH,NV=1,NVAR)
      WRITE (IOT, 1702) (X(NV),NV=1,NVAR)
C
C      CHECK FOR CONVERGENCE
C      IF NOT DONE RETURN TO 8002 TO RECOMPUTE A FROM X,P,AND Q
C
      IF(KPT(1).EQ.0) GO TO 8100
      XDEL=TECL/TEC
      TECL=TEC
      IF(XDEL.GT.LIMIT) GO TO 8002
C
C      PRINT EOJ
```

```
      WRITE (IOT, 8060) XDEL,LIMIT
 8060 FORMAT(/10X,21HSOLUTION IMPROVEMENT=  F13.7/10X,
     + 16HLESS THAN LIMIT=  F20.7)
      WRITE (IOT, 1699) (IDSH,I=1,NTERM)
 8100 CONTINUE
      WRITE(ITTY,8101)
8101  FORMAT(//1X,'END OF JOB'/1X,'DO YOU WISH A RERUN?')
      PAUSE
      GO TO 6
C*     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 998  CONTINUE
 999  CONTINUE
      ISTOP=999
      GO TO 9005
C*     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C      ERROR PROCESSING ROUTINES
C
C
C*     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 9001 CONTINUE
      WRITE(ITTY,9051) NCAR
 9051 FORMAT(' CARD FORMAT ERROR -- CARD NUMBER',I3)
      GO TO 9900
 9002 CONTINUE
      KCOL=KCOL+6
      WRITE(ITTY,9052) KCOL,ISTOP
 9052 FORMAT(' SYNTAX ERROR IN COL ',I4/' DETECTED FROM STATEMENT',I6)
      GO TO 9900
 9003 CONTINUE
      WRITE(ITTY,9053)
 9053 FORMAT(' PARENS DO NOT CLOSE ')
      GO TO 9900
 9004 CONTINUE
      WRITE(ITTY,9054)
 9054 FORMAT(' DETERMINANT ZERO FOR DELTA SOLN')
      GO TO 9900
 9005 CONTINUE
      WRITE(ITTY,9055) ISTOP
 9055 FORMAT(' STATEMENT INCOMPLETE - DETECTED FROM' I6)
      GO TO 9900
 9006 CONTINUE
      WRITE(ITTY,9056) KTYPE(LCN),LCN
 9056 FORMAT(' ILLEGAL CONSTRAINT TYPE'/' TYPE'I3,'LCN ' ,I3)
      GO TO 9900
 9007 CONTINUE
      WRITE(ITTY,9057)
 9057 FORMAT(' ISN LE ZERO FOR NEGATIVE POLYNOMIAL')
      GO TO 9900
 9008 CONTINUE
      WRITE(ITTY,9058) ISTOP
```

```
9058 FORMAT(' COEF EQUAL TO ZERO FROM LINE',I6)
     GO TO 9900
9009 CONTINUE
     WRITE(ITTY,9059)
9059 FORMAT(' DETERMINANT ZERO FOR VARIABLE SOLN')
     GO TO 9900
9900 CONTINUE
     WRITE(ITTY,9901)
9901 FORMAT(' JOB ABORTED')
     GO TO 8100
     END
```

```
      SUBROUTINE CALC1(CDEL,NTERM,KT,NVEC,A,NVAR,NEQ)
      DIMENSION CDEL(NTERM,NTERM)
      DIMENSION A(20,20,10),KT(1),NVEC(1)
      IOT=4
      K=KT(1)
C
C      FIRST ROW IS NORMALITY CONSTRAINT
C
      DO 8020 NT=1,K
 8020 CDEL(1,NT)=1.
 8021 K=K+1
      IF(K.GT.NTERM) GO TO 8022
      CDEL(1,K)=0.
      GO TO 8021
 8022 CONTINUE
C
C      SUBSEQUENT ROWS ARE ORTHOGANALITY CONSTRAINTS
C      ONE ROW FOR EACH VARIABLE
C
      DO 8025 NV=1,NVAR
      NVEC(NV)=0
      J=NV+1
      IF(J.GT.NTERM) GO TO 199
      I=0
      DO 8025 KEQ=1,NEQ
      K=KT(KEQ)
      DO 8025 NT=1,K
      I=I + 1
 8025 CDEL(J,I)=A(NV,NT,KEQ)
 199   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE CALC2(CDEL,NVAR,KT,NEQ,NVEC,WAV,A)
      COMMON /NOT/ NOTTM,NOTEQ
      DIMENSION CDEL(NVAR,NVAR)
      DIMENSION A(20,20,10),NVEC(1),WAV(1),KT(1)
      IFLG2=0
      J=0
      DO 8038 KEQ=1,NEQ
      K=KT(KEQ)
      DO 8038 NT=1,K
      IFLG=0
      DO 8033 NV=1,NVAR
      WAV(NV)=A(NV,NT,KEQ)
      IF(IFLG2.EQ.1) GO TO 8031
      IF(WAV(NV).EQ.0.) GO TO 8033
      IF(NVEC(NV).EQ.1) GO TO 8033
      NVEC(NV)=1
 8031 IFLG=1
 8033 CONTINUE
      IF(IFLG.EQ.1) GO TO 8035
      NOTTM=NT
      NOTEQ=KEQ
      IFLG2=1
      GO TO 8038
 8035 J=J+1
      IF(J.GT.NVAR) RETURN
      DO 8037 NV=1,NVAR
 8037 CDEL(J,NV)=WAV(NV)
 8038 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE CALDELT(CDEL,NTERM,DELTA)
      DIMENSION CDEL(NTERM,NTERM),DELTA(1)
      IOT=4
      J=0
      DO 8027 NT=1,NTERM
      DELTA(NT)=CDEL(NT,1)
      IF(DELTA(NT).LE.0.) J=J-1
8027  CONTINUE
       IF(J.LT.0) CDEL(1,1)=J
      RETURN
      END
```

```
      SUBROUTINE CALCX(CDEL,NVAR,B,X)
      DIMENSION CDEL(NVAR,NVAR),X(1),B(1)
C*    * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     SOLVE FOR X FROM CDEL INVERSE AND RECOMPUTED B(J)
C
      DO 8055 NV=1,NVAR
      WC=0.
      DO 8054 K=1,NVAR
 8054 WC=WC+(CDEL(NV,K)*B(K))
 8055 X(NV)=EXP(WC)
      RETURN
      END
```

```
      SUBROUTINE GETTEC(KCOL,TECN)
      INTEGER VNAME,SHFT
      INTEGER TECN
      COMMON NVAR,NVARPG,IPFLG(20)
      COMMON KQT(10),KPT(10),KT(10)
      COMMON A(20,20,10),C(20,10),P(10,10,10),Q(10,10,10),PC(10,10),
     + QC(10,10)
      COMMON CARD(16),KARD(198)
      COMMON VNAME(20)
      COMMON WAV(20),SAV(20)
      COMMON LEVEL(10),LL2(10),IPRNO(10),IPRNC(10),KTYPE(10),LOCATE(10),
     + KNSTVAR(10),L2TAB(10)
      INTEGER E
      DATA E,IB/"75,"100/
      I=0
      KCOL=0
      TECN=0
1     CALL BUMPCOL(KCOL)
      IF(KCOL.LE.0) GO TO 100
      IF(KARD(KCOL).EQ.E) GO TO 50
      I=I+1
      IF(I.GT.5) GO TO 1
      TECN=SHFT(TECN,7).OR.SHFT(KARD(KCOL),1)
      GO TO 1
50    IF(I.GE.5) RETURN
      TECN=SHFT(TECN,7).OR.IB
      I=I+1
      GO TO 50
100   CONTINUE
      WRITE (4,1000)
1000 FORMAT(1X,'NO EQUALS SIGN IN EQUATION -- JOB ABORTED')
      PAUSE 7777
      END
```

```
      SUBROUTINE BUMPCOL(KOL)
      INTEGER BLANK
      COMMON /LIM/ LIMIT
      INTEGER VNAME
      COMMON NVAR,NVARPG,IPFLG(20)
      COMMON KQT(10),KPT(10),KT(10)
      COMMON A(20,20,10),C(20,10),P(10,10,10),Q(10,10,10),PC(10,10),
     + QC(10,10)
      COMMON CARD(16),KARD(198)
      COMMON VNAME(20)
      COMMON WAV(20),SAV(20)
      COMMON LEVEL(10),LL2(10),IPRNO(10),IPRNC(10),KTYPE(10),LOCATE(10),
     + KNSTVAR(10),L2TAB(10)
      DATA BLANK /"40 /
1     KOL=KOL+1
      IF(KOL.GT.LIMIT) GO TO  5
      IF(KARD(KOL).NE.BLANK) RETURN
      GO TO 1
5     KOL=0
      RETURN
      END
```

```
      SUBROUTINE WHATSIT(KCOL,ITYP,IWD)
      INTEGER VNAME
      COMMON NVAR,NVARPG,IPFLG(20)
      COMMON KQT(10),KPT(10),KT(10)
      COMMON A(20,20,10),C(20,10),P(10,10,10),Q(10,10,10),PC(10,10),
     + QC(10,10)
      COMMON CARD(16),KARD(198)
      COMMON VNAME(20)
      COMMON WAV(20),SAV(20)
      COMMON LEVEL(10),LL2(10),IPRNO(10),IPRNC(10),KTYPE(10),LOCATE(10),
     + KNSTVAR(10),L2TAB(10)
      INTEGER PLUS,MINUS,SLSH,STAR,OPEN,CLOSE
      INTEGER POINT
      DATA PLUS,MINUS,SLSH,STAR/"53,"55,"57,"52/
      DATA OPEN,CLOSE/"50,"51/
      DATA POINT/"56/
      KCOL1=KCOL
      IAFLG=0
      IWD=1
4     KAR=KARD(KCOL1)
      IF(KAR.GT."71) GO TO 70
      IF(KAR-PLUS) 10,5,10
5     IT=4
      GO TO 100
10    IF(KAR-MINUS) 20,15,20
15    IT=5
      GO TO 100
20    IF(KAR-SLSH) 30,25,30
25    IT=6
      GO TO 100
30    IF(KAR-OPEN) 40,35,40
35    IT=3
      GO TO 100
40    IF(KAR-CLOSE) 50,45,50
45    IT=9
      GO TO 100
50    IF(KAR-STAR) 60,55,60
55    IT=7
      IF(IAFLG.EQ.1) GO TO 75
      IF(KARD(KCOL1+1).NE.STAR) GO TO 100
      IT=8
      IWD=2
      GO TO 100
60    CONTINUE
      IF(KAR.EQ.POINT) GO TO 61
      IF(KAR.GT."71.OR.KAR.LT."60) GO TO 70
      IF(IAFLG.EQ.1) GO TO 71
61    IWD= KCOL1 - KCOL + 1
      CALL BUMPCOL(KCOL1)
      IF(KCOL1.LE.0) GO TO 62
      KAR=KARD(KCOL1)
```

```
      IF(KAR.EQ.POINT) 61,610
610   IF(KAR.GE."60.AND.KAR.LE."71) 61,62
62    ITYP=2
      RETURN
65    ITYP=10
      RETURN
70    IAFLG=1
71    CONTINUE
      IWD=KCOL1-KCOL +1
      CALL BUMPCOL(KCOL1)
      IF(KCOL1) 75,75,4
75    ITYP=1
      RETURN
100   IF(IAFLG.NE.0) GO TO 75
      ITYP=IT
      RETURN
      END
```

```
      SUBROUTINE ISITYET(KCOL,IVAR,IWD)
      INTEGER VNAME,SHFT
      COMMON NVAR,NVARPG,IPFLG(20)
      COMMON KQT(10),KPT(10),KT(10)
      COMMON A(20,20,10),C(20,10),P(10,10,10),Q(10,10,10),PC(10,10),
     + QC(10,10)
      COMMON CARD(16),KARD(198)
      COMMON VNAME(20)
      COMMON WAV(20),SAV(20)
      COMMON LEVEL(10),LL2(10),IPRNO(10),IPRNC(10),KTYPE(10),LOCATE(10),
     + KNSTVAR(10),L2TAB(10)
      IF(NVARPG.EQ.0) GO TO 101
      DO 1 I=1,NVARPG
      IF(KCOL.EQ.IPRNO(I)) GO TO 20
1     CONTINUE
101   CONTINUE
      K=KCOL-1
      ITEMP=0
      DO 5 I=1,5
      IF(I.GT.IWD) GO TO 2
      K=K+1
      ITEMP=SHFT(ITEMP,7).OR.SHFT(KARD(K),1)
      GO TO 5
2     ITEMP=SHFT(ITEMP,7).OR."100
5     CONTINUE
      IVAR=0
10    IVAR=IVAR+1
      IF(IVAR.GT.NVAR) GO TO 15
      IF(ITEMP.EQ.VNAME(IVAR)) RETURN
      GO TO 10
15    VNAME(IVAR)=ITEMP
      NVAR=IVAR
      IPFLG(NVAR)=0
      RETURN
20    IVAR=KNSTVAR(I)
      IWD=IPRNC(I)-KCOL + 1
      RETURN
      END
```

```
      SUBROUTINE GETPGV(KCOL,LCOL)
      INTEGER VNAME
      COMMON NVAR,NVARPG,IPFLG(20)
      COMMON KQT(10),KPT(10),KT(10)
      COMMON A(20,20,10),C(20,10),P(10,10,10),Q(10,10,10),PC(10,10),
     + QC(10,10)
      COMMON CARD(16),KARD(198)
      COMMON VNAME(20)
      COMMON WAV(20),SAV(20)
      COMMON LEVEL(10),LL2(10),IPRNO(10),IPRNC(10),KTYPE(10),LOCATE(10),
     + KNSTVAR(10),L2TAB(10)
      NVAR=NVAR+1
      NVARPG=NVARPG+1
      IPRNO(NVARPG)=KCOL
      IPRNC(NVARPG)=LCOL
      ENCODE(5,1000,NAME) NVARPG
1000  FORMAT(3HPGV,I2)
5     VNAME(NVAR)=NAME
      IPFLG(NVAR)=1
      RETURN
      END
```

```
      SUBROUTINE UNCODE(KCOL,VAR,IWD)
      INTEGER VNAME
      COMMON NVAR,NVARPG,IPFLG(20)
      COMMON KQT(10),KPT(10),KT(10)
      COMMON A(20,20,10),C(20,10),P(10,10,10),Q(10,10,10),PC(10,10),
     + QC(10,10)
      COMMON CARD(16),KARD(198)
      COMMON VNAME(20)
      COMMON WAV(20),SAV(20)
      COMMON LEVEL(10),LL2(10),IPRNO(10),IPRNC(10),KTYPE(10),LOCATE(10),
     + KNSTVAR(10),L2TAB(10)
      INTEGER POINT
      DATA POINT /"56/
      KCOL1=KCOL
      LCOL=KCOL1+IWD-1
      DAC=0.
      IAC=0
      DO 1 I=KCOL1,LCOL
      N=KARD(I) - "60
      IF(N.GT.9.OR.N.LT.0) GO TO 5
      IAC=IAC*10 + N
1     CONTINUE
      GO TO 10
5     IF(KARD(I).NE.POINT) GO TO 20
      J=0
6     I=I+1
      IF(I.GT.LCOL) GO TO 10
      N=KARD(I) - "60
      IF(N.GT.9.OR.N.LT.0) GO TO 20
      J=J-1
      T=N
      DAC=DAC+T*10.**J
      GO TO 6
10    VAR=IAC
      VAR=VAR + DAC
      RETURN
20    VAR=0.
      RETURN
      END
```

```
TITLE   SHFT
ENTRY   SHFT
SHFT:   Z
        MOVE    0,@0(16)
        MOVE    1,@1(16)
        LSH     0,0(1)
        JRA     16,2(16)
        END
```

```
      SUBROUTINE INVERT(A, N, M, DET)
C
C     A SUBPROGRAM FOR THE INVERSION OF A SQUARE MATRIX OF ORDER N USING
C     THE MAXIMUM PIVOT STRATEGY MODIFICATION OF THE GAUSS-JORDAN
C     ELIMINATION ALGORITHM. AS A BY-PRODUCT OF THE ELIMINATION
C     PROCEDURE, THE VALUE OF THE DETERMINANT IS ALSO PROVIDED. IN CASE
C     THE MATRIX IS DETERMINED TO BE SINGULAR, A VALUE OF ZERO IS RETURNED
C     FOR THE DETERMINANT AND THE ARRAY CONTAINS THE PARTIAL INVERSION UP
C     TO THE POINT WHERE SINGULARITY IS DETECTED.
C
C     PROVISION IS MADE FOR OPERATING ON A RECTANGULAR MATRIX. AS LONG AS
C     THE CALL TO THIS SUBPROGRAM SPECIFIES THAT THE NUMBER OF COLUMNS IS
C     NOT LESS THAN THE NUMBER OF ROWS IN THE ARRAY, THE PIVOT OPERATIONS
C     WILL BE ALLOWED. IF THE NUMBER OF COLUMNS EXCEEDS THE NUMBER OF
C     ROWS, THE ROUTINE WILL INVERT THE FIRST N COLUMNS WHILE CARRYING THE
C     PIVOT OPERATIONS INTO THE REMAINING M-N COLUMNS SO THAT IF THESE
C     COLUMNS CONTAIN THE RIGHT HAND SIDES OF SYSTEMS OF EQUATIONS, THE
C     SOLUTIONS OF THE EQUATIONS WILL APPEAR IN THESE M-N COLUMNS UPON
C     RETURN TO THE CALLING PROGRAM.
C
C     THE MAXIMUM PIVOT STRATEGY CAN INDUCE AN INTERCHANGE OF EITHER ROWS
C     OR COLUMNS OR BOTH. SINCE COLUMN INTERCHANGES WILL SCRAMBLE THE
C     ELEMENTS OF A SOLUTION VECTOR, AND EITHER ROW OR COLUMN INTERCHANGES
C     WILL SCRAMBLE THE ELEMENTS OF THE INVERSE, A PORTION OF THIS ROUTINE
C     IS DEVOTED TO UNSCRAMBLING THE FINAL ARRAY. FINALLY, SINCE ROW OR
C     COLUMN INTERCHANGES AFFECT THE SIGN OF THE DETERMINANT, THE NUMBER
C     OF INTERCHANGES IS CHECKED AND THE SIGN IS CHANGED ACCORDINGLY.
C
C     CALLING SEQUENCE
C
C     A       INPUT    THE ARRAY CONTAINING THE MATRIX TO BE INVERTED.
C             OUTPUT   THE RESULTS OF THE INVERSION ATTEMPT.
C
C     N       INPUT    THE NUMBER OF ROWS IN THE ARRAY.
C
C     M       INPUT    THE NUMBER OF COLUMNS IN THE ARRAY.
C
C     DET     OUTPUT   THE VALUE OF THE DETERMINANT OF THE FIRST N COLUMNS
C                      OF THE INPUT MATRIX. IF A SINGULARITY IS DETECTED,
C                      A MACHINE ZERO IS RETURNED.
C
C     LOCAL VARIABLES
C
C     R, C             VECTORS CONTAINING, RESPECTIVELY, THE ROW AND COLUMN
C                      INDICES OF THOSE ARRAY ELEMENTS SELECTED FOR PIVOTS.
C
C     S                AN AUXILIARY VECTOR USED IN THE UNSCRAMBLING AND
C                      SIGN CHECK PORTIONS.
C
C     I, J, K, L       DO LOOP INDICES.
C
```

```
C    TEST              CONTAINS TEST VALUES DURING  THE  SELECTION  OF  THE
C                      PIVOT.   ALSO  USED  AS TEMPORARY STORAGE DURING THE
C                      SIGN CHECK PORTION.
C
C    SUBPROGRAMS REQUIRED
C
C    RONORM  FUNCTION
C    PIVOT   SUBROUTINE
C
       DIMENSION A(N,M), S(50)
       INTEGER R(50), C(50)
       DATA TOL/0.001/
C
C    INITIALIZE THE VALUE OF THE DETERMINANT.
       DET = 1.
C
C    START THE INVERSION PROCESS FOR AT MOST N CYCLES.
       DO 40 K = 1,N
C
C    START THE SELECTION PROCEDURE.
       TEST = 0.
C
C    THE SELECTION PROCEDURE INVESTIGATES ALL N ROWS
C    BUT ONLY THE FIRST N COLUMNS.
       DO 30 I = 1,N
       DO 20 J = 1,N
C
C    FOR THE FIRST CYCLE EVERY ELEMENT IN THE N*N ARRAY
C    IS A LIKELY CANDIDATE FOR THE PIVOT.
       IF(K .EQ. 1) GO TO 15
C
C    OTHERWISE, AVOID THOSE ROW AND COLUMNS ALREADY
C    USED AS PIVOTS.
       DO 10 L = 1,K-1
C
C    FIRST CHECK IF A PIVOT HAS OCCURED IN ROW I.
       IF(I .EQ. R(L)) GO TO 30
C
C    THEN CHECK COLUMN J.
 10    IF(J .EQ. C(L)) GO TO 20
C
C    AT STATEMENT 15 I AND J LOCATE A CANDIDATE FOR THE
C    PIVOT.  CHECK IF IT IS THE LARGEST SO FAR.
 15    IF(ABS(A(I,J)) .LE. TEST) GO TO 20
C
C    IF SO RECORD THE LOCATION AND UPDATE THE TEST
C    VALUE.
       R(K) = I
       C(K) = J
       TEST = ABS(A(I,J))
 20    CONTINUE
```

```
 30   CONTINUE
C
C    AT THIS POINT IN THE PROGRAM R(K) AND C(K) CONTAIN
C    THE ROW AND COLUMN LOCATION OF THE LARGEST PIVOT
C    CANDIDATE.  IF THIS SATISFIES THE TOLERANCE
C    CONDITION, PROCEED WITH THE PIVOT.
       IF(TEST/RONORM(A,N,M) .GT. TOL) GO TO 35
C
C    OTHERWISE, SET THE DETERMINANT TO ZERO AND
C    RETURN WITHOUT UNSCRAMBLING.
       DET = 0.
       RETURN
C
C    BEFORE THE PIVOT UPDATE THE DETERMINANT.
 35    DET = DET*A(R(K),C(K))
 40    CALL PIVOT(A, N, M, R(K), C(K))
C
C    WHEN THE INVERSION IS COMPLETE, UNSCRAMBLE THE
C    ARRAY.
C
C    FIRST BY COLUMNS.
       DO 60 J = 1,M
       DO 50 I = 1,N
 50    S(C(I)) = A(R(I),J)
       DO 60 I = 1,N
 60    A(I,J) = S(I)
C
C    THEN BY ROWS.
       DO 80 I = 1,N
       DO 70 J = 1,N
 70    S(R(J)) = A(I,C(J))
       DO 80 J = 1,N
 80    A(I,J) = S(J)
C
C    NOW SET UP THE S VECTOR FOR THE SIGN CHECK.
       DO 90 K = 1,N
 90    S(R(K)) = C(K)
C
C    CLEAR THE INTERCHANGE COUNTER AND DO A BUBBLE SORT
C    ON THE VECTOR S.
       INT = 0
       DO 100 I = 1,N-1
       DO 100 J = I+1,N
       IF(S(J) .GE. S(I)) GO TO 100
       TEST = S(I)
       S(I) = S(J)
       S(J) = TEST
       INT = INT + 1
 100   CONTINUE
C
C    IF THE NUMBER OF INTERCHANGES REQUIRED TO SORT THE
```

```
C    S VECTOR IS ODD, CHANGE THE SIGN OF THE DETERMINANT.
       IF(INT/2*2 .NE. INT) DET = -DET
       RETURN
       END
```

```
      SUBROUTINE PIVOT(A, M, N, R, C)
C
C   THIS SUBPROGRAM PERFORMS ONE STANDARD PIVOT OPERATION IN PLACE FOR
C   EACH ENTRY TO THE ROUTINE.  THE SELECTION OF THE PIVOT ELEMENT AND
C   TESTING FOR A VALID PIVOT ELEMENT IS ASSUMED TO HAVE BEEN DONE PRIOR
C   TO ENTRY TO THIS SUBPROGRAM.
C
C   CALLING SEQUENCE
C
C   A       INPUT    THE MATRIX TO BE PIVOTED.
C           OUTPUT   CONTAINS THE RESULTS OF THE PIVOT OPERATION.  THE
C                    INPUT MATRIX HAS BEEN ALTERED.
C
C   M       INPUT    THE NUMBER OF ROWS IN THE MATRIX,
C
C   N       INPUT    THE NUMBER OF COLUMNS IN THE MATRIX,
C
C   R       INPUT    THE ROW INDEX OF THE PIVOT ELEMENT,
C
C   C       INPUT    THE COLUMN INDEX OF THE PIVOT ELEMENT.
C
C   LOCAL VARIABLES
C
C   I,J              DO LOOP INDICES.
C
      DIMENSION A(M,N)
      INTEGER R,C
C
C   TRANSFORM THE PIVOT ELEMENT.
      A(R,C) = 1./A(R,C)
C
C   TRANSFORM THE REST OF THE PIVOT ROW.
      DO 10 J = 1,N
   10 IF(J .NE. C) A(R,J) = A(R,J)*A(R,C)
C
C   TRANSFORM ALL ELEMENTS OF THE MATRIX EXCEPT FOR THOSE IN THE PIVOT
C   ROW AND PIVOT COLUMN.
      DO 30 I = 1,M
      IF(I .EQ. R) GO TO 30
      DO 20 J = 1,N
   20 IF(J .NE. C) A(I,J) = A(I,J) - A(I,C)*A(R,J)
   30 CONTINUE
C
C   NOW TRANSFORM THE REST OF THE PIVOT COLUMN.
      DO 40 I = 1,M
   40 IF(I .NE. R) A(I,C) = -A(I,C)*A(R,C)
      RETURN
      END
```

```
      FUNCTION RONORM(A, M, N)
C
C   OF THE SEVERAL MATRIX NORMS AVAILABLE THE ROW NORM WAS SELECTED FOR
C   THIS SUBPROGRAM.  THE ROW NORM IS DETERMINED BY ADDING THE ABSOLUTE
C   VALUES OF THE ELEMENTS IN EACH ROW AND SELECTING THE LARGEST OF
C   THESE SUMS FOR THE VALUE OF THE FUNCTION.
C
C   CALLING SEQUENCE
C
C   A        INPUT    THE MATRIX.
C
C   M        INPUT    THE NUMBER OF ROWS IN THE MATRIX.
C
C   N        INPUT    THE NUMBER OF COLUMNS IN THE MATRIX.
C
C   RONORM   OUTPUT   THE FUNCTION VALUE.
C
C   LOCAL VARIABLES
C
C   I,J               DO LOOP INDICES.
C
C   SUM               FOR COMPUTING THE SEVERAL ROW SUMS.
C
      DIMENSION A(M,N)
C
C   INITIALIZE THE NORM FOR COMPARISON PURPOSES.
      RONORM = 0.
C
C   COMPUTE M SUMS BY ROWS.
      DO 20 I = 1,M
C
C   INITIALIZE THE SUM.
      SUM = 0.
C
C   COMPUTE A SUM.
      DO 10 J = 1,N
 10   SUM = SUM + ABS(A(I,J))
C
C   CHECK IF THIS SUM IS THE LARGEST SO FAR.  IF SO, UPDATE THE FUNCTION
C   VALUE.
 20   IF(SUM .GT. RONORM) RONORM = SUM
C
C   WHEN THE I LOOP IS SATISFIED, RONORM CONTAINS THE LARGEST SUM
C   COMPUTED.
      RETURN
      END
```

```
     PROBLEM NO.  1
C PROBLEMS NOT REQUIRING CONDENSATION
       COST = 28333*TD**.5 + 70000*TD**-.5


-------------
           TD


---------------
EQ  1
  0.2833E+05
     0.5000E+00
  0.7000E+05
    -0.5000E+00
        DEGREE OF DIFICULTY=  0


-------------------------
        DELTAS DETERMINED AS
     0.5000E+00  0.5000E+00


-------------
        TRIAL SOLUTION FOUND              COST         0.8907E+05
          TD


-------------
     0.2471E+01
```

```
     PROBLEM NO.  2
       COST = 326.2*S**.5 + 100000000*S**-.5/P + 34.4*P


-------------------------
           S             P


-------------------------
EQ  1
  0.3262E+03
     0.5000E+00  0.0000E+00
  0.1000E+09
    -0.5000E+00 -0.1000E+01
  0.3440E+02
     0.0000E+00  0.1000E+01
        DEGREE OF DIFICULTY=  0


-------------------------------------
        DELTAS DETERMINED AS
     0.3333E+00  0.3333E+00  0.3333E+00


-------------------------
        TRIAL SOLUTION FOUND              COST        0.3117E+05
          S             P


-------------------------
     0.1015E+04  0.3021E+03
```

```
     PROBLEM NO.  3
       MIN = 1/(15.8*X - .47*X**2)


-------------------------
           PGV 1        X


-------------------------
EQ  1
  0.1000E+01
    -0.1000E+01  0.0000E+00


-------------------------
EQ  2
  0.6329E-01
     0.1000E+01 -0.1000E+01
  0.2975E-01
     0.0000E+00  0.1000E+01
        DEGREE OF DIFICULTY=  0


-------------------------------------
        DELTAS DETERMINED AS
     0.1000E+01  0.1000E+01  0.1000E+01


-------------------------
        TRIAL SOLUTION FOUND              MIN          0.7531E-02
          PGV 1        X


-------------------------
     0.1328E+03  0.1681E+02
```

```
     PROBLEM NO.  4
       MIN = 1/ (9000*TD**.5/(TD+.75))

-------------------------------------------
           PGV 1        PGV 2        TD

-------------------------------------------
EQ  1
  0.1000E+01
     0.0000E+00 -0.1000E+01  0.0000E+00

-------------------------------------------
EQ  2
  0.1000E+01
    -0.1000E+01  0.0000E+00  0.1000E+01
  0.7500E+00
    -0.1000E+01  0.0000E+00  0.0000E+00

-------------------------------------------
EQ  3
  0.1111E-03
     0.1000E+01  0.1000E+01 -0.5000E+00
        DEGREE OF DIFICULTY=  0

-------------------------------------------------------
        DELTAS DETERMINED AS
     0.1000E+01  0.5000E+00  0.5000E+00  0.1000E+01

-------------------------------------------
        TRIAL SOLUTION FOUND              MIN          0.1925E-03
          PGV 1        PGV 2        TD

-------------------------------------------
     0.1500E+01  0.5196E+04  0.7500E+00
```

```
     PROBLEM NO.  5
C ***** CONDENSED POSYNOMIALS *****
       CI = U + 1000/D + 100000*D/(D+U)

-------------------------------------------
           U             D             PGV 1

-------------------------------------------
EQ  1
  0.1000E+01
     0.1000E+01  0.0000E+00  0.0000E+00
  0.1000E+04
     0.0000E+00 -0.1000E+01  0.0000E+00
  0.1000E+06
     0.0000E+00  0.1000E+01 -0.1000E+01

-------------------------------------------
EQ  2
P=
  0.1000E+01
     0.0200E+00  0.0000E+00  0.1000E+01
Q=
  0.1000E+01
     0.0000E+00  0.1000E+01  0.0000E+00
  0.1000E+01
     0.1000E+01  0.0000E+00  0.0000E+00
        DEGREE OF DIFICULTY=  0

-------------------------------------------------------
        DELTAS DETERMINED AS
     0.1667E+00  0.1667E+00  0.6667E+00  0.6667E+00

-------------------------------------------
        TRIAL SOLUTION FOUND              CI          0.1115E+05
          U             D             PGV 1

-------------------------------------------
     0.1858E+04  0.5381E+00  0.7238E+01

-------------------------------------------------------
        DELTAS DETERMINED AS
     0.3333E+00  0.3333E+00  0.3334E+00  0.3334E+00

-------------------------------------------
        TRIAL SOLUTION FOUND              CI          0.1392E+04
          U             D             PGV 1

-------------------------------------------
     0.4639E+03  0.2155E+01  0.4645E+03
```

```
-------------------------------------------------------
        DELTAS DETERMINED AS
     Ø.3328E+ØØ  Ø.3328E+ØØ  Ø.3344E+ØØ  Ø.3344E+ØØ


-------------------------------------------
        TRIAL SOLUTION FOUND              CI          Ø.139ØE+Ø4
          U            D           PGV 1


-------------------------------------------
     Ø.4627E+Ø3  Ø.2161E+Ø1  Ø.4649E+Ø3
        SOLUTION IMPROVEMENT=     1.ØØ11872
        LESS THAN LIMIT=            1.Ø1ØØØØ


-------------------------------------------------------
```

```
     PROBLEM NO.  6
       CII = U**.5 + 1000*D**-.5 + 100000*D/(D+U)

-------------------------------------------
           U             D             PGV 1


-------------------------------------------
EQ  1
  0.1000E+01
     0.5000E+00  0.0000E+00  0.0000E+00
  0.1000E+04
     0.0000E+00 -0.5000E+00  0.0000E+00
  0.1000E+06
     0.0000E+00  0.1000E+01 -0.1000E+01


-------------------------------------------
EQ  2
P-
  0.1000E+01
     0.0000E+00  0.0000E+00  0.1000E+01
Q-
  0.1000E+01
     0.0000E+00  0.1000E+01  0.0000E+00
  0.1000E+01
     0.1000E+01  0.0000E+00  0.0000E+00
        DEGREE OF DIFICULTY=  0


-------------------------------------------------------
        DELTAS DETERMINED AS
     0.2500E+00  0.2500E+00  0.5000E+00  0.5000E+00


-------------------------------------------
        TRIAL SOLUTION FOUND              CII          0.3797E+04
          U             D             PGV 1


-------------------------------------------
     0.9011E+06  0.1110E+01  0.5846E+02


-------------------------------------------------------
        DELTAS DETERMINED AS
     0.4000E+00  0.4000E+00  0.2000E+00  0.2000E+00


-------------------------------------------
        TRIAL SOLUTION FOUND              CII          0.4551E+03
          U             D             PGV 1


-------------------------------------------
     0.3314E+05  0.3017E+02  0.3314E+05
```

```
-------------------------------------------------------
        DELTAS DETERMINED AS
     0.3999E+00  0.3999E+00  0.2001E+00  0.2001E+00


-------------------------------------------------------
        TRIAL SOLUTION FOUND              CII          0.4551E+03
          U            D            PGV 1


-------------------------------------------------------
     0.3312E+05  0.3019E+02  0.3315E+05
        SOLUTION IMPROVEMENT=     1.00018804
        LESS THAN LIMIT=            1.01000000


-------------------------------------------------------
```

```
     PROBLEM NO.  7
       CIII = U**2 + 1000*D**-2 + 100000*D/(D+U)

-------------------------------------------
           U             D            PGV 1


-------------------------------------------
EQ  1
  0.1000E+01
     0.2000E+01  0.0000E+00  0.0000E+00
  0.1000E+04
     0.0000E+00 -0.2000E+01  0.0000E+00
  0.1000E+06
     0.0000E+00  0.1000E+01 -0.1000E+01


-------------------------------------------
EQ  2
P=
  0.1000E+01
     0.0000E+00  0.0000E+00  0.1000E+01
Q=
  0.1000E+01
     0.0000E+00  0.1000E+01  0.0000E+00
  0.1000E+01
     0.1000E+01  0.0000E+00  0.0000E+00
        DEGREE OF DIFICULTY=  0


-------------------------------------------------------
        DELTAS DETERMINED AS
     0.1000E+00  0.1000E+00  0.8000E+00  0.8000E+00


-------------------------------------------
        TRIAL SOLUTION FOUND              CIII        0.2411E+05
          U             D            PGV 1


-------------------------------------------
     0.4910E+02  0.6441E+00  0.3339E+01


-------------------------------------------------------
        DELTAS DETERMINED AS
     0.2484E+00  0.2484E+00  0.5033E+00  0.5033E+00


-------------------------------------------
        TRIAL SOLUTION FOUND              CIII        0.4976E+04
          U             D            PGV 1


-------------------------------------------
     0.3515E+02  0.8996E+00  0.3592E+02
```

```
-------------------------------------------------------
        DELTAS DETERMINED AS
    0.2468E+00  0.2468E+00  0.5063E+00  0.5063E+00

-------------------------------------------------------
        TRIAL SOLUTION FOUND                CIII        0.4966E+04
          U             D             PGV 1

-------------------------------------------------------
    0.3501E+02  0.9032E+00  0.3592E+02
        SOLUTION IMPROVEMENT=     1.0018345
        LESS THAN LIMIT=            1.0100000

-------------------------------------------------------
```

```
     PROBLEM NO.  8
       CIV = 100*U + 10000/D + 1000*D/(D+U)

-------------------------------------------
           U             D            PGV 1


-------------------------------------------
EQ  1
  0.1000E+03
     0.1000E+01  0.0000E+00  0.0000E+00
  0.1000E+05
     0.0000E+00 -0.1000E+01  0.0000E+00
  0.1000E+04
     0.0000E+00  0.1000E+01 -0.1000E+01


-------------------------------------------
EQ  2
P-
  0.1000E+01
     0.0000E+00  0.0000E+00  0.1000E+01
Q-
  0.1000E+01
     0.0000E+00  0.1000E+01  0.0000E+00
  0.1000E+01
     0.1000E+01  0.0000E+00  0.0000E+00
        DEGREE OF DIFICULTY=  0


-------------------------------------------------------------
        DELTAS DETERMINED AS
     0.1667E+00  0.1667E+00  0.6667E+00  0.6667E+00


-------------------------------------------------------------
        TRIAL SOLUTION FOUND                CIV          0.1637E+04
          U             D            PGV 1


-------------------------------------------
     0.2728E+01  0.3666E+02  0.3360E+02


-------------------------------------------------------------
        DELTAS DETERMINED AS
     0.6083E-01  0.6083E-01  0.8783E+00  0.8783E+00


-------------------------------------------------------------
        TRIAL SOLUTION FOUND                CIV          0.1263E+04
          U             D            PGV 1


-------------------------------------------
     0.7683E+00  0.1302E+03  0.1173E+03
```

```
-------------------------------------------------------
        DELTAS DETERMINED AS
     0.5800E-02  0.5800E-02  0.9884E+00  0.9884E+00


-------------------------------------------------------
        TRIAL SOLUTION FOUND                CIV          0.1036E+04
          U             D             PGV 1


-------------------------------------------------------
     0.6010E-01  0.1664E+04  0.1624E+04


-------------------------------------------------------
        DELTAS DETERMINED AS
     0.3612E-04  0.3612E-04  0.9999E+00  0.9999E+00


-------------------------------------------------------
        TRIAL SOLUTION FOUND                CIV          0.1000E+04
          U             D             PGV 1


-------------------------------------------------------
     0.3613E-03  0.2768E+06  0.2767E+06


-------------------------------------------------------
        DELTAS DETERMINED AS
     0.1306E-08  0.0000E+00  0.1000E+01  0.1000E+01
        NOTE DELTAS LESS/EQ ZERO
        JOB ABORTED
```

LITERATURE CITED

Avriel, M., and Williams, A. C., 1970, Complementary geometric programming: SIAM J. Appl. Math., v. 19, p. 125-141.

_____1971, An extension of geometric programming with applications in engineering optimization: J. Eng. Math., v. 5, no. 3, p. 187-194.

Duffin, R. S., Peterson, E. L., and Zener, C. M., 1967, Geometric programming: New York, John Wiley and Sons, 278 p.

Jelen, F. C., ed., 1970, Cost and optimization engineering: New York, McGraw-Hill Book Co., 490 p.

Kabak, I. W., 1969, Some aspects of optimal design: AIIE Trans., v. 1, no. 4, p. 371-374.

Korzybski, A., 1946, Science and sanity: Lakeville, Conn., The International Non-Aristotelian Library Publishing Co., 806 p.