GEORGIA INSTITUTE OF TECHNOLOGY
OFFICE OF CONTRACT ADMINISTRATION
SPONSORED PROJECT INITIATION

Date: May 4, 1978

Project Title: Development of a Methodology for the Random Number Generation of Integer Programming Test Problems

Project No: E-24-666

Project Director: Dr. Ronald L. Rardin

Sponsor: U.S. Department of Commerce, National Bureau of Standards

Agreement Period: From 6/15/78 Until 11/1/78

Type Agreement: Delivery Order No. 805938, dated 2/13/78

Amount: $9,984 USDC/NBS
1,190 GIT (E-24-329)
$11,174 Total

Reports Required: Final Report

Sponsor Contact Person (s):

Technical Matters
(Contracting Officer's Technical Representative -- COTR)
Mr. R.H.F. Jackson
U.S. Dept. of Commerce
National Bureau of Standards
Room A428, Building 101
Washington, D.C. 20234
(301) 921-3855

Contractual Matters
(thru OCA)
Ms. Peggy L. Moore
Contracting Ordering Officer
U.S. Department of Commerce
National Bureau of Standards
Procurement Section
Washington, D.C. 20234
(301) 921-3103

Defense Priority Rating: n/a

Assigned to: ISyE (School/Laboratory)

COPIES TO:

- Project Director
- Division Chief (EES)
- School/Laboratory Director
- Dean/Director—EES
- Accounting Office
- Procurement Office
- Security Coordinator (OCA)
- Reports Coordinator (OCA)
- Library, Technical Reports Section
- EES Information Office
- EES Reports & Procedures
- Project File (OCA)
- Project Code (GTRI)
- Other

CA—3 (3/76)

GEORGIA INSTITUTE OF TECHNOLOGY
OFFICE OF CONTRACT ADMINISTRATION

SPONSORED PROJECT TERMINATION

Date: 9/15/80

Project Title: DEVELOPMENT OF A METHODOLOGY FOR THE RANDOM NUMBER GENERATION OF INTEGER PROGRAMMING TEST PROBLEMS

Project No: E-24-666

Project Director: RARDIN/ISYE

Sponsor: US DEPARTMENT OF COMMERCE, NATIONAL BUREAU OF STANDARDS

Effective Termination Date: 12/1/78

Clearance of Accounting Charges: 12/1/78

Grant/Contract Closeout Actions Remaining:

NONE

_ Final Invoice and Closing Documents
_ Final Fiscal Report
_ Final Report of Inventions
_ Govt. Property Inventory & Related Certificate
_ Classified Material Certificate
_ Other _____

Assigned to: INDUSTRIAL AND SYSTEMS ENGINEERING (School/Laboratory)

COPIES TO:

Project Director
Division Chief (EES)
School/Laboratory Director
Dean/Director–EES
Accounting Office
Procurement Office
Security Coordinator (OCA)
Reports Coordinator (OCA)
Library, Technical Reports Section
EES Information Office
Project File (OCA)
Project Code (GTRI)
Other OCA RESEARCH PROPERTY COORDINATOR

CA–4 (1/79)

THE RIP RANDOM INTEGER PROGRAMMING TEST PROBLEM GENERATOR

by

Ronald L. Rardin
Georgia Institute of Technology

and

Benjamin W. Lin
Rutgers University

June, 1980

with the support of

National Bureau of Standards
Applied Mathematics Division
Contract Number 805938

1. Introduction

Many important optimization problems in management and engineering take the form of integer linear programming problems. Thus a great deal of recent research has been directed toward the development of algorithms that produce either exact or heuristic optimal solutions to such problems.

In most cases the effectiveness of these algorithms can only be measured empirically, i.e., by comparing the performance of the algorithms on a specified set of test problems. Naturally, a large, diverse and widely available collection of test problems is required. Unfortunately, it is well recognized that the present supply of suitable test problems is quite limited; sound empirical research in integer programming has been correspondingly restricted.

The effectiveness of any optimization procedure can ultimately be measured only against problems taken from the actual setting where the algorithm is proposed for implementation. However, data for such "real world" test problems are often strictly proprietary, or at least poorly documented and difficult to obtain. They may also reflect atypical characteristics of a specific application site. Thus, such test problems do not seem to offer a practical source of the great volumes of flexible test data needed by algorithm developers.

For these reasons integer programming researchers, like empirical researchers in many other scientific fields, have often turned to "laboratory" abstractions of real problem data for a more adequate supply of test cases. The approach used is to randomly generate test problems from a population described by parametric problem properties (number of rows, number of columns, coefficient density, coefficient size, etc.). Such problems are synthesized by a generating code from a sequence of pseudo-random numbers.

Although they provide only models of true problems, and are thus subject to all the validation concerns inherent in any model, randomly-generated problems offer a variety of conveniences. Their number is virtually unlimited because each change in the seed of the pseudo-random number generator will yield a new problem. The problems are also compactly reported and communicated because only the code and the parameters are necessary to reproduce a test. Since problem characteristics are under the parametric control of the researcher, algorithm performance can be studied over a wide range of problem conditions. Perhaps most important, randomly generated problems provide the only available approach to measuring the effectiveness of complex heuristics designed for large-scale integer problems. When problems are too large to be solved exactly, one cannot know the degree of optimality of a heuristically-obtained solution unless the solved problem was generated so that its exact optimal solution is known a priori.

In this context the authors and others ([3], [4], [5], [6], [7], [8]) have proposed over the last few years a series of concepts and techniques that may be employed to generate random integer programming test problems with known parametric properties. This report presents the most recent and comprehensive product of that research. Under sponsorship of the National Bureau of Standards Division of Applied Mathematics, the authors have developed and preliminarily tested a generating code RIP (Random Integer Program) which controls, to at least some degree, a large number of problem properties usually considered by integer programming researchers to be importantly connected with algorithm performance.

All problems generated by RIP are general linear integer or linear mixed-integer programs that are feasible and have finite optimal solutions. Coefficients in the constraints are integers, but costs are real. At the completion of problem generation, both an optimal solution to the full problem and the optimal solution to its linear programming relaxation are available to

the user along with the problem data. Table 1 outlines the parameters under at least partial user control in RIP. Details on inputs and outputs of the RIP code are presented in Section 8.

TABLE 1

SUMMARY OF PROBLEM CHARACTERISTICS UNDER AT LEAST PARTIAL USER CONTROL WITH RIP

Problem Size and Form

- Number and Mix of Continuous, 0-1 and General Integer Variables
- Number and Mix of Equality, Greater Than and Less Than Constraints

Constraint Matrix Coefficients

- Density of Nonzeroes
- Nonnegative or Mixed-Sign
- Magnitude of Nonzeroes

Linear Relaxation Solution Properties

- Mix of Fractional and Integer-Valued Basic Integer Variables
- Mix of Tight and Slack Inequalities
- Determinant of the Optimal Basis
- Mix of Degenerate and Nondegenerate Basic Values

Integer Solution Properties

- Mix of Tight and Slack Inequalities at Integer Optimality
- Percent Difference Between the Linear Relaxation and Integer Solution Values
- Similarity Between the Linear Relaxation and Integer Solution Vectors

2. A Random Cut Concept

The most theoretically challenging aspect of generating random integer programs with known optimal solutions arises in guaranteeing a specified solution is optimal without explicitly solving the problem. Practical, necessary and sufficient conditions for optimality that would provide such a guarantee are not available for most integer problems. However, reasonable sufficient conditions can be formulated. Any problem construction which creates a solution satisfying such a sufficient solution can produce problems with known optima.

In [4 , 5 , 6] Fleisher and Meyer developed one such set of sufficient conditions for integer optimality as a generating tool. The approach taken in RIP is quite similar, and identical in some special cases. However, it will be developed here from a very general random cutting plane point of view that is quite different from that of Fleisher and Meyer.

To see the basic strategy of a random cut generator, assume that slack variables have been added to any inequality rows so that the generated problem is to have the form[1]

$$\min \quad cx \tag{2-1}$$

$$\text{s.t.} \quad Ax = b \tag{2-2}$$

$$d \geq x \geq 0 \tag{2-3}$$

$$x_j \text{ integer for } j \in \Psi \tag{2-4}$$

---

[1] Here and throughout this report upper case Latin letters represent matrices and lower case Latin letters represent vectors. The symbol 0 represents either a scalar, a vector or a matrix of zeroes as consistent with context. When particular components of a vector are discussed, subscripts are added to the vector name, e.g., x_j is the j^{th} component of x.

Here $d_j = \infty$ on any slack variables and $d_j = 1$ for 0-1 variables. A system of cutting planes

$$Hx \geq h \tag{2-5}$$

is said to be valid if every solution to (2-2) - (2-4) also satisfies (2-5). Furthermore, it is clear that any x^* that solves (2-1) through (2-3) plus (2-5) and also happens to satisfy (2-4) also solves (2-1) through (2-4).

The random cut strategy combines this observation with the fact that (2-1) through (2-3) plus (2-5) form a linear program, and convenient optimality conditions are available. If an integer problem and some implied valid cuts are generated so that a known solution solves the linear relaxation including the cuts, then the solution solves the integer problem as well. More specifically the strategy proceeds as follows:

(i) Generate constraints of an integer linear program in the form (2-2), (2-3) and (2-4).

(ii) Select a solution x^* that satisfies all constraints.

(iii) Randomly select a number of cutting planes (2-5) from a large family of valid ones for the constraints (2-2) through (2-4).

(iv) Construct costs c so that the specified x^* satisfies Kuhn-Tucker optimality conditions for the linear program (2-1) through (2-3) plus (2-5), i.e., so that c is an appropriate randomly weighted sum of the binding constraints at x^*.

(v) Discard the random cuts and return the problems (2-1) through (2-4) to a test problem user.

Clearly the difficult part of implementing this approach is finding a rich family of cutting planes for step (iii). In fact, cut generation is further complicated by the need to construct optimality conditions in (iv). Any cut

that is satisfied as a strict inequality by the solution x^* will have no role in the Kuhn-Tucker equations expressing c as a weighted sum of binding constraints. Thus it is not only necessary to find a family of valid cutting planes from which a random sample can easily be drawn, but all chosen cuts must be binding (satisfied as equalities) at x^*.

The family of valid cutting planes used in RIP is characterized by Lemma 1 below. In that lemma and all discussion later in this section, it will be assumed that A, x, c and d have been partitioned so that

$$A = [B,M,N]$$

where B is a specified basis matrix

$$x = [x_B, x_M, x_N]$$

$$c = [c_B, c_M, c_N]$$

$$d = [d_B, d_M, d_N]$$

M and N are characterized by the vector $\bar{x}$, the basic solution corresponding to B. Specifically,

$$\bar{x}_B = B^{-1}b - B^{-1}M\bar{x}_M - B^{-1}N\bar{x}_N \qquad (2\text{-}6)$$

where all components of $\bar{x}_M$ equal 0 and all components of $\bar{x}_N$ equal the corresponding components of d_N.

Lemma 1

Consider the constraints

$$x_B = B^{-1}b - B^{-1}Mx_M - B^{-1}Nx_N \qquad (2\text{-}7)$$

$$x_M \geq 0 \qquad (2\text{-}8)$$

$$x_N \leq d_N \tag{2-9}$$

$$x_j \text{ integer for } j \in \Psi \ . \tag{2-10}$$

Let $\bar{x}_M = 0$, $\bar{x}_N = d_N$, $\bar{x}_B$ be as in (2-6), and ℓ be an integer vector of the same dimension as x_B with nonzero entries only on components j belonging to Ψ. Also let

$$\sigma_M = \ell B^{-1} M \tag{2-11}$$

$$\sigma_N = \ell B^{-1} N \tag{2-12}$$

and ϕ = the fractional part of $\ell \bar{x}_B$. Then if $\phi > 0$ the following is a valid cut for (2-7) through (2-10):

$$\frac{1-\phi}{\phi} \sum_{\sigma_{Mj}>0} \sigma_{Mj}(x_{Mj}-\bar{x}_{Mj}) + \frac{1-\phi}{\phi} \sum_{\sigma_{Nj}<0} \sigma_{Nj}(x_{Nj}-\bar{x}_{Nj})$$

$$+ \sum_{\sigma_{Mj}<0} (-\sigma_{Mj})(x_{Mj}-\bar{x}_{Mj}) + \sum_{\sigma_{Nj}>0} (-\sigma_{Nj})(x_{Nj}-\bar{x}_{Nj}) \geq 1 - \phi \tag{2-13}$$

Proof. The definition of ℓ assures ℓx_B must be integer for any x_B satisfying (2-10). Since $\phi > 0$, either $\ell x_B \leq \ell \bar{x}_B - \phi$ or $\ell x_B \geq \ell \bar{x}_B + 1 - \phi$, i.e., either

$$-\ell(x_B - \bar{x}_B) \geq \phi \tag{2-14}$$

or

$$\ell(x_B - \bar{x}_B) \geq 1 - \phi \tag{2-15}$$

Substituting for $\bar{x}_B$ and x_B via (2-6) and (2-7), either

$$-\ell(B^{-1}b - B^{-1}Mx_M - B^{-1}Nx_N - B^{-1}b + B^{-1}M\bar{x}_M + B^{-1}N\bar{x}_N) \geq \phi$$

or

$$-\ell(-B^{-1}b + B^{-1}Mx_M + B^{-1}Nx_N + B^{-1}b - B^{-1}M\bar{x}_M - B^{-1}N\bar{x}_N) \geq 1 - \phi$$

Rearranging, and using the definitions of σ_M and σ_N, either

$$\frac{1-\phi}{\phi}(\sigma_M)(x_M - \bar{x}_M) + \frac{1-\phi}{\phi}(\sigma_N)(x_N - \bar{x}_N) \geq 1 - \phi \qquad (2\text{-}16)$$

or

$$(-\sigma_M)(x_M - \bar{x}_M) + (-\sigma_N)(x_N - \bar{x}_N) \geq 1 - \phi \qquad (2\text{-}17)$$

Now by definition of $\bar{x}_M$ and $\bar{x}_N$, together with (2-8) and (2-9),

$$(\bar{x}_M - x_M) = (0 - x_M) \leq 0$$

$$(\bar{x}_N - x_N) = (d_N - x_N) \geq 0$$

Thus (2-16) and (2-17) can further be simplified to

$$\frac{1-\phi}{\phi} \sum_{\sigma_{Mj}>0} (\sigma_{Mj})(x_{Mj} - \bar{x}_{Mj}) + \frac{1-\phi}{\phi} \sum_{\sigma_{Nj}<0} (\sigma_{Nj})(x_{Nj} - \bar{x}_{Nj}) \geq 1 - \phi \qquad (2\text{-}18)$$

or

$$\sum_{\sigma_{Mj}<0} (-\sigma_{Mj})(x_{Mj} - \bar{x}_{Mj}) + \sum_{\sigma_{Nj}>0} (-\sigma_{Nj})(x_{Nj} - \bar{x}_{Nj}) \geq 1 - \phi \qquad (2\text{-}19)$$

Recognizing that if one or the other of the totals in (2-18) and (2-19) exceeds $1-\phi$, then so must their sum yields the cut (2-13).

■

The derivation so far has developed a very broad family of cutting planes that can easily be generated by taking random linear combinations integer-

restricted rows of a constraint matrix expressed in terms of a given basis. It remains to show how cuts of this family can be made tight at integer optimality. The following lemma justifies the approach taken in RIP.

Lemma 2

Let x^* be a solution to (2-7) through (2-10), and let $\bar{x}$, σ_M, σ_N, ℓ and ϕ be as in the hypothesis of Lemma 1. Then if

(i) $\bar{x}_{Mj} = x^*_{Mj}$ whenever $\sigma_{Mj} > 0$

(ii) $\bar{x}_{Nj} = x^*_{Nj}$ whenever $\sigma_{Nj} < 0$, and

(iii) $0 < (x^*_B - \bar{x}_B) < 1$,

the cut (2-13) is satisfied as an equality by x^*.

Proof. From (iii), $\ell(x^*_B - \bar{x}_B)$ is a fraction. Now the fact that x^*_B satisfies (2-10) and that ℓ has nonzero coefficients only on integer components of x^*_B assures ℓx^*_B is an integer. Thus $\ell(x^*_B - \bar{x}_B) = \ell x^*_B - \ell\bar{x}_B$ = fractional part $(-\ell\bar{x}_B)$ = 1 - fractional part $(\ell\bar{x}_B)$ = $1 - \phi$. On the other hand, substitution to x^*_B and $\bar{x}_B$ via (2-7) gives

$$\ell(x^*_B - \bar{x}_B) = \ell(B^{-1}b - B^{-1}Mx^*_M - B^{-1}Nx^*_N - B^{-1}b + B^{-1}M\bar{x}_M + B^{-1}N\bar{x}_N)$$

$$= -\sigma_M(x^*_M - \bar{x}_M) - \sigma_N(x^*_N - \bar{x}_N)$$

Using (i) and (ii), we have

$$1-\phi = \ell(x^*_B - \bar{x}_B) = \sum_{\sigma_{Mj}<0} (-\sigma_{Mj})(x^*_M - \bar{x}_M) + \sum_{\sigma_{Nj}>0} (-\sigma_{Nj})(x^*_N - \bar{x}_N) \qquad (2\text{-}20)$$

Observation that (i) and (ii) render the first two sums of (2-13) equal to zero yields the lemma.

3. General Strategy

RIP employs the theory in Lemmas 1 and 2, together with a series of well-known algebraic properties, to generate random integer programs in a sequence of four phases. The problem is generated in the form

$$\min \quad c_B x_B + c_P x_P + c_Q x_Q + c_S x_S + c_T x_T$$

$$\text{s.t.} \quad Bx_B + Px_P + Qx_Q + Sx_S + Tx_T = b$$

$$0 \leq x_B \leq d_B \quad 0 \leq x_P \leq d_P \quad 0 \leq x_Q \leq d_Q$$

$$0 \leq x_S \leq d_S \quad 0 \leq x_T \leq d_T$$

$$x_j \text{ integer for } j \in \Psi .$$

Here B is the optimal basis matrix of the linear relaxation, P and Q collect nonbasic variables in the linear relaxation that change value in the integer solution, and S and T are nonbasic portions that do not change in the integer solution. P and S are distinguished from Q and T in that variables in P and S are nonbasic lower-bounded; those in Q and T are nonbasic upper-bounded. Thus, the M and N of Lemmas 1 and 2 satisfy M = [P,S], N = [Q,T]. At the completion of generation, columns are randomly [1/] rearranged, so that algorithms cannot easily exploit this (B,P,Q,S,T) partitioning. As with Lemmas 1 and 2, $\bar{x} = (\bar{x}_B, \bar{x}_P, \bar{x}_Q, \bar{x}_S, \bar{x}_T)$ represents an optimal linear relaxation solution, $x^* = (x^*_B, x^*_P, x^*_Q, x^*_S, x^*_T)$ denotes an optimal integer solution, $c = (c_B, c_P, c_Q, c_S, c_T)$ is the cost vector and $d = (d_B, d_P, d_Q, d_S, d_T)$ is the vector of the

---

[1/] Here and in all further discussion the term "randomly" should be understood to mean any outcome in the required range is equally likely.

upper bounds. The problem may have any specified mix of equality rows and inequality rows, but we shall imagine adding slack variables to x for all inequalities.

Phase 1 of generation concentrates on the optimal linear relaxation basis matrix, B. Taking advantage of the well-known fact that any integer matrix can be reduced by elementary row and column operations to a diagonal matrix, B is constructed as

$$B = R^{-1} D\, C^{-1} \tag{3-1}$$

where R^{-1} and C^{-1} are integer, triangular, unimodular matrices, and D is a diagonal matrix. The factorization of (3-1) will be seen to facilitate control of numerous properties of B and the corresponding basic solution components $\bar{x}_B$ and x^*_B.

Simultaneously with the creation of B in Phase 1, a matrix L of cut multipliers is generated. Each row of L produces one linear combination, vector ℓ in Lemmas 1 and 2. Basic solution vectors $\bar{x}_B$ and x^*_B are created to satisfy, property (iii) of Lemma 2, i.e.,

$$0 < L(x^*_B - \bar{x}_B) < 1$$

as well as numerous other user-specified parameters.

Phase 2 of RIP centers on P and Q, the nonbasic sectors where $\bar{x}$ may differ from x^*. Sign conventions are enforced so that all cut coefficients, $LB^{-1}(P,Q)$, have the signs required by properties (i) and (ii) of Lemma 2. At the same time coefficients of P and Q, and solution segments $\bar{x}_P$, $\bar{x}_Q$, x^*_P and x^*_Q are chosen so that

$$B(\bar{x}_B - x^*_B) + P(\bar{x}_P - x^*_P) + Q(\bar{x}_Q - x^*_Q) = 0 \tag{3-2}$$

i.e., so that the problem will be feasible with $\bar{x}_S = x^*_S$ and $\bar{x}_T = x^*_T$.

RIP's Phase 3 turns to costs, c. Random, feasible and complementary dual multipliers, u for the main constraints and v for cuts, are generated for both the linear relaxation and the integer solution. The implied cost, c, is then computed by the Kuhn-Tucker equation as the specified sum of binding constraint and cut rows. Finally, a reduced-gradient-like procedure is employed to make the cost yield as nearly as possible the desired ratio of the optimal linear and integer solution values.

Problem generation is completed in Phase 4. The relatively free S and T segments of the problem are generated to produce the desired A matrix coefficient density while retaining Kuhn-Tucker optimality of both the linear and integer solution. The right-hand-side vector, b, is then computed via direct substitution of one generated solution vector into the constraints.

4. Phase 1: Basic Aspects and Cuts

The first major phase of RIP generates the optimal linear relaxation basis matrix B, corresponding solution and bound vectors $\bar{x}_B$, x_B^* and d_B, and the cut-creating linear combination matrix L. To assure B will be the basis produced by any linear programming code, $\bar{x}$ is constructed as the unique linear relaxation optimum. The dimension of B is, m, the user-specified number of constraint rows. As outlined above, B is generated in factored form (3-1), where R^{-1} and C^{-1} are integer, unimodular and triangular matrices, and D is a matrix with diagonal entries δ_i satisfying

$$\text{Determinant } (B) = \prod_{i=1}^{m} \delta_i \tag{4-1}$$

$$\delta_i \text{ integer for } i = 1,2,\ldots,m \tag{4-2}$$

$$\delta_i \text{ divides } \delta_{i+1} \text{ for } i = 1,2,\ldots,m-1 \tag{4-3}$$

RIP users may specify the δ_i directly or merely input the desired determinant. In the latter case, RIP computes a δ factorization that minimizes the number of $\delta_i = 1$. In either event the δ_i are available at the beginning of Phase 1 and establish the determinant of B.

Let $\bar{b}$ be the corrected right-hand-side when nonbasic upper-bounded variables in the linear relaxation solution have been considered, i.e.,

$$\bar{b} = b - B^{-1}P\bar{x}_P - B^{-1}Q\bar{x}_Q - B^{-1}S\bar{x}_S - B^{-1}T\bar{x}_T \tag{4-4}$$

If $\bar{x}_B$ is to be feasible in the linear relaxation, we must have

$$B\bar{x}_B = R^{-1}DC^{-1}\bar{x}_B = \bar{b} \qquad (4\text{-}5)$$

At the same time $\bar{x}_B$ must satisfy a series of specific conditions on its makeup (no. fractional components, etc.).

For ease in later generation of P, Q, S and T, RIP constructs an all-integer $\bar{b}$. The approach taken is to first construct $\bar{x}_B$ with all the desired properties and then choose C^{-1} so that $C^{-1}\bar{x}_B$ is of the form $(\gamma_1/\delta_1, \gamma_2/\delta_2, \ldots, \gamma_m/\delta_m)$ with all γ_i integer. Clearly if $C^{-1}\bar{x}_B$ has such a form, $\bar{b} = Bx_B = R^{-1}DC^{-1}\bar{x}_B$ will be integer.

To generate $\bar{x}_B$ we shall introduce a categorization of the rows.[1] Rows 1 through m_{slk} will have an inequality slack as the basic variable at linear relaxation optimality. Rows $(m_{slk}+1)$ through m_{int} will have integer-restricted ($j\varepsilon\Psi$) decision variables basic at integer values. Rows $(m_{int}+1)$ through m_{cont} will have continuous (not integer-restricted) variables basic. Finally, rows $m_{cont}+1$ through m will have integer-restricted variables basic at fractional values. If the first three groups are further subdivided as to whether the row's basic variable is degenerate ($\bar{x}_{Bj} = 0$ or d_{Bj}) or nondegenerate ($0 < \bar{x}_{Bj} < d_{Bj}$), it is clear that any mix of $\bar{x}_B$ characteristics mandated by the "Linear Relaxation Solution Properties" in Table 1 can be easily constructed. Moreover, control of the number of nondegenerate rows among the first m_{slk} is exactly control of the fraction of tight constraints at linear optimality. Of course row categories for any grouping not present in the desired problem form (e.g., continuous variables in an all-integer problem) is merely omitted in the above.

RIP generation of $\bar{x}_B$ begins with creation of d_B. The first m_{slk} components are $+\infty$, integer variables which are to be 0-1 have $d_{Bi} = 1$, and all other d_{Bi} are drawn from a uniform distribution over the

---

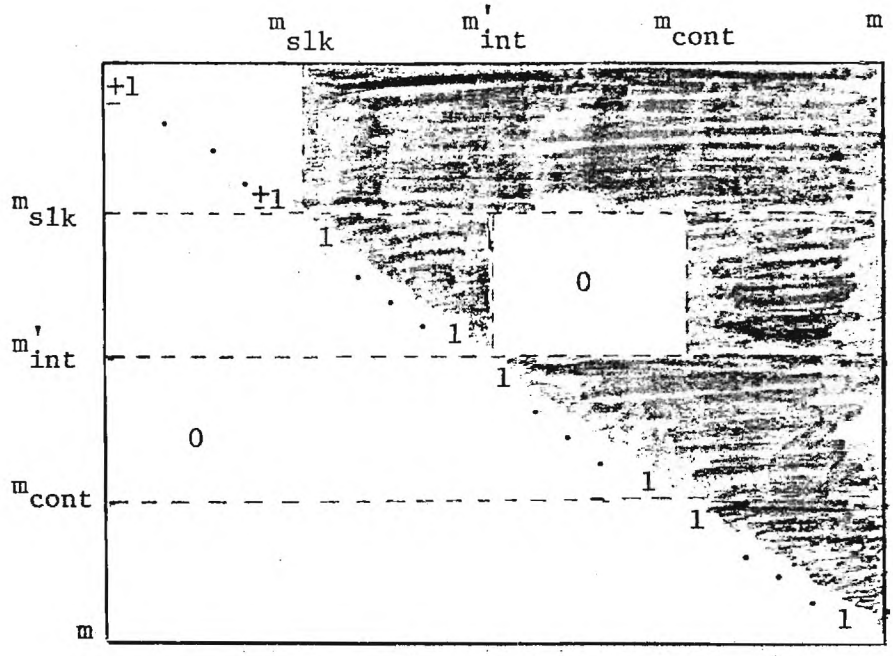
1/ As with the column categorization, row sequence will be randomized at the completion of problem generation, so that row categorization cannot be easily exploited by an algorithm.

allowable range. Each $\bar{x}_{Bi}$ is then constructed as η_i/τ_i with integers η_i and τ_i satisfying

- Either $\eta_i = 0$ or $\eta_i = d_{Bi}\tau_i$ for degenerate rows i (4-6)
- $0 \leq \eta_i/\tau_i \leq d_{Bi}$ for i = 1,2,...,m (4-7)
- τ_i divides τ_{i+1} for i = 1,2,...,m-1 (4-8)
- τ_m divides δ_m of D (4-9)
- τ_i divides η_i for i = 1,2,...,i_f-1 (4-10)
- τ_i and η_i are relatively prime for i = i_f,i_{f+1},...,m (4-11)

Here i_f indexes the starting point of the fractional portion of $\bar{x}_B$. Clearly (4-6) and (4-7) assume $\bar{x}_{Bi} = \eta_i/\tau_i$ is within its bounds and degenerate where required. Also, (4-10) and (4-11) make $\bar{x}_{Bi}$ integer for $i < i_f$ and fractional thereafter. Properties (4-8) and (4-9) will be exploited below in the generation of C^{-1}. Our assumed row categorization, together with the requirement that $\bar{b}$ be integer, leave any $i\varepsilon[m_{int}+1, m_{cont}+1]$ an admissible choice for i_f. RIP selects i_f randomly within that interval. Denominators τ_i satisfying (4-8) and (4-9) are obtained by randomly "sliding left" the δ_i. The τ_i with $1 \leq i \leq i(1)$ all equal the first non-unit δ_k, say $\delta_{k(1)}$; those in the interval i(1)+1,...,i(2) equal $\delta_{k(1)+1}$; etc.

The next generation step is creation of C^{-1}. Although entries may be added or deleted to affect problem density, the general form of RIP's C^{-1} matrix is



(4-12)

The matrix is upper triangular. All rows have ±1 diagonal entries, so that C^{-1} is unimodular as required. Each row also has at least one integer entry, α_i, in the shaded area to the right of the diagonal.

Observe that (4-12) is further subdivided according to the row categories already introduced. Here, and in many later generation steps, different row and column groups will receive different treatments. In (4-12) columns $1,2,\ldots,m_{slk}$ are positive or negative unit vectors so that corresponding columns of B will be appropriate for slack variables. A block of zeros is maintained above the diagonal in (4-12) to facilitate cut computations at (4-17) below.

When row/column groups are treated differently, a greater variety of problems can be obtained if all or nearly all possible combinations of treatments actually occur. The "effective integer area boundary," m'_{int}, of (4-12) is derived with such considerations in mind. If user input precludes enough rows in the interval $[m_{int}+1, m_{cont}]$ some of those in $[m_{slk}+1, m_{int}]$ are

transferred to create an "effective continuous region", $[m'_{int}+1, m_{cont}]$ with $m_{slk}+1 \leq m'_{int} \leq m_{int}$.

Recall that we wish to have

$$C^{-1}\bar{x}_B = (\gamma_1/\delta_1, \gamma_2/\delta_2, \ldots, \gamma_m/\delta_m) \tag{4-13}$$

for integers $\gamma_1, \gamma_2, \ldots, \gamma_m$. Property (4-9) makes this automatic for row m. To accomplish it for other rows, we must choose some C^{-1} coefficients rather carefully. Specifically, when there is exactly one α_i in the shaded part of each (4-12) row, we must have

$$\frac{\gamma_i}{\delta_i} = \frac{\eta_i}{\tau_i} + \alpha_i \frac{\eta_k}{\tau_k}, \tag{4-14}$$

where $k > i$ is the column of C^{-1} containing α_i. Rearranging (4-14), yields the diophantive equation form

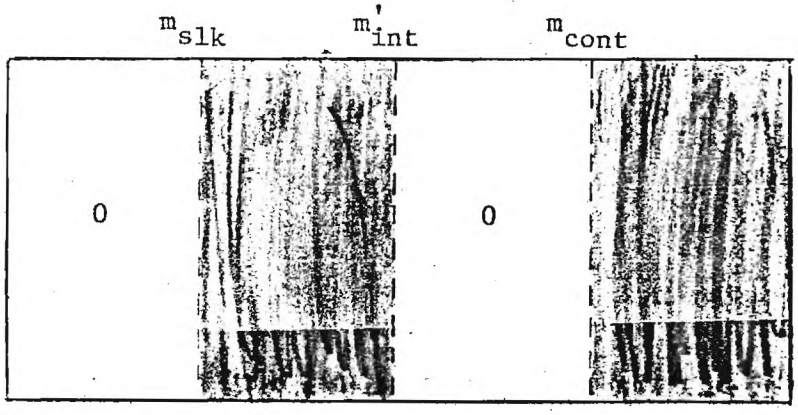
$$(\tau_k)\gamma_i + (-\eta_k\delta_i)\alpha_i = (\eta_i\delta_i\tau_k/\tau_i) \tag{4-15}$$

with unknown integer variables γ_i and α_i. The right-hand-side is integer because $k > i$ and (4-8). A solution can easily be obtained by the Euclidean algorithm (see Blankinship [2]) if $\gcd(\tau_k, -\eta_k\delta_i)$, the greatest common divisor of (τ_k) and $(-\eta_k\delta_i)$, divides $(\eta_i\delta_i\tau_k/\tau_i)$. To see that this will always be the true recall properties (4-8) through (4-11). If $k \geq i_f$, (4-11) assures $\gcd(\tau_k, -\eta_k\delta_i)$ is the same as $\gcd(\tau_k, -\delta_i)$, and so divides $(\eta_i\tau_k/\tau_i)$ times δ_i. If $k < i_f$, (4-10) makes $\tau_k = \gcd(\tau_k, -\eta_k\delta_i)$. But since $i < k$ must also be less than i_f, τ_i divides η_i. Thus the right-hand-side for this case is the required integer multiple of τ_k.

The next problem component to be generated is x_B^*, the basic segment of the solution that is to be integer optimal. Non-slack components of x_B^* are generated by shifting $\bar{x}_B$ values up or down to integer-feasible values. Where

options are available in choosing x^*_{Bj}, a special RIP distance distribution selects $|x^*_B - \bar{x}_B|$ relatively large or small as indicated by a user parameter.[1/] Slack components of x^*_B are chosen to satisfy the desired degree of integer solution constraint slackness. The linear relaxation basis contains a user-specified number of positive-valued slacks and a separate number of zero-valued, degenerate slacks. A third user parameter selects how many of these slack variables are to be positive in the integer optimal solution, x^*_B. Thus by controlling the three parameters, the user can make the $\bar{x}$ solution more or less tight than the x^* one and either can be very tight or very slack.

To see how to complete the basic sector of our problem, we first look ahead to the generation of RIP's random cuts. The cuts are created by an integer linear combination matrix, L, having the form



(4-16)

Zeroes in columns for the slack ($i \leq i \leq m_{slk}$) and continuous ($m'_{int} < i \leq m_{cont}$) sectors assure that rows of L place weight only on integer-constrained variables as required for Lemma 1. The number of rows of L is the user-specified number of random cuts.

[1/] See Table 3 below for details of the distribution.

The cut-controlling matrix, L, is constructed in the form

$$L = \tilde{F}C^{-1} \tag{4-17}$$

where $\tilde{F}$ is an intermediate integer matrix of the form (4-16). The carefully placed zero block of C^{-1} (see (4-12)) assures the (4-16) structure of $\tilde{F}$ is transmitted to $L = \tilde{F}C^{-1}$.

To conform to condition (iii) of Lemma 2, an L generated via (4-17) must satisfy

$$0 < L(x_B^* - \bar{x}_B) = \tilde{F}[C^{-1}(x_B^* - \bar{x}_B)] < 1 \tag{4-18}$$

At the same time, we shall wish to control the sign of LB^{-1} to achieve conditions (i) and (ii) of Lemma 2. Define $F = LB^{-1}$. The scheme employed in RIP requires

$$F\Omega = LB^{-1}\Omega \geq 0 \tag{4-19}$$

where Ω is a diagonal, orientation matrix. Diagonal entries ω_i of Ω are 0 on slack ($1 \leq i \leq m_{slk}$) and continuous rows ($m'_{int} < i \leq m_{cont}$) rows; ω_i equals +1 or -1 elsewhere, with greater than or equal to inequalities having +1, less than or equal to inequalities having -1, an equalities having a random choice.

When x_B^* and C^{-1} are completed, $C^{-1}(x_B^* - \bar{x}_B)$ is easily computed. $\tilde{F}$ is then randomly constructed--one row at a time--to satisfy (4-18) and take a first step toward (4-19) by keeping

$$\tilde{F}\Omega \geq 0 \tag{4-20}$$

Components $\tilde{f}_{ki}$ conforming to (4-16) and (4-20) are generated in random i-order--each a random portion of the remaining slack in the inequality $0 < \tilde{f}_k C^{-1}(x_B^* - \bar{x}_B) < 1$.

Now by (4-17) and (3-1)

$$F = LB^{-1} = (\tilde{F}C^{-1})(R^{-1}DC^{-1})^{-1} = \tilde{F}C^{-1}CD^{-1}R = \tilde{F}D^{-1}R \tag{4-21}$$

$\tilde{F}$ and D^{-1} are fixed, so our one remaining opportunity to satisfy (4-19) is through picking R to obtain

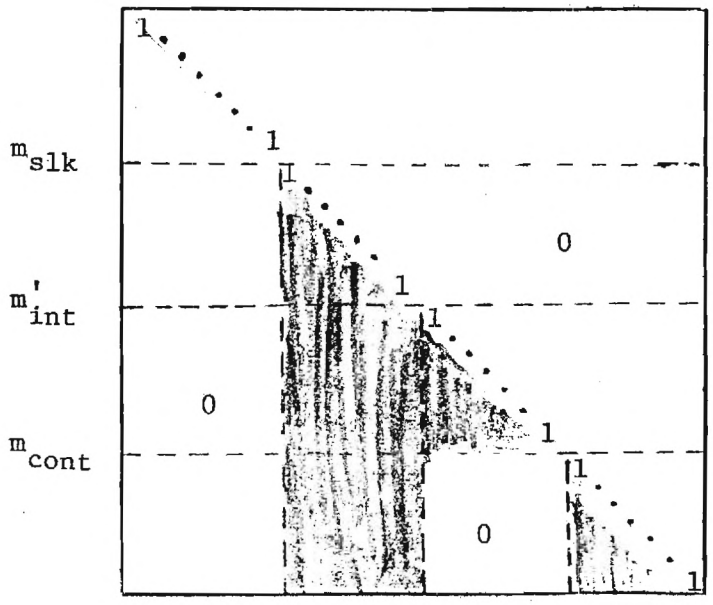
$$\tilde{F}DR\Omega \geq 0 \tag{4-22}$$

On the other hand $B = R^{-1}DC^{-1}$ must still lead to a difference $B(x_B^* - \bar{x}_B)$ that can be matched in the P, Q, S and T portions generated later. The specific requirement is

$$0 \leq \Omega B(x_B^* - \bar{x}_B) = \Omega R^{-1}DC^{-1}(x_B^* - \bar{x}_B) \tag{4-23}$$

The only unfixed component is R^{-1}, which must at the same time be integer, lower triangular and unimodular.

RIP's strategy for meeting these specifications on R and its inverse is essentially one of careful bookkeeping. The R form generated is



(4-24)

Diagonal elements are all 1 to assure unimodularity of R^{-1}. The first m_{slk} vectors of both matrices are unit vectors, so that $R = R^{-1}DC^{-1}$ has unit vectors for basic slack variables. The zero (unshaded) block at the bottom of columns $m'_{int}+1$ through m_{cont} assures that $\tilde{F}D$, which has the structure of (4-16), times the R of (4-24) will yield an LB^{-1} of the form (4-16). Shaded elements of R and R^{-1} are generated randomly, one column at a time. Running totals are maintained on the conditions (4-22) and (4-23) to assure they are simultaneously satisfied in the results.

This approach is greatly facilitated if R and R^{-1} can be thought of as being generated simultaneously. To achieve this simultaneity, one additional technique is employed. If at any time an entry of R's column i is placed in row k (>i by triangularity) then when the time comes to generate column k, it is left a unit vector. The effect of this scheme is to make off-diagonal entries in R^{-1} the negatives of those in R.

5. Phase 2: P and Q

Phase 2 of RIP generates the P and Q sectors of the problem, i.e., variables that are nonbasic in the linear relaxation solution, but may change value in the integer solution. The P variables are nonbasic lower-bounded at linear optimality and the Q variables are nonbasic upper-bounded.

As with Phase 1, the process begins with the generation of appropriate upper bounds, d_P and d_Q. RIP permits no slack vectors in P and slack vectors cannot be nonbasic upper-bounded (e.g., in Q). Thus all d_{Pj} and d_{Qj} are finite, with those for 0-1 variables fixed at 1 and others randomly chosen within the user-controlled range.

The second Phase 2 step is selection of the solution vectors $\bar{x}_P$, $\bar{x}_Q$, x_P^* and x_Q^*. By definition

$$\bar{x}_P = 0 \tag{5-1}$$

$$\bar{x}_Q = d_Q. \tag{5-2}$$

Values of x_P^* and x_Q^* are random integers satisfying

$$d_P \geq x_P^* \geq 0 \tag{5-3}$$

$$d_Q \geq x_Q^* \geq 0 \tag{5-4}$$

and chosen to conform to RIP's user-controlled distribution of $|\bar{x}_j - x_j^*|$.[1/]

Generation of the coefficient matrices P and Q is the most complex aspect of Phase 2. These coefficients must simultaneously assure conditions (i) and (ii) of Lemma 2, and feasibility of the $\bar{x}$ and x^* solutions. The feasibility requirement is (3-2) because $\bar{x}_S = x_S^*$, $\bar{x}_T = x_T^*$ implies no deviation

---

[1/] See Table 3 for details.

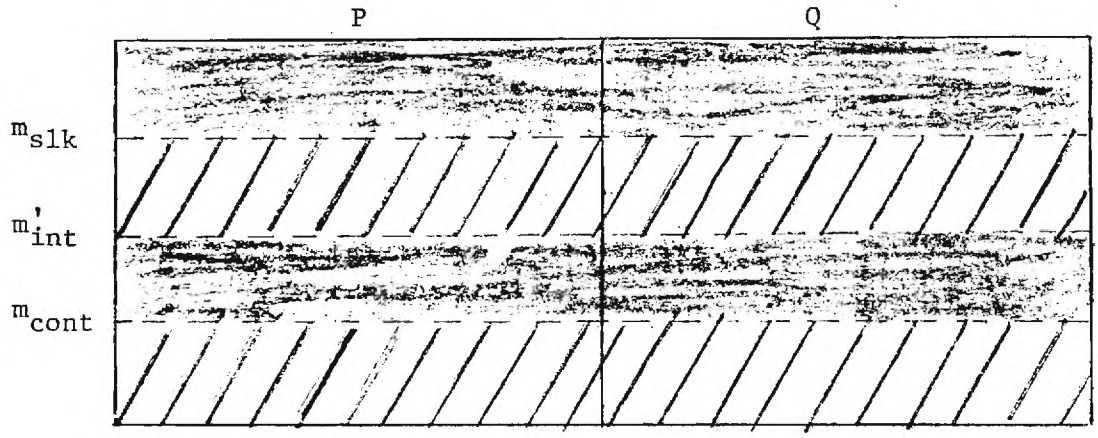
between linear and integer solution row sums can be resolved after P and Q are set. In our full matrix form, conditions (i) and (ii) of Lemma 2 mandate

$$LB^{-1}P = FP \leq 0 \tag{5-5}$$

and

$$LB^{-1}Q = FQ \geq 0. \tag{5-6}$$

To meet these requirements we generate P and Q in the format



(5-7)

Coefficients of the slack and continuous rows regions (solid shading) are unrestricted in sign. Coefficients p_{ij} in the integer (cross-hatched) areas of P must satisfy the sign convention

$$\omega_i p_{ij} \leq 0 \tag{5-8}$$

where the ω_i are as defined in Section 4. Integer-region coefficients q_{ij} of Q conform to

$$\omega_i q_{ij} \geq 0 \tag{5-9}$$

Of course, if the user specifies the A matrix to be nonnegative, no negative coefficients are created in any part of P and Q.

Recall that F has the form (4-16) and that its columns are oriented so

that $F\Omega \geq 0$. This implies slack and continuous rows will receive zero weight when F premultiplies either P or Q. Sign orientation of integer rows in (5-8) and (5-9) yields (5-5) and (5-6).

The remaining issue is feasibility, i.e., (3-2). Phase 2 constructs feasibility by generating rows of (P,Q) one at a time. Coefficients are chosen in random column order to conform to (5-8) and (5-9), each being formed as a random portion of the remaining slack in (3-2). It is important to note that it might be impossible to achieve feasibility with (5-8) and (5-9). Together with (5-1) through (5-4) those limits show

$$\Omega[P(\bar{x}_P - x_P^*) + Q(\bar{x}_Q - x_Q^*)]$$

$$= \Omega[P(0 - x_P^*) + Q(d_Q - x_Q^*] \geq 0 \qquad (5\text{-}10)$$

However, from (4-23) we know $\Omega B(\bar{x}_B - x_B^*) \leq 0$. It follows that suitable p_{ij} and q_{ij} can be found to make each row of (3-2) sum to 0.

Although it may have little practical significance, it would be at least esthetically displeasing if continuous variables always took on integer values in RIP solutions. We have already seen that choice if i_f in (4-10) and (4-11) allows a random number of continuous $\bar{x}_{Bi}$ to be fractional. A final adjustment to continuous x_{Pj}^* and x_{Qj}^* assures some of them are fractional in the RIP integer solution. The above generation process is first completed with all x_{Pj}^* and x_{Qj}^* integer. Then, for continuous components, a denominator ν_j is generated as

$$\nu_j = d_j/\text{gcd}(d_j, x_j^*) \qquad (5\text{-}11)$$

A change of variables replaces x_j by x_j/ν_j. Correspondingly revision of problem data as

$$P_j = \nu_j P_j \quad \text{or} \quad Q_j = \nu_j Q_j \tag{5-12}$$

$$c_j = \nu_j c_j \tag{5-13}$$

$$d_j = d_j/\nu_j \tag{5-14}$$

$$\bar{x}_j = \bar{x}_j/\nu_j \tag{5-15}$$

$$x_j^* = x_j^*/\nu_j \tag{5-16}$$

creates some fractional x_j^* without impacting other problem properties.

6. Phase 3: Costs

Suppose for the moment that S and T are vacuous. Then, in addition to the already-assured basic structure and primal feasibility, Kuhn-Tucker optimality conditions for the linear relaxation problem are

$$c_B = \bar{u}B \tag{6-1}$$

$$c_P > \bar{u}P \tag{6-2}$$

$$c_Q < \bar{u}Q \tag{6-3}$$

$$\bar{u}_i \geq 0 \text{ for } i\varepsilon\Gamma \tag{6-4}$$

$$\bar{u}_i \leq 0 \text{ for } i\varepsilon\Lambda \tag{6-5}$$

$$\bar{u}_i = 0 \text{ for rows i slack at } \bar{x} \tag{6-6}$$

where $\bar{u}$ is an optimal dual solution, $\Gamma = \{i:$ row i is a greater than or equal to inequality$\}$ and $\Lambda = \{i:$ row i is a less than or equal to inequality$\}$. Conditions (6-1) through (6-5) assure dual feasibility, and (6-6) plus the equality in (6-1) provides complementary slackness. The strict inequality of (6-2) and (6-3) is valid because $\bar{x}_P = 0$, $\bar{x}_Q = d_Q$ and necessary because the $\bar{x}$ solution is to be unique.

In terms of the P, Q, S, T partition, cut (2-13) have the form

$$\sum_{f_k P_j > 0} \frac{1-\phi_k}{\phi_k} f_k P_j (x_{Pj} - \bar{x}_{Pj}) + \sum_{f_k Q_j < 0} \frac{1-\phi_k}{\phi_k} f_k Q_j (x_{Qj} - \bar{x}_{Qj})$$

$$+ \sum_{f_k P_j < 0} (-f_k P_j)(x_{Pj} - \bar{x}_{Pj}) + \sum_{f_k Q_j > 0} (-f_k Q_j)(x_{Qj} - \bar{x}_{Qj}) \tag{6-7}$$

$$+ \sum_{f_k S_j > 0} \frac{1-\phi_k}{\phi_k} f_k S_j (x_{Sj} - \bar{x}_{Sj}) + \sum_{f_k T_j < 0} \frac{1-\phi_k}{\phi_k} f_k T_j (x_{Tj} - \bar{x}_{Tj})$$

$$+ \sum_{f_k S_j < 0} (-f_k S_j)(x_{Sj} - \bar{x}_{Sj}) + \sum_{f_k T_j > 0} (-f_k T_j)(x_{Tj} - \bar{x}_{Tj}) \geq 1-\phi_k$$

where ϕ_k is the fractional part of $\ell_k \bar{x}_B$, ℓ_k is the kth row of L, and $f_k = \ell_k B^{-1}$, the kth row of F. However, the sign conventions (4-19), (5-8) and (5-9) assure the first two sums are vacuous. Moreover, primal feasibility gives the equality

$$B(x_B - \bar{x}_B) + P(x_P - \bar{x}_P) + Q(x_Q - \bar{x}_Q) + S(x_S - \bar{x}_S) + T(x_T - \bar{x}_T) = 0 \quad (6\text{-}8)$$

Adding f_k times (6-8) to (6-7) and simplifying, gives the restated cut

$$\ell_k(x_B - \bar{x}_B) + \sum_{f_k S_j > 0} \frac{1}{\phi_k} f_k S_j (x_{Sj} - \bar{x}_{Sj}) + \sum_{f_k T_j < 0} \frac{1}{\phi_k} f_k T_j (x_{Tj} - \bar{x}_{Tj}) \geq 1-\phi_k \quad (6\text{-}9)$$

It is in this (6-9) form that RIP considers cut rows in constructing integer optimality. Note that the coefficient row for x_B is ℓ_k; x_P and x_Q have zero coefficients. Accordingly, integer optimality conditions corresponding to those of (6-1) to (6-6) are

$$c_B = u^*B + v^*L \quad (6\text{-}10)$$

$$c_P = u^*P \quad (6\text{-}11)$$

$$c_Q = u^*Q \quad (6\text{-}12)$$

$$v^* \geq 0 \quad (6\text{-}13)$$

$$u_i^* \geq 0 \quad \text{for } i \in \Gamma \quad (6\text{-}14)$$

$$u_i^* \leq 0 \quad \text{for } i \in \Lambda \quad (6\text{-}15)$$

$$u_i^* = 0 \quad \text{for rows } i \text{ slack at } x^* \tag{6-16}$$

Here u^* is an integer-optimal dual multiplier for the main constraints and v^* is the corresponding dual multiplier for the cuts.

Of course, full optimality conditions must also include the sectors corresponding to S and T. However, the requirements imposed by those relatively "free" matrices are easily satisfied. Thus RIP proceeds to pick dual multipliers and costs in Phase 3 considering only (6-1) through (6-6) and (6-10) through (6-16). Corrections will be added for S and T in Phase 4.

The approach taken in Phase 3 is to first construct $\bar{u}$, u^*, v^*, c_B, c_P and c_Q that provably satisfy the above conditions, and then to search for choices that produce a ratio of the optimal integer and linear relaxation solution values closely approximating the user-specified value. The first constructed solution derives from randomly generated nonnegative vectors v^* and r^*, together with their sum

$$\bar{r} = r^* + v^* \tag{6-17}$$

The $\bar{u}$ and u^* are then created as

$$\bar{u} = \bar{r}F \tag{6-18}$$

$$u^* = r^*F \tag{6-19}$$

Initial c_B, c_P and c_Q follow directly from $\bar{u}$, u^* and v^* via (6-10), (6-11) and (6-12).

Recall that F has zero columns in the slack area and nonzero columns oriented positively for $i \varepsilon \Gamma$ and negatively for $i \varepsilon \Lambda$. These facts, together with nonnegativity of $\bar{r}$ and r^* demonstrate our construction satisfies (6-4) through (6-6) and (6-13) through (6-16). Properties (6-17) through (6-19), together with (5-5) and the (6-11) cost computation yield

$$\bar{u}P = \bar{r}FP = r^*FP + v^*FP \leq r^*FP = u^*P = c_P \qquad (6\text{-}20)$$

Similarly the dual construction plus (5-6) and (6-12) gives

$$\bar{u}Q = \bar{r}FQ = r^*FQ + v^*FQ \geq r^*FQ = u^*Q = c_Q \qquad (6\text{-}21)$$

Finally, (6-1), (6-17), (6-18), (6-19) and $F = LB^{-1}$ lead to

$$c_B = \bar{u}B = \bar{r}FB = r^*FB + v^*FB = u^*B + v^*L \qquad (6\text{-}22)$$

Relations (6-20), (6-21) and (6-22) essentially prove (6-2), (6-3), and (6-10), respectively. The only issue is whether inequality in (6-20) and (6-21) is strict. Such will be the case if v^*FP and $-v^*FQ$ are strictly positive vectors. RIP addresses this requirement by rejecting columns P_j and Q_j in Phase 2 that do not satisfy $FP_j > 0$ and $FQ_j < 0$. That rejection scheme together with a choice of v^* strictly positive Phase 3 assures that this strict inequality carries over to (6-20) and (6-21).

Before proceeding to the cost improvement procedure, we consider one modification. The (4-16) structure of F, together with (6-18) and (6-19) produces not only the desired (6-6) and (6-16), but also

$$\bar{u}_i = u_i^* = 0 \quad \text{for } i = m'_{int}, \ldots, m_{cont}.$$

However, it is easy to see that none of the proofs (6-20) through (6-22) are impacted by making such $\bar{u}_i$ and u_i^* nonzero, so long as they remain equal. The actual RIP construction assigns such nonzero values to the continuous region u_i before computing c's. Of course, the values are chosen to conform to (6-4) and (6-5) or (6-14) and (6-15).

For a user-specified fraction, μ, of the gap between the optimal linear relaxation solution value that of the integer optimum, we would like to have

$$\mu \gtrsim \frac{c_B(x_B^* - \bar{x}_B) + c_P(x_P^* - \bar{x}_P) + c_Q(x_Q^* - \bar{x}_Q)}{|c_B x_B^* + c_P x_P^* + c_Q x_Q^*|} \tag{6-23}$$

Substituting for c_B, c_P and c_Q via (6-10), (6-11) and (6-12), and collecting terms gives

$$\mu \gtrsim \frac{u^*[B(x_B^* - \bar{x}_B) + P(x_P^* - \bar{x}_P) + Q(x_Q^* - \bar{x}_Q)] + v^*[L(x_B^* - \bar{x}_B)]}{\pm u^*[Bx_B^* + Px_P^* + Qx_Q^*] \pm v^*[Lx_B^*]} \tag{6-24}$$

where the plus denominator applies when $cx^* \geq 0$ and minus is chosen otherwise. Primal feasibility of (3-2) implies the u^* term of the (6-24) numerator is always zero. Under our Phase 3 convention that S and T are vacuous, the u^* term of the denominator is $\pm u^*b$. Thus the objective function gap ratio can be rewritten as

$$\theta(u^*,v^*) = \pm \frac{v^* L(x_B^* - \bar{x}_B)}{u^*b + v^*L(x_B^* - \bar{x}_B)} \tag{6-25}$$

with $\pm$ as in (6-24).

Phase 3's cost ratio improvement procedure interatively modifies u* and v* to bring $\theta(u^*,v^*)$ closer to the user-specified μ. Of course, care must be taken to assure that the optimality conditions constructed above are maintained. Specifically, the implied

$$\bar{u} = c_B B^{-1} = u^* + v^* F \tag{6-26}$$

must continue to satisfy (6-1) through (6-6); u* and v* must fulfill (6-13) through (6-16).

At each iteration, the direction of change in (u^*,v^*) is derived directly from the gradient of $\theta(u^*,v^*)$. If $\theta(u^*,v^*)$ is too small, the direction parallels

the gradient; if $\theta(u^*,v^*)$ is too large, the negative gradient is used. Each step proceeds until a satisfactory $\theta(u^*v^*)$ is reached or until further move-movement would violate one of the constraints on u^* and v^*. Thus, the scheme can be viewed as a reduced gradient algorithm (see for example Bazaraa and Shetty [1]) truncated when an objective function threshold is reached, or when a basis change would be required.

7. Phase 4: S, T and b

The only remaining elements of our problem to be fixed are the right-hand-side vector b and components connected with the nonbasic submatrices S and T. Phase 4 sets these problem segments.

As with other phases, the process begins with the generation of appropriate upper bounds d_S and d_T. S may contain slack vectors that must have $d_{Sj} = \infty$. Components of either x_S or x_T that are to be 0-1 variables must have d_{Sj} or d_{Tj} equal 1. All other upper bounds are picked randomly over the allowed range.

Once d_S and d_T are set, the definition of S and T mandates

$$\bar{x}_S = x_S^* = 0 \tag{7-1}$$

$$\bar{x}_T = x_T^* = d_T. \tag{7-2}$$

The next step is generation of the S and T matrices themselves. Since (7-1) and (7-2) hold, S and T figure neither in primal feasibility nor in cut tightness. Thus they are essentially open to completely random generation. RIP uses this opportunity to establish the user's desired density of nonzero entries in the full constraint matrix, A. The total number of entries so far generated in B, P and Q is counted, and the desired remainder placed randomly through S and T. The only restrictions enforced are that each column must have at least one entry and that S and T are nonnegative if the user desires.

Completion of S and T makes possible direct calculation of b via

$$b = B\bar{x}_B + P\bar{x}_P + Q\bar{x}_Q + S\bar{x}_S + T\bar{x}_T \tag{7-3}$$

or equivalently

$$= Bx_B^* + Px_P^* + Qx_Q^* + Sx_S^* + Tx_T^*$$

The last needed problem elements are costs c_S and c_T. In terms of the dual multipliers u^* and v^* generated in Phase 3, integer optimality requires

$$c_{Sj} \geq u^* S_j + \sum_{f_k S_j > 0} \frac{v_k^* f_k}{\phi_k} S_j \quad \text{for all } j \tag{7-4}$$

$$c_{Tj} \leq u^* T_j + \sum_{f_k T_j < 0} \frac{v^* f_k}{\phi_k} T_j \quad \text{for all } j \tag{7-5}$$

where f_k and ϕ_k are as in (6-9). Noting (6-13), (6-26) and $0 < \phi_k < 1$,

$$\bar{u}S_j = u^* S_j + \sum_k (v_k^* f_k) S_j \leq u^* S_j + \sum_{f_k S_j > 0} \frac{v_k^* f_k}{\phi_k} S_j \tag{7-6}$$

and

$$\bar{u}T_j = u^* T_j + \sum_k (v_k^* f_k) T_j \geq u^* T_j + \sum_{f_k T_j < 0} \frac{v_k^* f_k}{\phi_k} T_j \tag{7-7}$$

We can conclude that the linear relaxation optimality conditions $c_{Sj} > \bar{u}S_j$ and $c_{Tj} < \bar{u}T_j$ will hold if (7-4) and (7-5) are satisfied as strict inequalities. Phase 4 computes c_S and c_T in exactly this way.

The term $c_S x_S = c_S(0)$ has no effect on the cost ratio (6-23). However, $c_T x_T$ does add the constant $c_T \bar{x}_T = c_T x_T^* = c_T d_T$ to the denominator. Thus, some final adjustment is necessary to be sure the cost ratio balance achieved in Phase 3 is not destroyed in Phase 4.

As a function of a denominator constant, ε, the ratio (6-23) is

$$\theta(\varepsilon) = \frac{(cx^* - c\bar{x})}{|cx^* + \varepsilon|} \tag{7-8}$$

Figure 1 plots a typical $\theta(\varepsilon)$. Observe that a given $\mu > 0$ can be achieved at

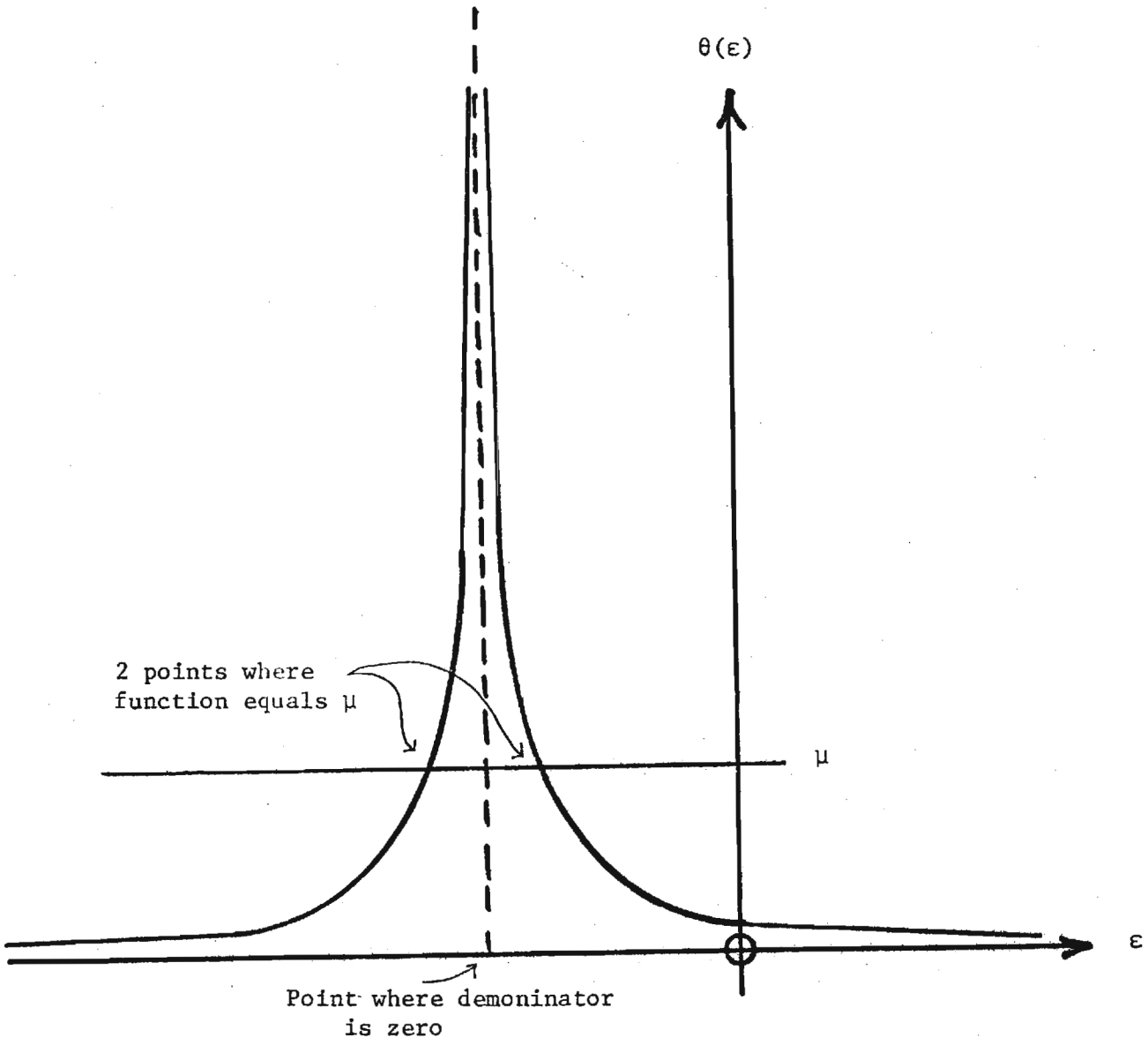


FIGURE 1. TYPICAL FORM OF SOLUTION
GAP RATIO AS A FUNCTION OF
DENOMINATOR CONSTANT ε

either of two ε's. One makes the denominator $cx^* + \varepsilon < 0$, and the other has $cx^* + \varepsilon > 0$. These cases can be loosely interpreted as making the generated problem a maximization or a minimization respectively.

The last step of Phase 4 is to increase or decrease the c_{Tj} and d_{Tj} to achieve one of the two available ε's. If both can be achieved while (7-5) remains a strict inequality, the nearest is selected. Otherwise, the RIP chooses the one ε that can most nearly be attained.

On some individual problems it may not be possible to reach either ε without violating (7-5). RIP partially anticipates this difficulty by empirically biasing T_j coefficients and the Phase 3 ratio goal toward Phase 4 flexibility. Although initial testing showed these adjustment techniques are sufficient, only approximate fulfillment of the user-specified ratio can be guaranteed.

8. Using the RIP Generator

The RIP generator is implemented as a series of (relatively) machine independent FORTRAN subroutines, all having names beginning with the three letters RIP. Users need only prepare a proper call to subroutine RIP. That routine supervises all subsequent generation activities and finishes by returning the problem to the user's program.

The problem that is returned should be thought of in the form

$$\min \quad cx$$

$$s\,t \quad A\begin{pmatrix} x \\ x_s \end{pmatrix} = b$$

$$d \geq x \geq 0$$

$$x_s \geq 0$$

$$x_j \text{ integer for } j \in \Psi$$

Here x_s is a vector of slack and surplus variables of inequality constraints and all other notation is as above. The call to RIP that yields such a problem has the format

```
CALL RIP (AAA, ACOL, RHS, CCC, DDD, ROWTYP,
* VBLTYP, XBAR, XSTAR, DENBAR, DENSTR, SOLBAR, SOLSTR,
* PARAM)
```

CCC, SOLBAR, SOLSTR and RHS are real,[1/] but all other parameters are integer. Details of their definition and meaning are given in Table 2. The following definitions apply throughout

n = the dimension of x, c and d

s = the dimension of x_s

m = the number of rows in A, i.e. the dimension of b

$\bar{x}$ = an optimal vector x for the linear programming relaxation of the full problem

---

[1/] DOUBLE PRECISION for UNIVAC or IBM, REAL for CDC.

TABLE 2. ELEMENTS OF THE RIP CALL

Item	Description	Required User Dimension	Notes
AAA	The list of nonzero components in the A matrix, packed by columns. where for single precision [1/] $t \approx 20m+5n+3n(m+\text{no. cuts})(\text{A matrix density})$ and for double precision [1/] $t \approx 40m+5n+3n(m+\text{no. cuts})(\text{A matrix density})$	(2, max{1000,t})	1. AAA(1,k) = The row number in which a nonzero entry falls 2. AAA(2,k) = The nonzero entry 3. Unusually large dimensions are required because AAA is also used by RIP as temporary work area 4. The A matrix density (times 1000) is specified in PARAM.
ACOL	The vector of right AAA subscripts corresponding to the last entry for a particular column	(n+s)	1. Nonzero entries for column 1 are in AAA(1,k) and AAA(2,k) for k=1,2,...,ACOL(1). 2. Nonzero entries for column j > 1 are in AAA(1,k) and AAA(2,k) for k=ACOL(j-1) +1,..., ACOL(j)
RHS	The right-hand-side vector b	(m)	1. RHS is real [1/].
CCC	The cost vector c	(n+s)	1. CCC is real [1/].
DDD	The upper bound vector d	(n+s)	1. DDD(j)= will equal 1 for a 0-1 x_j
ROWTYP	A vector of codes showing the form of each row	(m)	1. ROWTYP(i) = -1 if row i is $\geq$ 0 if row i is = +1 if row i is $\leq$
VBLTYP	A vector of codes showing the type of each variable	(n+s)	1. VBLTYP(j) = 1 if j refers to a surplus variabl for a $\geq$ constraint = 2 if j refers to a slack variable for a $\leq$ constraint = 3 if j refers to a continuous variable = 4 if j refers to a general integer variable = 5 if j refers to a 0-1 integer iab

TABLE 2 (CONT'D)

Item	Description	Required User Dimension	Notes
XBAR	The vector of numerators of the (unique) solution linear programming relaxation of the problem, i.e. $\bar{x}$	(n+s)	1. $\bar{x}$ can be obtained from XBAR by dividing all entries by DENBAR
XSTAR	The vector of numerators of the (possibly not unique) solution to the full (mixed) integer problem, i.e. x^*	(n+s)	1. x^*can be obtained from XSTAR by dividing all entries by DENSTR
DENBAR	The denominator of $\bar{x}$ values above	none	
DENSTR	The denominator of x^* values above	none	
SOLBAR	The value of the linear programming solution, $c\bar{x}$	none	1. SOLBAR is real[1/]
SOLSTR	The value of the integer programming solution, cx^*	none	1. SOLSTR is real[1/]
PARAM	A vector of input parameters passed to RIP by the user to describe characteristics of the problem to be generated	(m+65)	1. See Tables 3 and 4 for definitions and restrictions on parameters

1/ DOUBLE PRECISION is used on IBM and UNIVAC, SINGLE PRECISION (i.e., REAL) on CDC computers.

$\bar{x}_s$ = an optimal vector x_s for the linear programming relaxation of the full problem

x^* = an optimal x for the full integer problem

x^*_s = an optimal x_s for the full integer problem

Users describe desired characteristics of the output problem through the parameter vector PARAM. Table 3 shows the meaning, allowable range, and default value (if any) for each parameter. RIP automatically checks that PARAM values fall within the allowable range and then replaces zero parameter values by defaults where defaults are provided. RIP also checks a number of internal consistency properties which should be satisfied by the parameters. Table 4 details those properties.

If either range tests or internal consistency tests of parameters fail, RIP prints brief error messages and proceeds to a STOP 2. The user should correct the indicated errors and call RIP again.

RIP is also equipped with an internal verification routine that checks Kuhn-Tucker optimality conditions on generated problems. The verification feature is activated by making PARAM (41) = 1. Normally this would only be done by users installing the program for the first time. When errors are detected, verification routines print brief error messages including the words "VERIF FAILURE". Although a program bug may be evident these messages usually indicate a need to adjust internal RIP tolerances rather than a true failure of optimality. The generated problem may still be useful for user experimentation.

TABLE 3. RIP INPUT PARAMETERS

Parameter Number	Description	Minimum Value	Maximum Value	Default Value
1-10	A 40-character problem name (10 A4)	none	none	none
11	Sign limitations on A matrix coefficients (0= mixed sign 1= nonnegative)	0	1	0
12	Ten times the percent non-zero coefficients in the non-slack part of A.	50 (i.e 5.0%)	900 (i.e.90.0%)	none
13	Maximum absolute value of random nonzero coefficients in A	0	1000	50
14	Not used	0	0	none
15	m = the number of constraints	6	500	none
16	n = the number of decision variables	10	1000	none
17	Number of equality constraints	0	500	none
18	Number of $\geq$ inequality constraints	0	500	none
19	Number of $\leq$ inequality constraints	0	500	none
20	Number of decision variables basic in the $\bar{x}$ solution	6	500	none
21	Number of slack/surplus variables basic and 0 (i.e. degenerate) in the $\bar{x}_s$ solution	0	494	none
22	Number of slack variables basic and positive (i.e. nondegenerate) in the $\bar{x}_s$ solution	0	494	none

TABLE 3. (CONT'D)

Parameter Number	Description	Minimum Value	Maximum Value	Default Value
23	Number of tight rows (equalities and tight inequalities) in the (x^*,x_s^*) solution	3	500	none
24	Number of slack rows (inequalities with positive slack) in the (x^*,x_s^*) solution	0	500	none
25	Number of continuous (not integer constrained) decision variables	0	900	none
26	Number of general integer (integer constrained, but not necessarily 0-1) decision variables	0	1000	none
27	Number of 0-1 decision variables	0	1000	none
28	Number of continuous variables in the $\bar{x}$ basis at degenerate values (i.e. = 0 or = d_j)	0	494	none
29	Number of continuous variables in the $\bar{x}$ basis at nondegenerate values (i.e. < d_j and >0)	0	494	none
30	Number of general integer and/or 0-1 variables in the $\bar{x}$ basis at degenerate (and thus integer) values (i.e. = 0 or = d_j)	0	494	none
31	Number of general integer variables in the $\bar{x}$ basis at nondegenerate (i.e. interior) integer values (i.e. > 0 and < d_j and thus not 0-1)	0	494	none
32	Number of general integer and/or 0-1 variables in the $\bar{x}$ basis at fractional (and thus nondegenerate) values > 0	3	500	none
33	Number of nonbasic and lower bounded decision variables in the $\bar{x}$ solution that change value in the x^* solution	1	500	none

TABLE 3. (CONT'D)

Parameter Number	Description	Minimum Value	Maximum Value	Default Value
34	Number of nonbasic and upper bounded decision variables in the $\bar{x}$ solution that change value in the x^* solution	1	500	none
35	Number of nonbasic and lower bounded decision variables in the $\bar{x}$ solution that keep the same value in the x^* solution	1	500	none
36	Number of nonbasic and upper bounded decision variable in the $\bar{x}$ solution that keep the same value in the x^* solution	1	500	none
37-40	Not used	0	0	none
41	Verify switch (1=do check generated problems for verifiable optimality, 0=do not)	0	1	0
42	Maximum value of generated upper bounds d	0	100	10
43	Not used	0	0	none
44	Ten times the desired percent, μ, that the linear relaxation to integer solution value gap forms of the absolute integer solution value	10 (i.e. 1%)	500 (i.e. 50%)	100 (i.e. 10%)
45	Number of random cutting planes to be constructed by RIP	3	500	none
46	Ten times the percent by which the probability that $\lvert \bar{x}_j - x^*_j \rvert = \alpha$ changes as α changes to $\alpha+1$ (e.g. 1000 implies distance is uniformly distributed, 900 implies near distances are more probable and 1100 imples far distances are more probable)	0	3000 (i.e 300%)	1000 (i.e. 100%)
47	Not used	0	0	none
48-49	The seed pair for the main RIP pseudo-random number generator ((49) must be positive and odd)	0	65535	none

TABLE 3. (CONT'D)

Parameter Number	Description	Minimum Value	Maximum Value	Default Value
50-51	The seed pair for the RIP pseudo-random number generator used to generate the basic portion of the $(\bar{x},\bar{x}_s)$ solution ((51) must be odd if not defaulted)	0	65535	generated from (48) and (49)
52-53	The seed pair for the RIP pseudo-random number generator used to generate the LP basic portion of the (x^*,x^*_s) solution ((53) must be odd if not defaulted)	0	65535	generated from (48) and (49)
54-55	The seed pair for the RIP pseudo-random number generator used to generate the LP-basic portion of the A matrix and the random cutting planes ((55) must be odd if not defaulted)	0	65535	generated from (48) and (49)
56-57	the seed pair for the RIP pseudo-random number generator used to generate the LP-nonbasic portions of $(\bar{x},\bar{x}_s)$, A and (x^*,x^*_s) that change between $(\bar{x},\bar{x}_s)$ and (x^*,x^*_s) ((57) must be odd if not defaulted)	0	65535	generated from (48) and (49)
58-59	The seed pair for the RIP pseudo-random number generator used to generate costs c. ((59) must be odd if not defaulted)	0	65535	generated from (48) and (49)
60-61	The seed pair for the RIP pseudo-random number generator used to generate the LP- nonbasic portions of $(\bar{x},\bar{x}_s)$, A and (x^*,x^*_s) that do not change between $(\bar{x},\bar{x}_s)$ and (x^*,x^*_s) ((61) must be odd if not defaulted)	0	65535	generated from 48) and (49)
62	The logical unit number of the user's FORTRAN printer	0	99	6
63	The desired level of RIP printout (0 = none, 1 = summary of problem parameters and properties, 2 = as 1 plus the $\bar{x}$ and x^* vectors, 3= as 2 plus c, d, A and b.	0	3	0

TABLE 3. (CONT'D)

Parameter Number	Description	Minimum Value	Maximum Value	Default Value
64	Right dimension of the AAA matrix in the RIP call	1000	100000	none
65	The determinant of the optimal basis in the linear programming relaxation (if not defaulted must be a product of positive integers each $\leq$ 541 with at least one to a nonunit power)	0	1000000	product of (66) through (65+m)
66-(65+m)	The elements of the Smith diagonal partitioning of the determinant of the optimal basis in the linear programming relaxation (if not defaulted each is positive and divides the next)	0	none	generated from (65)

TABLE 4. RIP Parameter Properties Checked

Reference Number	Property (Numbers in parentheses are parameter numbers).
901	(15) = (17) + (18) + (19)
902	(15) = (20) + (21) + (22)
903	(15) = (23) + (24)
904	(16) = (28) + (29) + (30) + (31) +(32) +(33) + (34) + (35) + (36)
905	(20) = (28) + (29) + (30) + (31) + (32)
906	Some parameter (66) through (65+m) does not divide its successor
907	Both (65) and $\prod_{i=1}^{m}$ (65+i) are 0
908	(24) $\leq$ (18) + (19)
909	(21) + (22) $\leq$ (18) + (19)
910	(24) $\leq$ (21) + (22)
911	3 $\leq$ (28) + (29) + (30) + (31)
912	(16) = (25) + (26) + (27)
913	(65) is nonzero and not the product of integers each of which is $\leq$ 541
914	(31) $\leq$ (26)
915	(30) + (31) + (32) $\leq$ (26) + (27)
916	(28) + (29) $\leq$ (25)
917	Right member of a specified pseudo-random number seed pair is not odd
918	Either the specified (64+m) is 1 or (65) is not a product of integers with at least one to the second power.
919	The first ((21) + (22)) of nondefaulted (66) through (65+m) are not unity.
920	(32) $\leq$ (45)
921	Inadequate AAA dimension, (64).

As indicated under PARAM (63) in Table 3, RIP users have the option to select any of several levels of automatic problem printout. A PARAM (63) value of 0 produces no RIP print. A value of 1 produces the log of problem characteristics and input parameter values illustrated in Figure 2.

A user can quickly verify whether he has reproduced the same problem as another researcher by comparing the "DATA CHECK TOTALS" of the Figure 2 printout. Those values are modulo-reduced sums of problem data that should exactly match if the same parameters are supplied to RIP. The only allowable exceptions are small variations in the "CCC" check total caused by differences among computers in the accuracy of real arithmetic.

Setting PARAM (63) = 2 yields this problem summary plus the optimal solution vectors $\bar{x}$ and x^* in the format shown in Figure 3. When users choose PARAM (63) = 3, the full generated problem is printed out. A parameter summary like the one in Figure 2 is followed by a column-by-column statement of the generated problem. Figure 4 shows an abridged example. The meaning of entries for each column is as follows:

COL = the column number

XBAR = the linear relaxation solution component $\bar{x}_j$

XSTAR = the generated integer optimal solution component x^*_j

C = the cost c_j for the column

D = the upper bound d_j for the column

TYPE = the type of this variable (3 = a continuous variable, i.e., $j \notin \Psi$; 4 = a general variable, i.e., $j \in \Psi$; 5 = a 0-1 variable, i.e., $j \in \Psi$ and $d_j = 1$)

ρ IN ROW i = a nonzero coefficient ρ is in row i of the generated column (zero entries are not reported).

The level 3 print illustrated in Figure 4 concludes with a row-by-row listing of the right-hand-side vector, b_i, and a listing of row types.

$$\text{ROW TYPE} = \begin{cases} -1 \text{ for greater than or equal to rows } i \varepsilon \Gamma \\ +1 \text{ for less than or equal to rows } i \varepsilon \Lambda \\ 0 \text{ for equality rows } i \end{cases}$$

```
**************************
*  RIP PROBLEM SUMMARY   *
**************************

PROBLEM NAME=1ST TEST
NUMBER VARIABLES=    25
NUMBER CONSTRAINTS=    12
NUMBER INTEGER VARIABLES=    20
LINEAR RELAXATION SOLUTION VALUE=   .50542077E+04
INTEGER SOLUTION VALUE=   .56157864E+04
DENSITY OF NONZERO COEFFICIENTS DESIRED=      300
ACTUAL DENSITY OF NONZERO COEFFICIENTS=      300
SOLUTION GAP PERCENT OF IP SOLUTION DESIRED=      100
ACTUAL SOLUTION GAP PERCENT OF IP SOLUTION=      100


INPUT PARAMETER LIST--
 / 11=          0/ 12=        300/ 13=          0/ 14=          0
 / 15=         12/ 16=         25/ 17=          4/ 18=          4
 / 19=          4/ 20=          9/ 21=          1/ 22=          2
 / 23=         10/ 24=          2/ 25=          5/ 26=         10
 / 27=         10/ 28=          1/ 29=          1/ 30=          2
 / 31=          2/ 32=          3/ 33=          4/ 34=          4
 / 35=          4/ 36=          4/ 37=          0/ 38=          0
 / 39=          0/ 40=          0/ 41=          1/ 42=          0
 / 43=          0/ 44=        100/ 45=          5/ 46=          0
 / 47=          0/ 48=          0/ 49=      42237/ 50=          0
 / 51=          0/ 52=          0/ 53=          0/ 54=          0
 / 55=          0/ 56=          0/ 57=          0/ 58=          0
 / 59=          0/ 60=          0/ 61=          0/ 62=          0
 / 63=          1/ 64=       5000/ 65=      64920/ 66=          0
 / 67=          0/ 68=          0/ 69=          0/ 70=          0
 / 71=          0/ 72=          0/ 73=          0/ 74=          0
 / 75=          0/ 76=          0/ 77=          0/


ENDING RANDOM SEED PAIRS--
      1=      59090      61605
      2=      40520      19505
      3=      25236      45167
      4=      63813      55151
      5=       4782      56549
      6=      38349      18839
      7=       8714      15697


DATA CHECK TOTALS--
     AAA=     -25028
     RHS=      -8655
     CCC=     -21397
     DDD=       1096
     XBAR=      5538
     XSTAR=     1478
     ROWTYP=      21
     VBLTYP=    1363
```

FIGURE 2. RIP

PROBLEM SUMMARY PRINTOUT

VARIABLE	XBAR(J)	XSTAR(J)
1	0.	.10000000E+01
2	0.	.50000000E+01
3	.51324707E-01	0.
4	0.	.10000000E+01
5	0.	0.
6	.10000000E+01	.10000000E+01
7	.10000000E+01	.10000000E+01
8	.55000000E+01	.50000000E+01
9	0.	.10000000E+01
10	.10000000E+01	0.
11	0.	.10000000E+01
12	.10000000E+01	0.
13	.10000000E+01	.10000000E+01
14	.30000000E+01	.20000000E+01
15	.10000000E+01	.10000000E+01
16	.10000000E+01	.10000000E+01
17	.10000000E+01	.10000000E+01
18	0.	0.
19	.25000000E+01	.30000000E+01
20	0.	0.
21	.10000000E+01	0.
22	0.	.10000000E+01
23	0.	0.
24	.35000000E+01	.50000000E+01
25	.50000000E+01	.46666667E+01

FIGURE 3. RIP PRINT LEVEL

2 SOLUTION PRINTOUT

```
*****************************
* GENERATED PROBLEM DATA *
*****************************

COLUMN DATA--

COL=     1,XBAR= 0.                ,XSTAR=  .10000000E+01,C= -.36014371E+03,D=          6,TYPE=  3
    /          7 IN ROW  10/          1 IN ROW   8/

COL=     2,XBAR= 0.                ,XSTAR=  .50000000E+01,C=  .98716907E+04,D=          8,TYPE=  4
    /          2 IN ROW   9/         12 IN ROW   2/        -12 IN ROW   3
    /         -5 IN ROW  10/         98 IN ROW  11/

                                        .
                                        .
                                        .

COL=    24,XBAR=  .35000000E+01,XSTAR=  .50000000E+01,C= -.58855191E+02,D=          8,TYPE=  3
    /          1 IN ROW  10/         10 IN ROW   2/

COL=    25,XBAR=  .50000000E+01,XSTAR=  .46666667E+01,C= -.26627663E+05,D=          5,TYPE=  3
    /        -30 IN ROW   4/        -27 IN ROW  12/         78 IN ROW  10
    /         -3 IN ROW   1/          3 IN ROW   7/       -249 IN ROW  11


RIGHT-HAND-SIDES--
    /       -37. IN ROW   1/         55. IN ROW   2/        -88. IN ROW   3
    /      -239. IN ROW   4/        202. IN ROW   5/          1. IN ROW   6
    /        91. IN ROW   7/       -179. IN ROW   8/       -120. IN ROW   9
    /       -55. IN ROW  10/       -573. IN ROW  11/        -94. IN ROW  12


ROW TYPES--
    /  1 ON ROW   1/ -1 ON ROW   2/  1 ON ROW   3/ -1 ON ROW   4
    / -1 ON ROW   5/  0 ON ROW   6/  0 ON ROW   7/  0 ON ROW   8
    / -1 ON ROW   9/  1 ON ROW  10/  0 ON ROW  11/  1 ON ROW  12
```

FIGURE 4. RIP PRINT LEVEL 3

PROBLEM DATA PRINTOUT

REFERENCES

1. Bazaraa, M. S. and C. M. Shetty, Nonlinear Programming: Theory and Algorithms, John Wiley and Sons, New York, 1979.

2. Blankenship, W. A., "A New Version of the Euclidean Algorithm," American Math Monthly, 70, 742-745, 1965.

3. Breu, Raymond and Claude-Alain Burdet, "Branch and Bound Experiments in Zero-One Programming," Mathematical Programming Study, 2, 1-50, 1974.

4. Fleisher, J. M., "Construction of Generalized Capital Budgeting Test Problems with Known Optimal Solutions," Computer Science Technical Report No. 260, University of Wisconsin, Madison, August, 1975.

5. Fleisher, J. M., "Construction of Plant Location Test Problems", Computer Science Technical Report No. 263, University of Wisconsin, Madison, December, 1975.

6. Fleisher, J. M. and R. R. Meyer, "New Sufficient Optimality Conditions for Integer Programming and their Application," Computer Sciences Technical Report No. 278, University of Wisconsin, Madison, October, 1976.

7. Lin, B. W., "Development of Controlled Computational Experiments on Integer Linear Programming Procedures, Ph.D. dissertation, Georgia Institute of Technology, 1975.

8. Lin, B.W. and R. L. Rardin, "Development of a Parametric Generating Procedure for Integer Programming Test Problems," Journal of the ACM, 24, 465-472, 1977.

APPENDIX: RIP FORTRAN CODE

```
CDCCDCPROGRAM RIPTST(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
C-----PROGRAM TO TEST EXECUTE THE RIP RANDOM INTEGER PROGRAM
C      GENERATOR
C
       IMPLICIT INTEGER (A-Y)
C
       DIMENSION AAA(2,5000),ACOL(37),RHS(17),CCC(37),DDD(37)
      *    ,
      *    ROWTYP(17),VBLTYP(37),XBAR(37),XSTAR(37),PARAM(77)
      *    ,NAM(20)
C
       DOUBLE PRECISION CCC,SOLSTR,SOLBAR,RHS
C
       DATA PARAM/
      *4H1ST ,4H2ND,4H    ,4H    ,4H    ,4H    ,4H    ,
      *4H    ,4H    ,4H    ,4H    ,3000,0,0,12,25,4,4,4,9,
      * 1,2,10,2,5,10,10,1,1,2,
      * 2,3,4,4,4,4,0,0,0,0,
      * 1,1,0,100,5,0,0,0,42237,0,
      * 0,0,0,0,0,0,0,0,0,0,
      * 0,0,0,3,5000,,6492,,
      * 0,0,0,0,0,0,0,0,0,0/
C
       DATA NAM/3H1ST,3H2ND,3H3RD,3H4TH,3H5TH,3H6TH,
      *    3H7TH,3H8TH,3H9TH,4H10TH,4H11TH,
      *    4H12TH,4H13TH,4H14TH,4H15TH,4H16TH,
      *    4H17TH,4H18TH,4H19TH,4H20TH/
C
       DO 9900 TEST=1,20
C
       PARAM(1)=NAM(TEST)
       PARAM(65)=6492
       IF(MOD(TEST,2).EQ.0)PARAM(65)=6561
       PARAM(49)=42237
       IF(MOD(TEST/2,2).EQ.1)PARAM(49)=13
       PARAM(11)=0
       IF(TEST.GT.4)PARAM(11)=1
       IF(TEST.LE.8)GO TO 8000
       PARAM(17)=0
       PARAM(18)=12
       PARAM(19)=0
       IF(TEST.LE.12)GO TO 8000
       PARAM(18)=0
       PARAM(19)=12
       IF(TEST.LE.16)GO TO 8000
       PARAM(19)=0
       PARAM(17)=12
       PARAM(20)=0
       PARAM(21)=0
       PARAM(22)=0
       PARAM(23)=12
       PARAM(24)=12
       IF(TEST.GT.17)GO TO 8000
       PARAM(30)=PARAM(30)+1
       PARAM(31)=PARAM(31)+1
       PARAM(32)=PARAM(32)+1
       PARAM(34)=PARAM(34)-1
       PARAM(35)=PARAM(35)-1
       PARAM(36)=PARAM(36)-1
C
 8000  CALL RIP(AAA,ACOL,RHS,CCC,DDD,ROWTYP,VBLTYP,XBAR,
      *    XSTAR,DENBAR,DENSTR,SOLBAR,SOLSTR,PARAM)
C
 8900  CONTINUE
C-----EXIT
 9000  CONTINUE
       STOP
       END
C
C-------------------------------------------------------------
C
C-----SUBROUTINE RIP(AAA,ACOL,RHS,CCC,DDD,ROWTYP,VBLTYP,
C    XBAR,
C     *    XSTAR,DENBR,DENSR,SNBAR,SNSTR,PARAM)
C-----MAIN SUPERVISING SUBROUTINE OF THE RIP RANDOM INTEGER
C      PROGRAMMING PROBLEM GENERATOR.  PARAMETERS OF THE CALL
C      ARE AS FOLLOWS--
C         AAA=THE AAA MATRIX NONZERO COEFFICIENTS IN THE
C            FORMAT
C            (1,XXX)=THE ROW NUMBER OF THE COEFFICIENT AND
C            (2,XXX)=THE (INTEGER) COEFFICIENT
C         ACOL=A VECTOR POINTING TO THE LAST 2-TUPLE OF AAA
C            CONTAINING
C            INFORMATION OF EACH COLUMN
C         RHS=THE RIGHT-HAND-SIDE VECTOR OF THE GENERATED
C            PROBLEM
C         CCC=THE COST ROW OF THE GENERATED PROBLEM
C         DDD=THE VECTOR OF UPPER BOUNDS FOR THE GENERATED
C            PROBLEM
C         ROWTYP=A VECTOR SHOWING THE CONSTRAINT TYPE OF
C            EACH ROW
C            (=-1 FOR .GE.,=0 FOR .EQ.,=+1 FOR .LE.)
C         VBLTYP=A VECTOR SHOWING THE TYPE OF EACH VARIABLE
C            (=3 FOR CONTINUOUS VARIABLES, =4 FOR GENERAL
C               INTEGER
```

```
C          VARIABLES,ED FOR  -1 INTEGER VARIABLES)
C       XBARETH  OPTIMAL LINEAR PROGRAMMING RELAXATION
C          SOLUTION
C       VECTOR (LINES THE DENOMINATOR DE XBAR)
C       XSTARETH (THOUGH POSSIBLY NOT THE ONLY) INTEGER
C          PROGRAM
C          SOLUTION VECTOR
C       SNBARETH  VALUE OF THE SOLUTION XBAR
C       SNSTRETH  VALUE OF THE SOLUTION XSTAR
C       PARAMETHE VECTOR OF INPUT PARAMETERS SUPPLIED TO
C          THE
C          RIP GENERATOR BY THE USER.
C
C     OF THE ABOVE SNBAR,SNSTR,RHS AND CCC ARE DOUBLE
C        PRECISION
C     AND ALL OTHER VARIABLES ARE INTEGER.
C
C     OTHER RIP VARIABLES AND VECTORS ARE DEFINED IN THE
C     CONTEXT WHERE THEY ARISE EXCEPT FOR THOSE WHICH
C        CORRESPOND
C     EXACTLY TO THE NAMES USED IN THE RARDIN AND LIN PAPER
C     DESCRIBING THE RIP GENERATOR.  SUCH VARIABLE NAMES ARE
C     REFERENCED BY TRIPLING THE VARIABLE NAME WITHOUT
C        FURTHER
C     DEFINITION.  MATRICES AAA,BBB,PPP,QQQ,SSS,TTT,RRR,R-
C        INVERSE,
C     ASTAR,C-INVERSE AND FFF OF THE RARDIN AND LIN PAPER
C        ARE
C     NOT ACTUALLY REPRESENTED AS MATRICES IN THE PROGRAM.
C     INSTEAD, THEIR NONZERO ENTRIES ARE LINKED IN LISTS.
C     THE PROGRAM VARIABLES BBB,PPP,QQQ,SSS AND TTT LIST
C     THE COLUMNS IN AAA BELONGING TO THE CORRESPONDING
C     SUBMATRICES.
C
C     THE ARRAY PASSED AS PARAMETER AAA IS EXTENSIVELY
C     USED BY THE RIP PROGRAM AS A SCRATCH WORK AREA.
C     NUMEROUS TEMPORARY VECTORS AND MATRICES ARE
C        DYNAMICALLY
C     ASSIGNED SECTORS OF THE SPECIFIED AAA DIMENSION AS
C     THE PROGRAM PROGRESSES.
C
C     THROUGHOUT MOST OF THIS DYNAMIC USE OF AAA, MATRICES
C     ARE STORED AS LINKED 3-TUPLES.  WITHIN EACH 3-TUPLE
C        COMPONENT (1,XXX)=THE ROW (OR COLUMN) OF A
C           NONZERO ENTRY IN A COLUMN (OR ROW)
C        COMPONENT (2,XXX)=THE NONZERO VALUE ITSELF
C        COMPONENT (3,XXX)=A POINTER TO THE NEXT ENTRY
C           FOR THIS COLUMN (OR ROW)
C           IF EQUALS 0 IF THERE IS NONE
C     INDIVIDUAL MATRICES CAN BE RETRIEVED THROUGH
C     THIS SCHEME BY RETAINING A LIST OF THE FIRST
C     3-TUPLE FOR EACH COLUMN (OR ROW).
C
C     THE RIP CODE IS MODULAR WITH A NUMBER
C     OF SUBROUTINES AND FUNCTIONS.  EACH SUCH SUBPROGRAM
C     HAS A NAME BEGINNING WITH THE LETTERS RIP SO THAT
C     A USER WILL NOT HAVE TO BE CONCERNED WITH DUPLICATING
C     ONE.  WITHIN THE CODE EACH SUBPROGRAM IS INTRODUCED
C        BY A
C     BRIEF DESCRIPTION OF ITS FUNCTION AND A DEFINITION
C     OF ALL ITS PARAMETERS.  THE FOLLOWING TABLE
C     PRESENTS AN OVERVIEW OF THE PURPOSE AND CALLING
C     STRUCTURE RELATING THE SUBPROGRAMS.
C
C--------+------------------+-------------+--------------+
C  NAME  I    PURPOSE       I  CALLS      I  CALLED BY   I
C--------+------------------+-------------+--------------+
C RIP    I MAIN ROUTINE     I RIPPH,RIPERR I USERS        I
C        I                  I RIPUNF       I PROGRAMS     I
C--------+------------------+-------------+--------------+
C RIPPH  I SUPERVISOR OF    I RIPPRM,RIPUNFI RIP          I
C        I ALL CALCULATION  I RIPSEQ,RIPTYPI              I
C        I                  I RIPPH1,RIPPH2I              I
C        I                  I RIPPH3,RIPPH4I              I
C        I                  I RIPORG,RIPVERI              I
C        I                  I RIPERR,RIPEUCI              I
C--------+------------------+-------------+--------------+
C RIPPH1 I EXECUTES PHASE I  I RIPUNF,RIPPRMI RIPPH        I
C        I                  I RIPADD,RIPSEQI              I
C        I                  I RIPELG,RIPDSTI              I
C        I                  I RIPDOT,RIPVPRI              I
C        I                  I RIPDEL,RIPRDTI              I
C--------+------------------+-------------+--------------+
C RIPPH2 I EXECUTES PHASE II I RIPUNF,RIPDSTI RIPPH        I
C        I                  I RIPSPL,RIPADDI              I
C        I                  I RIPVPR       I              I
C--------+------------------+-------------+--------------+
C RIPPH3 I EXECUTES PHASE    I RIPUNF,RIPRDTI RIPPH        I
C        I   III            I              I              I
C--------+------------------+-------------+--------------+
C RIPPH4 I EXECUTES PHASE IV I RIPUNF,RIPADDI RIPPH        I
C        I                  I RIPDEL,RIPRDTI              I
C        I                  I RIPVPR       I              I
C--------+------------------+-------------+--------------+
C RIPADD I STORES A 3-TUPLE  I NONE         I RIPPH1,RIPPH2I
C--------+------------------+-------------+--------------+
C RIPDEL I DELETES A VECTOR  I NONE         I RIPPH1,RIPPH4I
C        I OF 3-TUPLES       I              I              I
C--------+------------------+-------------+--------------+
C RIPDOT I DOTS 2 VECTORS    I NONE         I RIPPH1       I
```

```
C--------+----------------------+----------------+----------------+
C RIPDST I GENERATES RANDOM     I RIPUNF         I RIPPH1,RIPPH2I
C        I X-DISTANCE           I                I RIPNU        I
C--------+----------------------+----------------+----------------+
C RIPERR I PRINTS ERROR         I NONE           I RIP,RIPPH    I
C        I MESSAGES             I                I              I
C--------+----------------------+----------------+----------------+
C RIPEUS I SOLVES DIOPHAN-      I NONE           I RIPPH1,RIPPH I
C        I TINE EQUATIONS       I                I              I
C--------+----------------------+----------------+----------------+
C RIPNU  I GENERATES DENOM.     I RIPDST,RIPUNFI RIPPH2         I
C        I CONTINUOUS XSTAR     I                I              I
C--------+----------------------+----------------+----------------+
C RIPORG I REORGANIZES MATRIXI NONE              I RIPPH        I
C        I WORK AREA            I                I              I
C--------+----------------------+----------------+----------------+
C RIPPRM I RETURNS PRIMES       I NONE           I RIPPH1,RIPPH I
C--------+----------------------+----------------+----------------+
C RIPROT I DOTS 2 VECTORS       I NONE           I RIPPH3,RIPPH1I
C        I (1 IS FLOATING PT)I                   I RIPPH4       I
C--------+----------------------+----------------+----------------+
C RIPSEQ I RANDOMIZES ORDER     I RIPUNF         I RIPPH,RIPPH1 I
C        I OF A LIST OF NOS.    I                I RIPSPL       I
C--------+----------------------+----------------+----------------+
C RIPSPL I RANDOMLY DIVIDES     I RIPSEQ,RIPUNFI RIPPH2         I
C        I A WEIGHTED TOTAL     I                I              I
C--------+----------------------+----------------+----------------+
C RIPTYP I RANDOMLY ASSIGNS     I RIPUNF         I RIPPH        I
C        I TYPES TO COLUMNS     I                I              I
C--------+----------------------+----------------+----------------+
C RIPUNF I GENERATES RANDOM     I NONE           I ALMOST ALL   I
C        I UNIFORM VARIATES     I                I RIP ROUTINES I
C--------+----------------------+----------------+----------------+
C RIPVER I VERIFIES PROBLEM     I NONE           I RIPPH        I
C        I PROPERTIES           I                I              I
C--------+----------------------+----------------+----------------+
C RIPVPR I PRINTS MATRICES      I NONE           I RIPPH1,RIPPH2I
C        I                      I                I RIPPH4       I
C--------+----------------------+----------------+----------------+
C
      SUBROUTINE RIP(AAA,ACOL,RHS,CCC,DDD,ROWTYP,VBLTYP,
     *   XBAR,
     *   XSTAR,DENBR,DENSR,SNBAR,SNSTR,PARAM)
C
      IMPLICIT INTEGER (A-Y)
C
      COMMON /RIPCOM/SOLBAR,SOLSTR,TOLMOV,TOLOFL,TOLOPT,
     *   TOLSTP,TOLVER,BCONT,BDCONT,BDINT,BDSLK,BFRAC,
     *   BINT,BSLK,BTITST,CCREOT,CORTOP,DATTYF,DELTM,
     *   DENSA,DENSF,DENSP,DENSPQ,DENSQ,DETERM,
     *   DIFDEG,DIMFRM,DISFAT,ECONT,EFRAC,EINT,ESLK,
     *   FRACI,INFIN,LIMA,LIMALP,LIMD,LIMF,LIMR,LIMSLK,
     *   LIMVMN,LIMVMX,LIMWMN,LIMWMX,LUPR,LVLPRT,
     *   MAXCON,MAXCUT,MAXINT,MAX 1,MMM,MU,
     *   NNN,NNNP,NNNQ,NNNS,NNNT,NUMERR,NXTMAT,PRECN,PSLK,
     *   RTPSLK,SEEC(2,7),SSLK,TOLMU,VERIFY,VERSN
C
      DIMENSION AAA(1),ACOL(1),RHS(1),CCC(1),DDD(1),
     *   ROWTYP(1),
     *   VBLTYP(1),XBAR(1),XSTAR(1),PARAM(1),LIMPMN(55),
     *      LIMPMX(55)
C
      DOUBLE PRECISION CCC,SNBAR,SNSTR,RHS
      DOUBLE PRECISION SOLBAR,SOLSTR,TOLMOV,TOLOFL,TOLOPT,
     *   TOLSTP,TOLVER
C
C-----LOWER LIMITS ON INPUT PARAMETERS
      DATA LIMPMN/[illegible],5 ,[illegible],10,0,0,[illegible],0,0,3,0,0,
     *   [illegible],0,3,1,1,1,1,[illegible],
     *   [illegible],3,[illegible],1,[illegible],
     *   [illegible],1000,[illegible]/
C
C-----UPPER LIMITS ON INPUT PARAMETERS
      DATA LIMPMX/1,9.0,1000,0,500,1000,500,500,500,500,494,
     *   494,
     *  500,500,900,1000,1000,494,494,494,494,500,500,500,
     *     500,
     *  500,[illegible],0,1,1000,500,500,3000,0,65535,65535,
     *     65535,
     *  65535,65535,65535,65535,65535,65535,65535,65535,
     *     65535,
     *  65535,65535,99,03,110000,100000/
C
C-----SET VERSION AND PRECISION INDICATORS
      VERSN=800725
      PRECN=2
C
C        ***SECTION TO CHECK PARAMETER VALUES FOR ERRORS***
C
      NUMERR=0
C-----ESTABLISH THE PRINTER
      LUPR=PARAM(62)
      IF(PARAM(62).LE.0)LUPR=6
C-----ESTABLISH THE NUMBER OF PARAMETERS
      MAXPAR=65
C-----CHECK UPPER AND LOWER LIMITS ON MAIN PARAMETERS
      DO 2500 PAR=11,MAXPAR
      IF(PARAM(PAR).GE.LIMPMN(PAR-10)
     *   .AND.PARAM(PAR).LE.LIMPMX(PAR-10)
     *   .OR.PARAM(PAR).EQ.VERSN)GO TO 2500
```

```
      CALL RIPERR(PAR)
 2500 CONTINUE
C-----CHECK WITH DIAGONAL INPUT IF ANY
      MMM=MAX (PARAM(25),1)
      OLDPAR=PARAM(65)
      DO 2600 III=1,MMM
      IF(PARAM(III+65).NE.OLDPAR)GO TO 2550
      CALL RIPERR(III+65)
      GO TO 2600
 2550 IF(OLDPAR.LE. )GO TO 2600
C-----CHECK IF EACH DIVIDES THE NEXT
      IF(MOD(PARAM(III+65),OLDPAR).NE. )CALL RIPERR(906)
      OLDPAR=PARAM(III+65)
 2600 CONTINUE
C-----CHECK RELATIONSHIPS B TWEEN PARAMETERS AND SAVE
C     THOSE WITH INTERNAL NAMES
      IF(PARAM(17)+PARAM(18)+PARAM(19).NE.MMM)CALL RIPERR
     *   (901)
      IF(PARAM(20)+PARAM(21)+PARAM(22).NE.MMM)CALL
     *   RIPERR(902)
      IF(PARAM(23)+PARAM(24).NE.MMM)CALL RIPERR(903)
      BSLK=PARAM(21)+PARAM(22)
      IF(PARAM(24).GT.BSLK)CALL RIPERR(908)
      BDSLK=PARAM(21)
      IF(PARAM(20)+PARAM(19).LT.BSLK)CALL RIPERR(909)
      IF(PARAM(18)+PARAM(19).LT.PARAM(24))CALL RIPERR(908)
      IF(PARAM(24).GT.BSLK)CALL RIPERR(910)
      NNNP=PARAM(33)
      NNNQ=PARAM(34)
      NNNS=PARAM(35)
      NNNT=PARAM(36)
      MAXCON=PARAM(25)
      MAXINT=PARAM(26)
      MAX 1=PARAM(27)
      NNN=PARAM(10)
      IF(MAXCON+MAX 1+MAXINT.NE.NNN)CALL RIPERR(912)
      BDCONT=PARAM(28)
      BCONT=PARAM(29)+BDCONT
      BDINT=PARAM(30)
      BINT=PARAM(31)+BDINT
      IF(BINT+BCONT.LT.3)CALL RIPERR(911)
      BFRAC=PARAM(32)
      SUM=BCONT+BINT+BFRAC
      IF(SUM.NE.PARAM(28))CALL RIPERR(905)
      IF(SUM+NNNP+NNNQ+NNNS+NNNT.NE.NNN)CALL RIPERR(904)
      DATTYP=PARAM(11)
      MAXOUT=PARAM(45)
      VERIFY=PARAM(41)
      DENSA=PARAM(12)
      DETERM=PARAM(65)
      IF(PARAM(29).LT.PARAM(28)+PARAM(29))CALL RIPERR(916)
      IF(PARAM(31).GT.PARAM(26))CALL RIPERR(914)
      IF(PARAM(28)+PARAM(27).LT.PARAM(3.)+PARAM(31)
     *   +PARAM(32))CALL RIPERR(915)
      LIMD=PARAM(42)
      DO 2900 SEQ=1,7
      PAR=46+2*SEQ
      SEED(1,SEQ)=PARAM(PAR)
      SEED(2,SEQ)=PARAM(PAR+1)
      IF(SEED(2,SEQ).LE.0)GO TO 2950
      IF(MOD(SEED(2,SEQ),2).NE.1)CALL RIPERR(917)
 2900 CONTINUE
      LVLMAT=PARAM(63)
      MD=PARAM(44)
      LIMA=PARAM(13)
      DISRAT=PARAM(46)
      IF(PARAM(32).GT.PARAM(45))CALL RIPERR(920)
C
C-----SET NONAUTOMATIC DEFAULTS
      IF(LIMD.EQ.0)LIMD=10
      IF(LIMA.EQ.0)LIMA=50
      IF(DISRAT.EQ.0)DISRAT=100
      PROD=1
      DO 2950 III=1,MMM
      PROD=PROD*PARAM(65+III)
 2950 CONTINUE
      IF(PROD.EQ.0.AND.DETERM.EQ.0)CALL RIPERR(907)
      IF(PROD.NE.0)DETERM=PROD
C
C-----ABORT RUN ON PARAMETER ERRORS
 2990 CALL RIPERR(0)
C
C        ***SECTION TO SET NON PARAMETER PROGRAM LIMITS***
C
      DIMPRM=100
      DENSP=2*DENSA
      RIPSLK=100
      DENSQ=DENSP
      DENSPQ=DENSP
      LIMALP=MAX0(1,LIMA/4)
      LIMP=10
      LIMR=LIMALP
      LIMSLK=MMM*LIMD/3
      LIMVMN=20
      LIMVMX=200
      INFIN=10000000
      LIM*MN=10
      LIM*MX=100
      TOLMU=20
      TOLMOV=10
```

```
C-----FIX BREAK OF THE N INTEGER AND FRACTIONAL ROWS
      FRACI=PINT-BFRAC+1
      IF(BCONT-BCONT.GT..1)FRACI=FRACI-RIPUNF(1,BCONT-
     *    BCONT,1)
C-----REVISE BASIC ROW COUNTS TO EFFECTIVE TREATMENT VALUES
      SSLK=BSLK
      EINT=BINT
      ECONT=BCONT
      EFRAC=BFRAC
      IF(PARAM(19).NE.MMM.AND.PARAM(16).NE.MMM)GO TO 3220
      ECONT=ECONT+EINT
      EINT=0
      GO TO 322
 3200 IF(BCONT.GE.MMM/4)GO TO 322
      IF(BINT.LE.2)GO TO 3220
C-----SHIFT SOME INTEGER-INTEGER ROWS TO EFFECTIVE
C    CONTINUOUS
      SHIFT=RIPUNF(1,BINT*2/3,1)
      ECONT=ECONT+SHIFT
      EINT=EINT-SHIFT
C-----SET SLACK COUNTS
 3220 PSLK=
      BTEST=BSLK-PARAM(24)
      SSLK=PARAM(16)+PARAM(19)-BSLK-PSLK
      NNNS=NNNS+SSLK
C
C        ***SECTION TO SETUP CALL TO MAIN WORKING
C           SUBROUTINE***
C
C-----SET MATRIX WORK VECTOR SUBSCRIPTS
 4000 BBB=PARAM(24)*2-MMM+1
      PPP=BBB-NNNP
      QQQ=PPP-NNNQ
      SSS=QQQ-NNNS-NNTP
      TTT=SSS-NNNT-NNTQ
      UBAR=TTT-PRECN*MMM
      IF(UBAR/2*2.EQ.UBAR)UBAR=UBAR-1
      USTAR=UBAR-PRECN*MMM
      VSTAR=USTAR-PRECN*MAXCUT
      OMEGA=VSTAR-PRECN*(MMM+MAXCUT)
      SCRCH=OMEGA
      NU=SCRCH-NNN
      ACUT=NU-NNN
      DELTA=ACUT-MMM
      CORBOT=3*((DELTA-1)/3)
C-----CALCULATE THE WORK VECTOR AREA REQUIRED ABOVE MATR3
      PH1=1+MAX(PRECN*NMM,DIMPRM)+6*MMM+2*MAXCUT
      PH2=6*NNNQ+4*NNNP+2*MAXCUT+MMM
      PH3=2*MAXCUT+1+PRECN*(3*MAXCUT+3*MMM+2*NNNP+2*NNNQ)
      PH4=2*MAXCUT+1+PRECN*MMM+MMM
      CORTOP=MAX(PH1,PH2,PH3,PH4)
      CORTOP=(CORTOP/3+1)*3
C-----ESTIMATE CORE REQUIREMENTS
      ZWORK=3*DENSA
      ZWORK=ZWORK/100*NNN*(MAXCUT+MMM)+CORTOP
      IF(ZWORK.LT.CORBOT)GO TO 9000
      CALL RIPERR(921)
      CALL RIPERR(J)
C
C-----CALL THE MAIN SUBROUTINE
 9000 CALL RIPPH(XBAR,XSTAR,ACOL,AAA(ACUT),AAA,AAA,AAA,
     *    AAA(BBB),
     *    AAA(PPP),AAA(QQQ),AAA(SSS),AAA(TTT),CCC,
     *    DDD,RHS,ROWTYP,VBLTYP,AAA(OMEGA),AAA(DELTA),
     *    AAA(UBAR),AAA(USTAR),AAA(VSTAR),PARAM,AAA(NU),
     *    DENBAR,DENSTR)
C
      SNBAR=SOLBAR
      SNSTR=SOLSTR
      DENBR=DENBAR
      DENSR=DENSTR
C
C        ***EXIT SECTION***
C
 9000 CONTINUE
      RETURN
      END




C
C-------------------------------------------------------------
C
C-----SUBROUTINE RIPPH(XBAR,XSTAR,ACOL,ACUT,MATR1,MATR2,
C   MATR3,BBB,
C     *    PPP,QQQ,SSS,TTT,CCC,DDD,RHS,ROWTYP,VBLTYP,OMEGA,
C     *    DELTA,UBAR,USTAR,VSTAR,PARAM,NU,DENBAR,DENSTR)
C-----SUBROUTINE TO SUPERVISE PROCESSING OF THE FOUR PHASES
C     OF THE RIP RANDOM INTEGER PROGRAM GENERATOR.
C        PARAMETERS
C     NOT DEFINED IN THE LIN AND RANDIN PAPER ON THE
C        GENERATOR
C     ARE AS FOLLOWS--
C        MATR1,MATR2,MATR3=THE (EQUIVALENCED THROUGH THE
C           CALL)
C           RIP GENERAL MATRIX WORK AREA IN 1-TUPLE,2-TUPLE,
C           AND 3-TUPLE FORM RESPECTIVELY
```

```
C       RHS=THE PROBLEM RIGHT-HAND-SIDE VECTOR (LITTLE B
C       ROWTYP=A V CTOR DEFINING THE ROW TYPES (-1=.GE.,
C          =0.,+1=.LE.)
C       VBLTYP=A V CTOR DEFINING THE VARIABLE TYPE AT EACH
C          SUBSCRIPT
C          (=1 FOR .LE. SLACK,=2 FOR .GE. SLACK,=3 FOR
C             CONTINUOUS,
C           =4 FOR GENERAL INTEGER,=5 FOR 0-1 INTEGER)
C       OMEGA=A VECTOR SHOWING THE REQUIRED ORIENTATIONS OF
C          COLUMNS OF THE FFF MATRIX IN RIP. (=-1 ON .LE.
C             AND
C          ASSIGNABLE 0-0. INTEGER ROWS, =+1 ON .GE. AND
C             ASSIGN 0
C          .EQ. INTEGER ROWS, =0 ON CONTINUOUS AND SLACK
C             ROWS)
C          NOTE: OMEGA SPACE IS USED IN LATE PHASES AS
C             SCRATCH
C       PARAM=THE VECTOR OF INPUT PARAMETERS TO RIP
C       DENBAR=THE DENOMINATOR OF COMPONENTS OF XBAR
C          USED TO AVOID ROUNDOFF ERRORS
C       DENSTR=THE DENOMINATOR OF COMPONENTS OF XSTAR
C          USED TO AVOID ROUNDOFF ERRORS
C       ACUT=A POINTER VECTOR SIMILAR TO ACOL THAT RECORDS
C          THE LOCATION OF CUT COEFFICIENTS
C     IN ADDITION TO THESE PARAMETERS, MANY VECTORS
C     ARE CREATED FOR SPECIFIC PHASES OF RIP GENERATION
C     BY COMPUTING LOAD POINTS WITHIN MATR1.  THOSE
C     INVOLVED IN SEVERAL SEPARATE CALLS ARE THE FOLLOWING--
C       ACOL=THE 1ST MATRIX 3-TUPLE OF A COLUMN IN THE
C          AAA MATRIX (WHILE IT IS IN 3-TUPLE (LINKED)
C             FORM)
C       LROW=THE 1ST MATRIX 3-TUPLE OF ROWS OF THE MATRIX
C          LLL
C       FROW=THE 1ST MATRIX 3-TUPLE OF ROWS OF THE MATRIX
C          FFF
C       SCRCH=A WORK VECTORS USED FOR MANY INTERMEDIATE
C          RESULTS
C
      SUBROUTINE RIPPH(XBAR,XSTAR,ACOL,ACUT,MATR1,MATR2,
     *   MATR3,BBB,
     *   PPP,QQQ,SSS,TTT,CCC,DDD,RHS,ROWTYP,VBLTYP,OMEGA,
     *   DELTA,UBAR,USTAR,VSTAR,PARAM,NU,DENBAR,DENSTR)
C
      IMPLICIT INTEGER (A-Y)
C
      COMMON /RIPCOM/SOLBAR,SOLSTR,TOLMOV,TOLOFL,TOLOPT,
     *   TOLSTP,TOLVER,BCONT,BDCONT,BDINT,BDSLK,BFRAC,
     *   DINF,ESLK,BTITLT,COREQT,CUTOP,DATTYP,DELTA,
     *   DENBAR,DENSP,DENSTR,DENSPD,DENSD,DETERR,
     *   DIFDEG,DIMPRM,DISMAT,ECONT,EFRAC,EINT,ESLK,
     *   FRACT,INFIN,LIMA,LIMALP,LIMD,LIMF,LIMR,LIMSLK,
     *   LIMVAR,LIMIMX,LIMMIN,LIMMAX,LOPP,LVLPRT,
     *   MAXCON,MAXCUT,MAXINT,MAX1,MMM,NU,
     *   NNN,NNNP,NNND,NNNS,NNNT,NUMEPR,NXTMAT,PRECN,PSLK,
     *   RIPSLK,SEED(2,7),SSLK,TOLMU,VERIFY,VERSN
C
      DOUBLE PRECISION SOLBAR,SOLSTR,TOLMOV,TOLOFL,TOLOPT,
     *   TOLSTP,TOLVER
C
      DIMENSION XBAR(1),XSTAR(1),ACOL(1),ACUT(1),MATR1(1),
     *   MATR2(2,1),
     *   MATR3(3,1),BBB(1),PPP(1),QQQ(1),SSS(1),TTT(1),
     *   CCC(1),DDD(1),RHS(1),ROWTYP(1),VBLTYP(1),OMEGA(1),
     *   DELTA(1),UBAR(1),USTAR(1),VSTAR(1),PARAM(1),NU(1)
C
      DOUBLE PRECISION RWORK,USTAR,UBAR,VSTAR,RHS,CCC,RELXB,
     *   RELXS
C
      EQUIVALENCE (INTGNO,ZRELNO)
C
      DATA LITB,LITF,LITQ,LITS,LITT
     *   /1HB,1HP,1HQ,1HS,1HT/
C
C        ***SECTION TO SET PREVIOUSLY UNSET PARAMETERS***
C
C-----DELTAS OF THE SMITH NORMAL FOR--CHECK IF PRESENT
      IF(PARAM(65).EQ.0)GO TO 1460
C-----INITIALIZE DELTA TO A VECTOR OF ONES
      DO 1310 III=1,MMM
      DELTA(III)=1
 1310 CONTINUE
C-----FACTOR THE DETERMINANT TO BUILD THE DELTAS
      PROD=DETNO
      DO 1390 NUM=1,DIMPRM
      III=MMM
      PRIME=RIPPRM(NUM)
      IF(PRIME.GT.PROD)GO TO 1500
 1350 IF(MOD(PROD,PRIME).NE.0)GO TO 1390
      PROD=PROD/PRIME
      DELTA(III)=DELTA(III)*PRIME
      IF(PROD.EQ.1)GO TO 1500
      III=III-1
      IF(III.LE.ESLK)III=MMM
      GO TO 1350
 1390 CONTINUE
C-----DETERMINANT NOT SUITABLE PRODUCT OF PRIMES
      CALL RIPERR(913)
 1395 CALL PIMERR(6)
      GO TO 1500
C
```

```
C-----SAVE INPUT PARAMETERS AS SPECIFIED DELTAS
 1400 DO 1420 III=1,MMM
      DELTA(III)=PARAM(III+20)
      IF(III.GT.ESLK)GO TO 1420
      IF(DELTA(III).NE.1)CALL RIPERR(918)
 1420 CONTINUE
      GO TO 1390
C
C-----FINAL DELTA CHECKS
 1500 LIMSLK=MAX0(LIMSLK,.96*665536/DELTA(MMM))
      IF(DELTA(MMM-1).NE.1)GO TO 1510
      CALL RIPERR(918)
      GO TO 1390
C
C-----SET BAR DENOMINATOR
 1510 DELTM=DELTA(MMM)
C
C-----GENERATE RANDOM NUMBER SEEDS IF NECESSARY
      DO 1590 SED=5,7
      IF(SEED(2,SED).NE.0)GO TO 1590
      SEED(1,SED)=RIPUNF(0,65535,1)
      SEED(2,SED)=2*RIPUNF(0,32767,1)+1
 1590 CONTINUE
C
C         ***SECTION TO ALLOCATE ROW AND COLUMN TYPES***
C
C
C-----SET DIFFICULTY DEGREE
      IF(DATTYP.EQ.0)DIFDEG=1
      IF(DATTYP.EQ.1)DIFDEG=2
      IF(DATTYP.EQ.1.AND.(PARAM(17)*2.GE.MMM.OR.PARAM(19)
     *    *2.GE.MMM))
     *    DIFDEG=3
      IF(DATTYP.EQ.1.AND.PARAM(19)*10.GE.MMM*9)DIFDEG=4
C-----INITIALIZE ALL ROWS AS IF .LE.
      DO 2320 III=1,MMM
      ROWTYP(III)=1
 2320 CONTINUE
C-----INITIALIZE ALL CUT ROWS TO .GE.
      DO 2322 CUT=1,MAXCUT
      ROWTYP(MMM+CUT)=-1
 2322 CONTINUE
      REMFRC=EFRAC
      REMEQ=PARAM(17)
      DO 2325 ICOMPL=1,2
      IF(REMEQ.LE.0)GO TO 2500
      III=MMM-ICOMPL+1
      ROWTYP(III)=0
      REMEQ=REMEQ-1
      REMFRC=REMFRC-1
 2325 CONTINUE
      IF(REMEQ.LE.0)GO TO 2550
C-----PLACE THE EQUALITIES--1ST THOSE IN THE ECONT RANGE
      CALL RIPSEQ(MATR1,ECONT,1,ECONT+ESLK+EINT)
      PSN=ECONT-PSLK
C-----LOAD OTHER EQUALITY CANDIDATES INTO RANDOMIZATION AREA
      IF(EINT.LE.0)GO TO 2350
      DO 2340 III=1,EINT
      PSN=PSN+1
      MATR1(PSN)=III+ESLK
 2340 CONTINUE
 2350 IF(REMFRC.LE.0)GO TO 2370
      DO 2360 III=1,REMFRC
      PSN=PSN+1
      MATR1(PSN)=MMM-III-1
 2360 CONTINUE
C-----RANDOMIZE CANDIDATE LIST
 2370 CALL RIPSEQ(MATR1,PSN,1,0)
      DO 2390 PSN=1,REMEQ
      III=MATR1(PSN)
C-----TAG AN EQUALITY
      ROWTYP(III)=0
 2390 CONTINUE
C
C-----PLACE THE .GE. INEQUALITIES
 2500 IF(PARAM(18).LE.0)GO TO 2600
      MAX=MMM-1
      IF(PARAM(18).EQ.MMM)MAX=MMM
      CALL RIPSEQ(MATR1,MAX,1,MAX)
      NUMGE=0
      DO 2530 PSN=1,MAX
      III=MATR1(PSN)
      IF(ROWTYP(III).EQ.0) GO TO 2530
      NUMGE=NUMGE+1
      ROWTYP(III)=-1
      IF(NUMGE.GE.PARAM(18))GO TO 2600
 2530 CONTINUE
C
C-----SET OMEGAS ALTERNATELY
 2600 ORIEN=(-1)**PARAM(17)
      IF(PARAM(17).GE.2)GO TO 2630
      IF(PARAM(18).LE.0)GO TO 2630
      ORIEN=-ORIEN
 2630 DO 2650 III=1,MMM
      OMEGA(III)=1
      IF(III.LE.ESLK.OR.III.GT.ESLK+EINT.AND.III.LE.
     *    MMM-EFRAC)GO TO 2650
      IF(ROWTYP(III).EQ.0)GO TO 2640
      OMEGA(III)=-ROWTYP(III)
      GO TO 2650
```

```
 2040 ORIGN=-ORIGN
      OMEGA(III)=ORIGN
 2050 CONTINUE
      IF(PARAM(17).GE.1)OMEGA(MMM)=-OMEGA(MMM-1)
C
C-----PREPARE TO ASSIGN COLUMN TYPES AND POSITIONS
      SLKPSN=0
      NUMINT=0
      NUMCON=0
      NUMINT=0
      MAX=PARAM(14)+PARAM(15)
      IF(MAX.LE.0)GO TO 29
      CALL RIPSEQ(MATR1,MAX,1,NNN+MAX)
C-----ASSIGN BBB SLACKS
      IF(BSLK.LE.0)GO TO 2820
      DO 2810 BPSN=1,BSLK
      SLKPSN=SLKPSN+1
      SLKJJJ=MATR1(SLKPSN)
      BBB(BPSN)=SLKJJJ
      VBLTYP(SLKJJJ)=2
      IF(CONTYP(BPSN).LT.0)VBLTYP(SLKJJJ)=1
 2810 CONTINUE
C-----ASSIGN PPP SLACKS
 2820 IF(PSLK.LE.0)GO TO 2840
      DO 2830 PPSN=1,PSLK
      SLKPSN=SLKPSN+1
      SLKJJJ=MATR1(SLKPSN)
      PPP(PPSN)=SLKJJJ
      VBLTYP(SLKJJJ)=1
 2830 CONTINUE
C-----ASSIGN SSS SLACKS
 2840 IF(SSLK.LE.0)GO TO 2860
      DO 2850 SPSN=1,SSLK
      SLKPSN=SLKPSN+1
      SLKJJJ=MATR1(SLKPSN)
      SSS(SPSN)=SLKJJJ
      VBLTYP(SLKJJJ)=1
 2850 CONTINUE
C
C-----RANDOMIZE DECISION VARIABLE NUMBERS TO ASSIGN TYPES
 2860 CALL RIPSEQ(MATR1,NNN,1,NNN)
      JPSN=0
C-----ASSIGN INTERIOR INTEGER BBB COLUMNS
      IF(BINT+BDINT.LE.0)GO TO 2910
      MINB=BSLK+BDINT+1
      MAXB=BSLK+BINT
      CALL RIPTYP(VBLTYP,MATR1,JPSN,BBB,MINB,MAXB,INFIN,
     *   NUMINT,INFIN)
C-----ASSIGN DECIMAL INTEGER BBB COLUMNS
 2910 IF(BDINT.LE.0)GO TO 2920
      MINB=BSLK+1
      MAXB=BSLK+BDINT
      CALL RIPTYP(VBLTYP,MATR1,JPSN,BBB,MINB,MAXB,NUMCI,
     *   NUMINT,INFIN)
C-----ASSIGN CONTINUOUS BBB COLUMNS
 2920 IF(BCONT.LE.0)GO TO 2930
      MINB=BSLK+BDINT+1
      MAXB=MMM-BFRAC
      CALL RIPTYP(VBLTYP,MATR1,JPSN,BBB,MINB,MAXB,INFIN,
     *   INFIN,NUMCON)
C-----ASSIGN FRACTIONAL INTEGER BBB
 2930 IF(BFRAC.LE.0) GO TO 2940
      MINB=MMM-BFRAC+1
      CALL RIPTYP(VBLTYP,MATR1,JPSN,BBB,MINB,MMM,NUMCI,
     *  NUMINT,INFIN)
C-----ASSIGN REMAINING PPP COLUMNS
 2940 IF(NNNP-PSLK.LE.0)GO TO 2950
      MINP=PSLK+1
      CALL RIPTYP(VBLTYP,MATR1,JPSN,PPP,MINP,NNNP,NUMCI,
     *   NUMINT,NUMCON)
C-----ASSIGN QQQ COLUMNS
 2950 CALL RIPTYP(VBLTYP,MATR1,JPSN,QQQ,1,NNNQ,NUMCI,
     *   NUMINT,NUMCON)
C-----ASSIGN REMAINING SSS COLUMNS
      IF(NNNS-SSLK.LE.0)GO TO 2970
      MINS=SSLK+1
      CALL RIPTYP(VBLTYP,MATR1,JPSN,SSS,MINS,NNNS,NUMCI,
     *   NUMINT,NUMCON)
C-----ASSIGN TTT COLUMNS
 2970 CALL RIPTYP(VBLTYP,MATR1,JPSN,TTT,1,NNNT,NUMCI,
     *   NUMINT,NUMCON)
      IF(LVLPRT.LT.4)GO TO 3160
C-----PRINT HEADER INTRODUCING HIGH PRINT LEVEL PRINTOUT
      WRITE(LUPR,3005)(PARAM(KKK),KKK=1,10),VERSN,PRECN
 3005 FORMAT(1H1/11X,11(5H*****),2H**
     *   /11X,1H*,10X,31HRIP PROBLEM GENERATION PRINTOUT
     *   ,14X,1H* /11X,1H*,10X,5HNAME=,10A4,1H*
     *   /11X,1H*,10X,12HRIP VERSION=,I6,1H-,I1,25X,1H*
     *   /11X,2H**,11(5H*****) /1X)
C-----PRINT SUBMATRIX ASSIGNMENTS
 3010 FORMAT(/3H M=,I4,3H,N=,I4,4H,NP=,I4,
     *   4H,NQ=,I4,4H,NS=,I4,4H,NT=,I4,8H,MAXCUT=,I5)
      WRITE(LUPR,3010)MMM,NNN,NNNP,NNNQ,NNNS,NNNT,MAXCUT
 3020 FORMAT(/21H COLUMNS IN SUBMATRIX,1X,A1
     *   /(6X,12I5))
 3030 FORMAT(/6H BSLK=,I5,6H,ESLK=,I5,6H,BINT=,I5,6H,EINT=,
     *   I5,
     *   7H,BCONT=,I5,7H,ECONT=,I5,7H,BFRAC=,I5,7H,EFRAC=,I5
     *   /7H FRAC1=,I5,8H,DIFDEG=,I5)
      WRITE(LUPR,3030)BSLK,ESLK,BINT,EINT,BCONT,ECONT,
```

```
     *    IFRAC1,FRAC,FRAC1,LIFLAG
      WRITE(LUPR,3 25)LITM,(BBB(PSN),PSN=1,MMM)
      WRITE(LUPR,3 25)LITP,(PPP(PSN),PSN=1,NNNP)
      WRITE(LUPR,3 25)LITQ,(QQQ(PSN),PSN=1,NNNQ)
      WRITE(LUPR,3 25)LITS,(SSS(PSN),PSN=1,NNNS)
      WRITE(LUPR,3 25)LITT,(TTT(PSN),PSN=1,NNNT)
C
C        ***SECTION TO PREPARE THE 3-TUPLE MATRIX AREA
C           FOR USE BY PHASES I, II AND III***
C
C-----INITIALIZE 3-TUPLE AREA AS A FREE LINKED LIST
 3100 MIN=CORTOP/3+1
      MAX=CORBOT/3
      DO 3200 PSN=MIN,MAX
      MATR3(3,PSN)=PSN-1
 3200 CONTINUE
      MATR3(3,MIN)=0
      NXTMAT=MAX
C
C        ***SECTION TO CALL THE PHASE I SUBROUTINE***
C
C-----INITIALIZE POINTERS TO SHOW ALL VECTORS EMPTY AND
C   ALL NU=1
      MAXJJJ=NNN+PARAM(18)+PARAM(19)
      DO 3350 JJJ=1,MAXJJJ
      ACOL(JJJ)=0
      IF(JJJ.GT.NNN)GO TO 3350
      ACOL(JJJ)=0
      NU(JJJ)=1
 3350 CONTINUE
C-----SET WORK VECTOR SUBSCRIPTS FOR PHASE I
      FROW1=1
      LROW1=FROW1+MAXCOL
      SCRCH1=LROW1+MAXCOL
      IF(SCRCH1/2*2.EQ.SCRCH1)SCRCH1=SCRCH1+1
      SCRCH2=SCRCH1+MAXL(DIMPRM,PRECN*MIN)
      SCRCH3=SCRCH2+MMM
      ETA=SCRCH3+MMM
      TAU=ETA+MMM
      KROW1=TAU
      KINVR1=ETA
      GGG=TAU+MMM
      SCRCH4=GGG+MMM
C
C-----CALL THE SUBROUTINE
      CALL KIPPH1(MATR3,XBAR,XSTAR,ACOL,NU,DELTA,BBB,
     *    MATR1(SCRCH1),
     *    MATR1(SCRCH2),MATR1(SCRCH3),MATR1(ETA),MATR1(TAU),
     *    OMEGA,ROWTYP,VBLTYP,MATR1(FROW1),MATR1(LROW1),
     *    MATR1(KROW1),MATR1(KINVR1),MATR1(GGG),DDD,
     *       MATR1(SCRCH4),
     *    MATR1(SCRCH1),MATR3,RHS)
C
C        ***SECTION TO CALL THE PHASE II SUBROUTINE***
C
C-----SET WORK VECTOR SUBSCRIPTS FOR PHASE II
      DELQ=LROW1+MAXCOL
      DELX=DELQ+NNNQ
      SCRCH1=DELX+NNNQ+NNNP
      SCRCH2=SCRCH1+NNNQ+NNNP
      DELB=SCRCH2+NNNQ+NNNP
      LIVX=DELB+MMM
      LIVQ=LIVX+NNNP+NNNQ
C
C-----CALL THE SUBROUTINE
      CALL KIPPH2(MATR3,ACOL,NU,XBAR,XSTAR,BBB,PPP,
     *    QQQ,OMEGA,VBLTYP,KOWTYP,MATR1(DELQ),
     *    MATR1(DELX),MATR1(SCRCH1),MATR1(SCRCH2),
     *    MATR1(DELB),DELTA,DDD,RHS,MATR1(LIVX),MATR1(LIVQ),
     *    SSS,TTT)
C
C        ***SECTION TO CALL THE PHASE III SUBROUTINE***
C
C-----SET WORK VECTOR SUBSCRIPTS FOR PHASE III
      LDELXB=LROW1+MAXCOL
      IF(LDELXB/2*2.EQ.LDELXB)LDELXB=LDELXB+1
      LXBSTR=LDELXB+PRECN*MAXCOL
      MOVV=LXBSTR+PRECN*MAXCOL
      MOVUST=MOVV+PRECN*MAXCOL
      MOVUBR=MOVUST+PRECN*MMM
      MOVWP=MOVUBR+PRECN*MMM
      WWWP=MOVWP+PRECN*NNNP
      MOVWQ=WWWP+PRECN*NNNP
      WWWQ=MOVWQ+PRECN*NNNQ
      SCRCH=WWWQ+PRECN*NNNQ
      KBAR=MOVV
      KSTAR=LXBSTR
C
C-----CALL THE SUBROUTINE
      CALL KIPPH3(MATR3,ACOL,ACUT,BBB,PPP,QQQ,OMEGA,
     *    ROWTYP,MATR1(FROW1),MATR1(LROW1),CCC,UBAR
     *    ,USTAR,VSTAR,MATR1(LDELXB),
     *    MATR1(LXBSTR),MATR1(MOVV),MATR1(MOVUST),
     *    MATR1(MOVUBR),MATR1(MOVWP),MATR1(WWWP),
     *    MATR1(MOVWQ),MATR1(WWWQ),MATR1(SCRCH),
     *    MATR1(KBAR),MATR1(KSTAR),DELTA,XBAR,XSTAR,RHS)
C
C
C        ***SECTION TO CALL THE PHASE IV SUBROUTINE***
C
C-----SETUP WORK AREAS
```

```
C
C-----CORRECT PROBLEM DATA FOR DENOMINATORS AND NU
      MULTXB=DENBAR/DELTM
      DO 43[illegible] JJJ=1,NNN
      DO 42[illegible] PSN=1,MMM
      JJJB=BBB(PSN)
      IF(JJJB.NE.JJJ)GO TO 42[illegible]
C-----XBAR,XSTAR,DDD IN XXXB
      XBAR(JJJ)=XBAR(JJJ)*MULTXB/NU(JJJ)
      XSTAR(JJJ)=XSTAR(JJJ)/DELTM*DENSTR/NU(JJJ)
      DDD(JJJ)=DDD(JJJ)/DELTM/NU(JJJ)
      GO TO 43[illegible]
 42[illegible] CONTINUE
C-----XBAR,XSTAR,DDD NOT IN XXXB
      XBAR(JJJ)=XBAR(JJJ)*DENBAR/NU(JJJ)
      XSTAR(JJJ)=XSTAR(JJJ)*DENSTR/NU(JJJ)
      DDD(JJJ)=DDD(JJJ)/NU(JJJ)
C-----CCC,AAA,ASTAR AND CHECK TOTALS
 43[illegible] IF(NU(JJJ).GT.1.AND.DATTYP.EQ.0)CCC(JJJ)=CCC(JJJ)
     *    *NU(JJJ)
      IF(NU(JJJ).GT.1.AND.DATTYP.EQ.1.AND.XBAR(JJJ).EQ.0)
     *    CCC(JJJ)=(CCC(JJJ)-.01)*NU(JJJ)+.01
      IF(NU(JJJ).GT.1.AND.DATTYP.EQ.1.AND.XBAR(JJJ).NE.0)
     *    CCC(JJJ)=(CCC(JJJ)+.01)*NU(JJJ)-.001
      MIN=1
      IF(JJJ.GT.1)MIN=ACOL(JJJ-1)+1
      MAX=ACOL(JJJ)
      DO 432[illegible] PSN=MIN,MAX
      MATR2(2,PSN)=MATR2(2,PSN)*NU(JJJ)
      PROD=JJJ*MATR2(1,PSN)*MATR2(2,PSN)
      CHKAAA=MOD(CHKAAA+PROD,32767)
 432[illegible] CONTINUE
      ICCC=CCC(JJJ)+.5
      CHKCCC=MOD(CHKCCC+ICCC,32767)
      CHKDDD=MOD(CHKDDD+DDD(JJJ)*JJJ,32767)
      CHKXST=MOD(CHKXST+XSTAR(JJJ)*JJJ,32767)
      CHKXBR=MOD(CHKXBR+XBAR(JJJ)*JJJ,32767)
      CHKVTY=MOD(CHKVTY+VBLTYP(JJJ)*JJJ,32767)
      MIN=ACOL(NNN)+1
      IF(JJJ.GT.1)MIN=ACOL(JJJ-1)+1
      MAX=ACOL(JJJ)
      IF(MIN.GT.MAX)GO TO 4500
      DO 435[illegible] PSN=MIN,MAX
      INTGNO=MATR2(2,PSN)
      ZRELNO=ZRELNO*NU(JJJ)
      MATR2(2,PSN)=INTGNO
 435[illegible] CONTINUE
 450[illegible] CONTINUE
C
C-----CALL THE VERIFICATION ROUTINE IF SPECIFIED
      XCRCH=LROW1+MAXOUT
      IF(XCRCH/2*2.EQ.XCRCH)XCRCH=XCRCH+1
      IBIAS=XCRCH+PRECN*MNN
C
C-----CALL THE SUBROUTINE
      CALL RIPPH4(MATR3,SSS,TTT,BBB,VSTAR,USTAR,UBAR,
     *    CCC,RHS,MATR1(LROW1),MATR1(FROW1),
     *    ACOL,ACUT,OMEGA,XBAR,XSTAR,DCD,ROWTYP,VBLTYP,
     *      MATR1(XCRCH),
     *    MATR1(IBIAS))
C
C
C         ***SECTION TO COMPLETE PROBLEM GENERATION***
C
C-----CALL THE MEMORY REORGANIZATION SUBROUTINE
      CALL RIPORG(MATR3,MATR2,ACOL,ACUT)
C
C-----COMPUTE STAR DENOMINATOR AS LCM OF THE NU(JJJ)
      DENSTR=NU(1)
      DO 413[illegible] JJJ=1,NNN
      PROD=DENSTR*NU(JJJ)
      DUMMY=RIPEUC(PROD,DENSTR,NU(JJJ),DUMMY2,GCD)
      DENSTR=PROD/GCD
 413[illegible] CONTINUE
C-----COMPUTE BAR DENOMINATOR AS LCM OF DENSTR AND DELTN
      PROD=DENSTR*DELTM
      DUMMY=RIPEUC(PROD,DENSTR,DELTM,DUMMY2,GCD)
      DENBAR=PROD/GCD
C
C-----PRINT DENBAR,DENSTR AND NU
      IF(LVLPRT.LT.4)GO TO 4150
 412[illegible] FORMAT(/11H ***DENBAR=,I2[illegible],8H/DENSTR=,I20,3H***)
 413[illegible] FORMAT(/1X,9HVECTOR NU /(5X,10I5))
      WRITE(LUPR,4120)DENBAR,DENSTR
      WRITE(LUPR,4130)(NU(JJJ),JJJ=1,NNN)
C
C-----CLEAR CHECK TOTALS
 415[illegible] CHKAAA=0
      CHKCCC=0
      CHKDDD=0
      CHKXST=0
      CHKXBR=MOD(DENBAR,32767)
      CHKRHS=MOD(DENSTR,32767)
      CHKVTY=0
      CHKRTY=0
      DO 417[illegible] III=1,MMM
      IRHS=RHS(III)+.5
      CHKRHS=MOD(CHKRHS+IRHS*III,32767)
      CHKRTY=MOD(CHKRTY+ROWTYP(III)*III,32767)
 417[illegible] CONTINUE
```

```
      IF(VERIFY.EQ. )GO TO 5...
C
      CALL RIPVER(MATR2,MATR2,RHS,CCC,DDD,UBAR,USTAR,XBAR,
     *    XSTAR,ROWTYP,VBLTYP,VSTAR,ACOL,ACUT,EBB,NU,OMEGA,
     *    DENBAR,DENSTR)
C
C-----RANDOMIZE ROW NUMBERS
 5000 CALL RIPSEQ(OMEGA,MMM,1,MMM)
      LAST=ACOL(NNN)
C-----BYPASS IF LVLPRT .GE. 5
      IF(LVLPRT.GE.5)GO TO 6001
C-----SAVE NECESSARY VALUES AND COMPLETE REARRANGEMENT
      DO 5200 PSN=1,LAST
      III=MATR2(1,PSN)
      MATR2(1,PSN)=OMEGA(III)
 5200 CONTINUE
      DO 5300 III=1,MMM
      MATR1(CORTOP+III)=RHS(III)+.5
      MATR1(CORTOP+MMM+III)=ROWTYP(III)
 5300 CONTINUE
      DO 5350 III=1,MMM
      INEW=OMEGA(III)
      RHS(INEW)=MATR1(CORTOP+III)
      ROWTYP(INEW)=MATR1(CORTOP+MMM+III)
 5350 CONTINUE
C
C         ***SECTION TO PRINT RESULTS***
C
C-----PRINT PARAMETER SUMMARY
 6000 IF(LVLPRT.LE.0)GO TO 9000
      TRUDEN=ACOL(NNN)*100./(MMM*NNN)
      RWORK=SOLSTR
      IF(RWORK.LT.0)RWORK=-RWORK
      TRURAT=.5+(SOLSTR-SOLBAR)*1000/RWORK
      NUMINT=PARAM(26)+PARAM(27)
 6100 FORMAT(1H1/20X,5(5H*****)
     *    /20X,25H*  RIP PROBLEM SUMMARY  *
     *    ,20X,12HRIP VERSION=,I3,1H-,I1
     *    /20X,5(5H*****)
     *    //14H PROBLEM NAME=,10A4
     *    /18H NUMBER VARIABLES=,I5
     *    /21H NUMBER CONSTRAINTS=,I5
     *    /26H NUMBER INTEGER VARIABLES=,I5
     *    /34H LINEAR RELAXATION SOLUTION VALUE=,E15.8
     *    /24H INTEGER SOLUTION VALUE=,E15.8
     *    /41H DENSITY OF NONZERO COEFFICIENTS DESIRED=,I8
     *    /41H ACTUAL DENSITY OF NONZERO COEFFICIENTS=,I8
     *    /45H SOLUTION GAP PERCENT OF IP SOLUTION DESIRED=,
     *       I8
     *    /44H ACTUAL SOLUTION GAP PERCENT OF IP SOLUTION=,
     *       I8)
C
      WRITE(LUPR,6100)VERSN,PRECN,(PARAM(KKK),KKK=1,11),NNN,
     *    MMM,
     *    NUMINT,SOLBAR,SOLSTR,DENSA,TRUDEN,NU,TRURAT
C
 6150 FORMAT(//23H INPUT PARAMETER LIST--/
     *    (2X,4(1HP,I3,1H=,I10)))
      MAX=65+MMM
      WRITE(LUPR,6150)(KKK,PARAM(KKK),KKK=11,MAX)
C
 6170 FORMAT(//27H ENDING RANDOM SEED PAIRS-- /
     *    (2X,I5,1H=,2I10))
      WRITE(LUPR,6170)(KKK,SEED(1,KKK),SEED(2,KKK),
     *    KKK=1,7)
C
 6210 FORMAT(//20H DATA CHECK TOTALS--/
     *    5X,4HAAA=,I12 /5X,4HRHS=,I12/5X,4HCCC=,I12/
     *    5X,4HDDD=,I12 /5X,5HXBAR=,I11/5X,6HXSTAR=,I10/
     *    5X,7HROWTYP=,I9/5X,7HVBLTYP=,I9)
      WRITE(LUPR,6210)CHKAAA,CHKRHS,CHKCCC,CHKDDD,
     *    CHKXBR,CHKXST,CHKRTY,CHKVTY
C
      IF(LVLPRT-2)9000,6500,6800
C-----PRINT SOLUTION VECTORS
 6300 FORMAT(//9H VARIABLE,6X,7HXBAR(J),12X,
     *    8HXSTAR(J)/1X,7(6H------),1H-)
 6310 FORMAT(I10,E15.8,5X,E15.8)
C
 6500 WRITE(LUPR,6300)
      DO 6550 JJJ=1,NNN
      RELXB=XBAR(JJJ)
      RELXB=RELXB/DENBAR
      RELXS=XSTAR(JJJ)
      RELXS=RELXS/DENSTR
      WRITE(LUPR,6310)JJJ,RELXB,RELXS
 6550 CONTINUE
      GO TO 9000
C
C-----PRINT FULL PROBLEM
 6800 WRITE(LUPR,6810)
 6810 FORMAT(1H1/20X,1H*,5(5H*****)
     *    /20X,2H* ,22HGENERATED PROBLEM DATA,2H *
     *    /20X,1H*,5(5H*****)
     *    //14H COLUMN DATA--)
C
      DO 6900 JJJ=1,NNN
      RELXB=XBAR(JJJ)
      RELXB=RELXB/DENBAR
      RELXS=XSTAR(JJJ)
      RELXS=RELXS/DENSTR
```

```
      MIN=1
      IF(JJJ.GT.1)MIN=ACOL(JJJ-1)+1
      MAX=ACOL(JJJ)
 6030 FORMAT(/9H COL=,I5,6H,XBAR=,E15.8,
     *    7H,XSTAR=,E15.8,3H,C=,E15.8,3H,D=,I10
     *    ,6H,TYPE=,I3
     *    /(5X,3(1H/,I10,7H IN ROW,I4)))
C
      WRITE(LUPR,6030)JJJ,RELXB,RELXS,
     *    CCC(JJJ),DDD(JJJ),VBLTYP(JJJ),(MATR2(2,KKK),
     *       MATR2(1,KKK),
     *    KKK=MIN,MAX)
C
 6900 CONTINUE
C
 6920 FORMAT(//19H RIGHT-HAND-SIDES--
     *    /(5X,3(1H/,F15.8,7H IN ROW,I4)))
      WRITE(LUPR,6920)(RHS(III),III,III=1,MMM)
C
 6950 FORMAT(//12H ROW TYPES--
     *    /(5X,4(1H/,I3,7H ON ROW,I4)))
      WRITE(LUPR,6950)(ROWTYP(III),III,III=1,MMM)
C
      IF(LVLPRT.LT.4)GO TO 9000
C-----PRINT ROW REARRANGEMENT MAPPING
 7010 FORMAT(//27H ROW REARRANGEMENT MAPPING
     *    ,29H(NOT USED IF LVLPRT .GE. 5)--
     *    /(5X,4(1H/,I3,7H BECAME,I4)))
      WRITE(LUPR,7010)(III,OMEGA(III),III=1,MMM)
C
C        ***EXIT SECTION***
C
 9000 CONTINUE
      RETURN
      END
C
C-------------------------------------------------------------
C
C-----SUBROUTINE RIPPH1(MATRIX,XBAR,XSTAR,ACOL1,NU,DELTA,
C    BBB,SCRCH1,
C     *    SCRCH2,SCRCH3,ETA,TAU,OMEGA,ROWTYP,VBLTYP,FROW1,
C        LROW1,
C     *    RROW1,RINVR1,GGG,DDD,SCRCH4,XCRCH,ZATRIX,RHS)
C-----SUBROUTINE TO EXECUTE PHASE I OF THE RIF RANDOM
C    INTEGER PROGRAM
C
C      GENERATOR.  PARAMETERS NOT DEFINED IN RIPPH OR THE
C         CORRESPONDING
C      RANDIN AND LIN PAPER ARE AS FOLLOWS--
C         MATRIX=THE INTEGER VERSION OF THE GENERAL RIP
C            3-TUPLE (LINKED) MATRIX STORAGE AREA
C         SCRCH1,2,3 AND 4=WORK VECTORS USED FOR NUMEROUS
C            TEMPORARY
C            QUANTITIES
C         RROW1=THE VECTOR OF MATRIX 3-TUPLE STARTING POINTS
C            FOR ROWS
C            OF THE MATRIX RRR (THIS VECTOR IS EQUIVALENCED
C               THROUGH THE
C            CALL TO TAU)
C         RINVR1=THE VECTOR OF MATRIX 3-TUPLE STARTING
C            POINTS FOR ROWS
C            OF THE MATRIX R-INVERSE (THIS VECTOR IS
C               EQUIVALENCED
C            THROUGH THE CALL TO ETA)
C         GGG=AN INTERMEDIATE VECTOR DEFINED AS
C            L-INVERSE*(XBAR-XSTAR)
C         XCRCH=A DOUBLE PRECISION VALUED SCRATCH VECTOR USED
C            FOR TEMPORARY QUANTITIES (THIS VECTOR IS
C               EQUIVALENCED
C            THROUGH THE CALL WITH SCRCH1)
C         ZATRIX=A FLOATING POINT VALUED VERSION OF THE 3-
C            TUPLE
C            MATRIX AREA (THIS VECTOR IS EQUIVALENCED THROUGH
C            THE CALL WITH MATRIX)
C
      SUBROUTINE RIPPH1(MATRIX,XBAR,XSTAR,ACOL1,NU,DELTA,
     *    BBB,SCRCH1,
     *    SCRCH2,SCRCH3,ETA,TAU,OMEGA,ROWTYP,VBLTYP,FROW1,
     *       LROW1,
     *    RROW1,RINVR1,GGG,DDD,SCRCH4,XCRCH,ZATRIX,RHS)
C
      IMPLICIT INTEGER (A-Y)
C
      COMMON /RIPCOM/SOLBAR,SOLSTR,TOLMJV,TOLCFL,TOLOPT,
     *    TOLSTP,TOLVER,BCONT,BDCONT,BDINT,BDSLK,BFRAC,
     *    BINT,BSLK,BTIST,CORBOT,CORTOP,DATTYP,DELTM,
     *    DENSA,DENSF,DENSP,DENSPQ,DENSQ,DETERM,
     *    DIFDEG,DIMPRM,DISRAT,ECONT,EFRAC,EINT,ESLK,
     *    FRAC1,INFIN,LIMA,LIMALP,LIMD,LIMF,LIMR,LIMSLK,
     *    LIMVAN,LIMVMX,LIMWMN,LIMWMX,LUPR,LVLPRT,
     *    MAXCON,MAXCUT,MAXINT,MAXJ1,MMM,MU,
     *    NNN,NNNP,NNNQ,NNNS,NNNT,NUMERR,NXTMAT,PRECN,PSLK,
     *    RTPSLK,SEED(2,7),SSLK,TOLMU,VERIFY,VERSN
C
      DIMENSION MATRIX(3,1),XBAR(1),XSTAR(1),ACOL1(1),NU(1),
     *    DELTA(1),
```

```
     *   BBB(1),SCRCH1(1),SCRCH2(1),SCRCH3(1),ETA(1),TAU(1),
     *      DDD(1)
     *  ,OMEGA(1),OMTYP(1),VBLTYP(1),FROW1(1),LROW1(1),
     *  RROW1(1),RINV(1),GGG(1),SCRCH(1),XCRCH(1),
     *      ZAMIX(3,1)
     *    ,RHS(1)
C
      DOUBLE PRECISION SOLBAR,SOLSTR,TOLMOV,TOLOFL,TOLOPT,
     *   TOLSTP,TOLVEC
      DOUBLE PRECISION RWORK,XCRCH,FTOL,RHS,RIPROT
C
      DATA LTGINV,LTRRR,LTKINV,LTBBB,LTLLL,LTFFF,LTFTIL,
     *   LTBBB
     *   /4HGINV,3HRRR,4HKINV,3HBBB,3HLLL,3HFFF,4HFTIL,
     *      3HBBB/
C
C        ***SECTION TO DETERMINE BASIC PART OF XBAR
C           SOLUTION***
C
C-----FIND THE FIRST NON-UNIT DELTA
      DO 250 III=1,MMM
      IF(DELTA(III).GT.1)GO TO 260
  250 CONTINUE
C-----LOOP TO SET TAU, DDD AND INTEGER PART OF XBAR
C     NUMERATORS
  260 TAU1=III
      DO 2200 III=1,MMM
C-----SET TAU
  290 TAU(III)=DELTA(III)
      IF(III.LT.TAU1)TAU(III)=DELTA(TAU1)
C-----FIND VARIABLE INDEX AND VARIABLE TYPE
      JJJ=BBB(III)
      TYP=VBLTYP(JJJ)
C-----SPLIT BY TYPE TO SET DDD AND INTEGER ETA
      GO TO (2110,2110,2120,2130,2140),TYP
C-----SLACKS/SURPLUSES
 2110 IF(III.LE.BSLK)GO TO 2125
      MULT=RIPUNF(1,LIMSLK,2)
      ETA(III)=TAU(III)*MULT
      GO TO 2180
C-----CONTINUOUS VARIABLE
 2120 DDD(JJJ)=RIPUNF(2,LIMD,3)
      IF(III.LE.BSLK+NINT+BDCONT)GO TO 2150
 2125 MULT=RIPUNF(1,DDD(JJJ)-1,2)
      ETA(III)=TAU(III)*MULT
      GO TO 2180
C-----GENERAL INTEGER VARIABLE
 2130 DDD(JJJ)=RIPUNF(2,LIMD,3)
      IF(III.LE.BSLK+BDINT)GO TO 2150
      GO TO 2125
C-----ZERO-ONE INTEGER VARIABLE
 2140 DDD(JJJ)=1
      IF(III.LE.BSLK+BDINT)GO TO 2150
      GO TO 2160
C-----DEGENERATE CASES
 2150 IF(OMEGA(III))2157,2155,2160
 2155 IF(RIPUNF(1,1,2).EQ.1)GO TO 2160
 2157 ETA(III)=TAU(III)*DDD(JJJ)
      GO TO 2180
 2160 ETA(III)=0
C-----SET XBAR AND MULTIPLY DDD BY DELTA(MMM)
 2180 XBAR(JJJ)=ETA(III)*DELTA(MMM)/TAU(III)
      IF(TYP.GE.3)DDD(JJJ)=DDD(JJJ)*DELTA(MMM)
C-----END OF III LOOP
 2200 CONTINUE
C
C-----PREPARE A LIST OF PRIMES NOT DIVIDING DELTA(MMM)
C     TO GENERATE FRACTIONAL PARTS RELATIVELY PRIME TO THE
C        TAU'S
      MAXPRM=0
      DO 2400 PRM=1,DIMPRM
      TEST=RIPPRM(PRM)
      QUOT=DELTA(MMM)/TEST
      IF(QUOT*TEST.EQ.DELTA(MMM))GO TO 2450
      MAXPRM=MAXPRM+1
      SCRCH1(MAXPRM)=TEST
 2400 CONTINUE
C
C-----LOOP THROUGH FRACTIONAL VARIABLES TO INCREASE ETAS
C     FOR FRACTIONAL PARTS
      DO 2600 III=FRAC1,MMM
      JJJ=BBB(III)
      PROD=1
      IF(MAXPRM.EQ.0)GO TO 2590
      LIM=RIPUNF(1,TAU(III)-1,2)
      PSN=0
C-----NEXT PRIME
 2520 PSN=PSN+1
      PRM=SCRCH1(PSN)
C-----DECIDE WHETHER TO USE IT
 2540 IF(PROD*PRM.GT.LIM)GO TO 2590
      IF(LIM*RIPUNF(3,1000,2)/PROD.LE.PRM*1000)GO TO 2520
      PROD=PROD*PRM
      GO TO 2540
C-----RESET ETA AND XBAR FOR FRACTIONAL PART
 2590 ETA(III)=ETA(III)+PROD
      XBAR(JJJ)=ETA(III)*DELTA(MMM)/TAU(III)
 2600 CONTINUE
      IF(LVLPRT.LT.4)GO TO 3000
C-----PRINT RUNTYP,OMEGA,ETA,TAU,DELTA
```

```
 2500 FORMAT(/4H ROW,2X,6H  TYPE,6H DELSA,5X,5HDELTA,7X,3HTAU,
     *   5X,5HD TAU/(1X,I4,2I6,3E10.3))
      WRITE(LUPR,2600)(III,ROWTYP(III),OMEGA(III),ETA(III),
     *    TAU(III),
     *    DELTA(III),III=1,MMM)
C
C         ***MAIN SECTION TO GENERATE C-INVERSE (IN THE BBB
C            MATRIX AREA)*
C
C-----LOOP THROUGH BASIC COLUMNS
 3000 MAXIII=MMM-1
      DO 3500 III=1,MAXIII
      JJJ=BBB(III)
      TYP=VBLTYP(JJJ)
      GO TO (3100,3100,3200,3300,3300),TYP
C-----SLACK/SURPLUS VARIABLES
 3100 COEF=ROWTYP(III)
      CALL RIPADD(MATRIX,ACOL1(JJJ),III,COEF)
C-----PLACE ONE COEFFICIENT TO THE RIGHT
      INEW=RIPUNF(ESLK+1,MAXIII,4)
      INEW2=RIPUNF(ESLK+1,MAXIII,4)
C-----GENERATE AND STORE AN ALPHA COEFFICIENT
 3150 JNEW=BBB(INEW)
      MULT=1
      IF(INEW.GE.FRAC1.AND.TAU(INEW).GT.DELTA(III))
     *   MULT=TAU(INEW)/DELTA(III)
      LIM=LIMALP/MULT
      LIM=MAX0(LIM,1)
      ALPHA=RIPUNF(1,LIM,4)*MULT
      CALL RIPADD(MATRIX,ACOL1(JNEW),III,ALPHA)
C-----ATTEMPT TO PLACE A SECOND COEFFICIENT IN SLACK ROWS
C    WHEN
C      MORE DENSITY IS REQUIRED
      IF(TYP.GT.2) GO TO 3500
      IF(DENSA.LE.2.0.OR.INEW.EQ.INEW2)GO TO 3500
      JNEW=BBB(INEW2)
      MULT=1
      IF(INEW2.GE.FRAC1.AND.TAU(INEW2).GT.DELTA(III))
     *   MULT=TAU(INEW2)/DELTA(III)
      LIM=LIMALP/MULT
      LIM=MAX0(LIM,1)
      ALPHA=RIPUNF(1,LIM,4)*MULT
      CALL RIPADD(MATRIX,ACOL1(JNEW),III,ALPHA)
      GO TO 3500
C-----CONTINUOUS VARIABLES
 3200 CALL RIPADD(MATRIX,ACOL1(JJJ),III,1)
C-----SEPARATE OUT FRACTIONAL ROWS
      IF(III.LT.FRAC1)GO TO 3250
      INEW=RIPUNF(III+1,MAXIII,4)
      GO TO 3230
C-----GENERATE AND STORE AN ALPHA IN THE FRACTIONAL AREA
 3220 IF(TAU(III).LE.DELTA(III))GO TO 3500
      MAX=MMM-III
      CALL RIPSEQ(SCRCH1,MAX,4,MMM)
      DO 3225 PSN=1,MAX
      INEW=SCRCH1(PSN)
      IF(OMEGA(INEW).EQ.OMEGA(III))GO TO 3230
 3225 CONTINUE
 3230 JNEW=BBB(INEW)
      COEF=ETA(III)*DELTA(III)*TAU(INEW)/TAU(III)
      COEF1=-ETA(INEW)*DELTA(III)
      ALPHA=RIPSOC(COEF,COEF1,TAU(INEW),GAMMAI,GCD)
      CALL RIPADD(MATRIX,ACOL1(JNEW),III,ALPHA)
      GO TO 3500
C-----INTEGER VALUED CONTINUOUS VARIABLE
 3250 INEW=RIPUNF(ESLK+1,MAXIII,4)
      IF(INEW-III)3150,3250,3150
C-----INTEGER VARIABLE
 3300 CALL RIPADD(MATRIX,ACOL1(JJJ),III,1)
C-----SEPARATE OUT FRACTIONAL ONES
      IF(III.GE.FRAC1)GO TO 3220
C-----END MAIN ROW LOOP
 3500 CONTINUE
C-----SET LAST ROW OF C-INVERSE
      JJJ=BBB(MMM)
      CALL RIPADD(MATRIX,ACOL1(JJJ),MMM,1)
C
C         ***SECTION TO GENERATE XSTAR SOLUTION FOR BASIC
C            VARIABLES***
C
C-----RANDOMIZE ORDER OF CHANGE DIRECTION ALTERNATION
      CALL RIPSEQ(SCRCH1,MMM,3,MMM)
      DIRECT=-1
      SLKPSN=0
C-----LOOP THROUGH ROWS
      DO 3900 PSN=1,MMM
      III=SCRCH1(PSN)
      JJJ=BBB(III)
      TYP=VBLTYP(JJJ)
      PTOCHN=XBAR(JJJ)
      IF(TYP.GE.3)GO TO 3700
C-----SLACK/SURPLUS VARIABLE
      IF(SLKPSN.GE.BTITST)GO TO 3690
C-----MAKE ROW TIGHT IN STAR SOLUTION
      SLKPSN=SLKPSN+1
      XSTAR(JJJ)=0
      GO TO 3900
C-----PROVIDE SLACK IN THE STAR SOLUTION
 3690 PTUP=LIMSLK*DELTA(MMM)-XBAR(JJJ)
      GO TO 3810
```

```
      IF(ROW.LE.ESLK.OR.ROW.GT.ESLK+EINT.AND.ROW.LE.MMM-
     *   EFRAC)
     *   GO TO 4050
      IF(ROW.GT.EFRAC1)ROW=ROW-ECONT
      ROW=ROW-ESLK
      GGG(ROW)=GGG(ROW)+DIFF*MATRIX(2,PSN)
 4050 PSN=MATRIX(3,PSN)
      GO TO 4040
 4090 CONTINUE
C
C-----REVISE INTEGER AREAS OF C-INVERSE TO REDUCE GGG
C     AND INCREASE BBB DENSITY
C
      DO 5000 KKK=1,MAXKKK
      III=KKK+ESLK
      IF(KKK.GT.EINT)III=III+ECONT
      JJJ=BBB(III)
      IF(KKK.GE.MAXKKK-1)GO TO 4990
      IF(IABS(GGG(KKK)).LE.DELTA(MMM)/3)GO TO 4990
C-----CHECK COLUMNS OF OPPOSITE ORIENTATION TO REDUCE GGG
      MIN=KKK+1
      MAX=MAXKKK
      DO 4980 KNEW=MIN,MAX,2
      INEW=KNEW+ESLK
      IF(INEW.GT.EINT)INEW=INEW+ECONT
      IF(OMEGA(INEW).EQ.OMEGA(III))GO TO 4980
      JNEW=BBB(INEW)
      DIFF=XBAR(JNEW)-XSTAR(JNEW)
      IF(DIFF.EQ.0)GO TO 4980
      MULT=1
      IF(INEW.GE.EFRAC1)MULT=TAU(INEW)/DELTA(III)
      ALPHA=-GGG(KKK)/(MULT*DIFF)
      ALPHA=ALPHA*MULT
      IF(ALPHA.EQ.0)GO TO 4980
      IF(IABS(GGG(KKK)).LT.DELTA(MMM).AND.GGG(KKK)+
     *   ALPHA*DIFF.EQ.0)ALPHA=ALPHA-MULT
      IF(ALPHA.EQ.0)GO TO 4980
      CALL RIPADD(MATRIX,ACOL1(JNEW),III,ALPHA)
      GGG(KKK)=GGG(KKK)+ALPHA*DIFF
 4980 CONTINUE
C-----ORIENT GGG
 4990 GGG(KKK)=GGG(KKK)*OMEGA(III)
 5000 CONTINUE
C
C-----MAIN LOOP TO GENERATE ROWS OF F-TILDA AND CUTS
      DO 9710 CUT=1,MAXCUT
C-----INITIALIZE LIST POINTER
      FROW1(CUT)=0
C-----SET SPECIAL PROCESSING OF 1ST EFRAC CUTS
      IF(CUT.GT.1)GO TO 5190
C-----ALL OTHER TYPES
 3710 PTUP=UBD(JJJ)-XBAR(JJJ)
C-----CHECK IF ONLY DOWN IS FEASIBLE
      IF(PTUP.GE.1)GO TO 3810
C-----SET XSTAR DOWN FROM XBAR
 3760 DST=XPOST(PTDOWN,3)
      IF(III.GT.ESLK.AND.III.LE.ESLK+EINT)
     *   DST=DELTA(MMM)
      IF(III.GT.MMM-EFRAC)
     *   DST=MOD(XBAR(JJJ),DELTA(MMM))
      TRUEX=(2*(XBAR(JJJ)-DST)+DELTA(MMM))/(2*DELTA(MMM))
      GO TO 3880
C-----CHECK IF ONLY UP IS FEASIBLE
 3800 IF(PTDOWN.GE.1)GO TO 3830
C-----SET XSTAR UP FROM XBAR
 3810 DST=XPOST(PTUP,3)
      IF(III.GT.ESLK.AND.III.LE.ESLK+EINT)
     *   DST=DELTA(MMM)
      IF(III.GT.MMM-EFRAC)
     *   DST=DELTA(MMM)-MOD(XBAR(JJJ),DELTA(MMM))
      TRUEX=(2*(XBAR(JJJ)+DST)+DELTA(MMM))/(2*DELTA(MMM))
      GO TO 3880
C-----BOTH DIRECTIONS FEASIBLE--SET IF INTEGER AREA
 3830 IF(OMEGA(III))3760,3850,3810
C-----BOTH DIRECTIONS POSSIBLE--ALTERNATE DIRECTIONS
 3850 DIRECT=-DIRECT
      IF(DIRECT)3760,3760,3810
C-----MULTIPLY XSTAR BY DELTA(MMM)
 3880 XSTAR(JJJ)=DELTA(MMM)*TRUEX
      XSTAR(JJJ)=MAX0(XSTAR(JJJ),0)
      IF(TYP.GE.3)XSTAR(JJJ)=MIN0(XSTAR(JJJ),UBD(JJJ))
 3900 CONTINUE
C
C          ***SECTION TO GENERATE F-TILDA AND LLL MATRICES***
C
C-----COMPUTE GGG=C-INVERSE*(XBAR-XSTAR) --INTEGER ROWS ONLY
      MAXKKK=EFRAC+EINT
      DO 4500 KKK=1,MAXKKK
      GGG(KKK)=0
 4500 CONTINUE
      DO 4700 KKK=1,MAXKKK
C-----DETERMINE JJJ INDEX
      III=KKK+ESLK
      IF(KKK.GT.EINT)III=III+ECONT
      JJJ=BBB(III)
      DIFF=XBAR(JJJ)-XSTAR(JJJ)
C-----ADD UP A COLUMN OF C-INVERSE TIMES DIFF
      PSN=ACOL1(JJJ)
 4600 IF(PSN.LE.0)GO TO 4700
      ROW=MATRIX(1,PSN)
```

```
C-----END CUT LOOP
 5700 CONTINUE
C
C        ***SECTION TO GENERATE RRR ***
C
C-----CLEAR ROW POINTERS
      DO 6050 III=1,MMM
      RROW1(III)=0
 6050 CONTINUE
C
      NEGROV=0
      POSROV=0
C-----LIST POTENTIAL NEGATIVE AND POSITIVE RECEIVER ROWS
      MINIII=ESLK+1
      DO 6066 III=MINIII,MMM
C-----SET COEFFICIENT LIMITS
      IF(III.GT.ESLK+EINT.AND.III.LE.MMM-EFRAC)GO TO 6510
      KKK=III
      IF(III.GE.FRAC1)KKK=KKK-ECONT
      KKK=KKK-ESLK
      RWORK=DELTA(III)
      SCRCH4(III)=GGG(KKK)*RWORK/DELTA(MMM)
C-----CLEAR RECEIVER TAG AREA
 6510 SCRCH3(III)=0
C-----SPLIT CASES BY OMEGA
      IF(OMEGA(III))6540,6530,6520
 6520 POSROV=POSROV+1
      SCRCH2(POSROV)=III
      GO TO 6066
 6530 NEGROV=NEGROV+1
      SCRCH1(NEGROV)=III
      GO TO 6520
 6540 NEGROV=NEGROV+1
      SCRCH1(NEGROV)=III
 6066 CONTINUE
C-----TAG LAST CONTINUOUS ROW TO NOT BE A SENDER
      SCRCH3(MMM-EFRAC)=1
      MINNEG=1
      MINPOS=1
C
C-----MAIN LOOP TO GENERATE ROWS OF RRR
      CALL RIPADD(MATRIX,RROW1(MMM),MMM,1)
      MAXIII=MMM-1
      DO 6900 III=1,MAXIII
C-----SET DIAGONAL TO +1
      CALL RIPADD(MATRIX,RROW1(III),III,1)
C-----BYPASS FURTHER GENERATION IF ALREADY A RECEIVER OR
C   SLACK
      IF(SCRCH3(III).EQ.1.OR.III.LE.ESLK)GO TO 6966
C-----SET COEFFICIENT
      DO 5160 KKK=1,MAXKKK
      SCRCH2(KKK)=MAXKKK-KKK+1
 5160 CONTINUE
      GO TO 5210
 5190 IF(COUT.GT.EFRAC)GO TO 5200
      SCRCH2(COUT)=SCRCH2(1)
      SCRCH2(1)=MAXKKK-COUT+1
      GO TO 5210
C-----RANDOMIZE COEFFICIENT ASSIGNMENT ORDER
 5200 CALL RIPSEQ(SCRCH2,MAXKKK,4,0)
C-----LOOP THROUGH TO CALCULATE (ORIENTED) F-TILDA ROW
 5210 REMAIN=DELTA(MMM)
      DO 5300 PSN=1,MAXKKK
      KKK=SCRCH2(PSN)
      IF(GGG(KKK).LT.0)GO TO 5250
C-----ZERO GGG VALUE
      SCRCH1(KKK)=RIPUNF(1,EINF,4)
      GO TO 5300
C-----NEGATIVE GGG
 5250 SCRCH1(KKK)=0
      IF(-GGG(KKK).GE.REMAIN)GO TO 5300
      SCRCH1(KKK)=REMAIN/(-GGG(KKK))
      REMAIN=REMAIN+SCRCH1(KKK)*GGG(KKK)
      IF(REMAIN.GT.0)GO TO 5300
      SCRCH1(KKK)=SCRCH1(KKK)-1
      REMAIN=REMAIN-GGG(KKK)
 5300 CONTINUE
      TOT=0
      DO 5400 KKK=1,MAXKKK
      TOT=TOT+GGG(KKK)*SCRCH1(KKK)
 5400 CONTINUE
C-----COMPRESS RESULTING ROW INTO STORAGE
      DO 5500 KKK=1,MAXKKK
      IF(SCRCH1(KKK).EQ.0)GO TO 5500
      III=KKK+ESLK
      IF(KKK.GT.EINT)III=III+ECONT
      CALL RIPADD(MATRIX,FROW1(COUT),III,SCRCH1(KKK)
     *   *OMEGA(III))
 5500 CONTINUE
C
C-----MULTIPLY F-TILDA ROW BY C-INVERSE TO OBTAIN LLL ROW
      LROW1(COUT)=0
      STRT=FROW1(COUT)
      DO 5600 III=1,MMM
      JJJ=BBB(III)
      IF(III.EQ.2)STRT=-1
      COEF=RIPOUT(MATRIX,STRT,ACOL1(JJJ),SCRCH4,MMM)
      IF(COEF.EQ.0)GO TO 5600
      CALL RIPADD(MATRIX,LROW1(COUT),III,COEF)
 5600 CONTINUE
```

```
      CALL RIPADD(MATRIX,RINVR1(1),1,1)
C-----MAIN ROW LOOP
      DO 7400 III=2,MMM
C-----CLEAR RESULT AREA
      SCRCH1(III)=0
      MAXKKK=III-1
      DO 7220 KKK=1,MAXKKK
      SCRCH1(KKK)=0
 7220 CONTINUE
C-----LOOP THROUGH ELEMENTS OF RRR ROW
      NOWK=RRCW1(III)
 7300 KKK=MATRIX(1,NOWK)
      IF(KKK.EQ.III)GO TO 7350
      COEF=MATRIX(2,NOWK)
C-----LOOP THROUGH CORRESPONDING R-INVERSE ROW
      NOWR=RINVR1(KKK)
 7330 PSN=MATRIX(1,NOWR)
      SCRCH1(PSN)=SCRCH1(PSN)-COEF*MATRIX(2,NOWR)
      NOWR=MATRIX(3,NOWR)
      IF(NOWR.GT.0)GO TO 7330
C-----END KKK LOOP
 7350 NOWK=MATRIX(3,NOWK)
      IF(NOWK.GT.0)GO TO 7300
C-----PACK THE CREATED ROW
      RINVR1(III)=0
      DO 7380 KKK=1,III
      IF(SCRCH1(KKK).EQ.0)GO TO 7380
      CALL RIPADD(MATRIX,RINVR1(III),KKK,SCRCH1(KKK))
 7380 CONTINUE
C-----END III LOOP
 7400 CONTINUE
C
      IF(LVLPRT.LT.5)GO TO 7500
C-----PRINT GGG,C-INVERSE,F-TILDA,RRR AND R-INVERSE
 7420 FORMAT(/5H GGG=,5I10/(5X,5I10))
      MAXKKK=EFRAC+EINT
      WRITE(LUPR,7420)(GGG(KKK),KKK=1,MAXKKK)
      CALL RIPVPR(2,MATRIX,ACOL1,MMM,BBB,1,MMM,2,LTCINV)
      CALL RIPVPR(1,MATRIX,FROW1,MMM,SCRCH1,1,MAXCUT,1,
     *   LTFTIL)
      CALL RIPVPR(1,MATRIX,RRCW1,MMM,SCRCH1,1,MMM,1,LTRRR)
      CALL RIPVPR(1,MATRIX,RINVR1,MMM,SCRCH1,1,MMM,1,LTRINV)
C
C        ***SECTION TO COMPLETE BBB BY PREMULTIPLYING C-
C           INVERSE BY
C           THE D-MATRIX AND THEN R-INVERSE***
C
C-----LOOP THROUGH COLUMNS OF BBB
 7500 DO 7700 III=1,MMM
      JJJ=BBB(III)
      COEF=RIPUNF(-LIMR,-1,4)
C-----INTEGER VARIABLE ROW
 6710 IF(III.LE.MMM-EFRAC.AND.III.GT.ESLK+EINT)
     *   GO TO 6850
C-----INCREASE MINS IF NECESSARY
      IF(MINNEG.EQ.0)GO TO 6720
 6710 IF(SCRCH1(MINNEG).GT.III)GO TO 6720
      MINNEG=MINNEG+1
      IF(MINNEG.LE.NEGRCV)GO TO 6710
      MINNEG=0
 6720 IF(MINPOS.EQ.0)GO TO 6750
 6730 IF(SCRCH2(MINPOS).GT.III)GO TO 6750
      MINPOS=MINPOS+1
      IF(MINPOS.LE.POSRCV)GO TO 6730
      MINPOS=0
C-----SPLIT CASES BY OMEGA
 6750 IF(OMEGA(III))6770,6800,6760
 6760 IF(MINNEG.EQ.0)GO TO 6900
      PSN=RIPUNF(MINNEG,NEGRCV,4)
      ROW=SCRCH1(PSN)
      GO TO 6880
 6770 IF(MINPOS.EQ.0)GO TO 6900
      PSN=RIPUNF(MINPOS,POSRCV,4)
      ROW=SCRCH2(PSN)
      GO TO 6880
C-----CONTINUOUS ROW
 6800 ROW=RIPUNF(III+1,MMM-EFRAC,4)
C-----STORE COEFFICIENT AND TAG RECEIVER ROW
 6880 IF(ROW.LE.ESLK+EINT.OR.ROW.GT.MMM-EFRAC)GO TO 6885
      IF(DATTYP.EQ.1.OR.DENSA.LE.200)SCRCH3(ROW)=1
      IF(DATTYP.EQ.0.AND.RIPUNF(0,2,4).NE.0)COEF=-COEF
      GO TO 6890
C-----INTEGER NEW ROW
 6885 SCRCH3(ROW)=1
      KKK=III-ESLK
      IF(III.GE.FRAC1)KKK=KKK-ECONT
      IF(GGG(KKK).EQ.0)GO TO 6890
      IF(SCRCH4(ROW).GE.-2)GO TO 6900
      RWORK=DELTA(III)
      COEF2=GGG(KKK)*RWORK/DELTA(MMM)
      COEF=MAX0(COEF,-(SCRCH4(ROW)+2)/COEF2)
      IF(COEF.EQ.0)GO TO 6900
      SCRCH4(ROW)=SCRCH4(ROW)+GGG(KKK)*COEF
 6890 CALL RIPADD(MATRIX,RRCW1(ROW),III,COEF)
C-----END KRK LOOP
 6900 CONTINUE
C
C        ***SECTION TO INVERT RRR****
C
 7000 RINVR1(1)=0
```

```
C-----MULTIPLY BY THE D-MATRIX AND EXPAND THE COLUMN
      DO 7600 KKK=1,MMM
      SCRCH1(KKK)=0.
 7600 CONTINUE
      NOW=ACOL1(JJJ)
 7620 KKK=MATRIX(1,NOW)
      SCRCH1(KKK)=MATRIX(2,NOW)*DELTA(KKK)
      NOW=MATRIX(3,NOW)
      IF(NOW.GT.0)GO TO 7620
C-----RELEASE OLD COLUMN SPACE
      CALL RIPDEL(MATRIX,ACOL1(JJJ))
C-----OUT EXPANDED COLUMN WITH ROWS OF R-INVERSE
      DO 7680 KKK=1,MMM
      COEF=RIPDOT(MATRIX,RINVR1(KKK),-1,SCRCH1,MMM)
      IF(COEF.EQ.0)GO TO 7680
      CALL RIPADD(MATRIX,ACOL1(JJJ),KKK,COEF)
 7680 CONTINUE
C-----END III LOOP
 7700 CONTINUE
C-----PRINT BBB MATRIX
      IF(LVLPRT.GE.5) CALL
     *   RIPVPR(1,MATRIX,ACOL1,MMM,BBB,1,MMM,2,LTBBB)
C
C-----RELEASE R-INVERSE MATRIX SPACE
      DO 7720 III=1,MMM
      CALL RIPDEL(MATRIX,RINVR1(III))
 7720 CONTINUE
C
C        ***SECTION TO POST MULTIPLY BY D-INV*RRR TO
C           COMPLETE
C           AND FLOAT THE MATRIX FFF***
C
      FTOL=1
      FTOL=FTOL/(2*DELTA(MMM))
C-----LOOP THROUGH ROWS OF FFF
      DO 7900 CUT=1,MAXCUT
C-----CLEAR RESULT AREA
      DO 7800 III=1,MMM
      XSCRCH(III)=0
 7800 CONTINUE
C-----LOOP THROUGH ELEMENTS OF A ROW OF FFF
      NOWF=FROW1(CUT)
 7820 COEF=MATRIX(2,NOWF)
      ROW=MATRIX(1,NOWF)
C-----LOOP THROUGH ELEMENTS OF RRR ROW
      NOWR=RROW1(ROW)
 7850 KKK=MATRIX(1,NOWR)
      RWORK=DELTA(ROW)
      XSCRCH(KKK)=XSCRCH(KKK)+COEF*MATRIX(2,NOWR)
     *   /RWORK
      NOWR=MATRIX(3,NOWR)
C-----END RRR LOOP
      IF(NOWR.GT.0) GO TO 7850
C-----END FFF LOOP
      NOWF=MATRIX(3,NOWF)
      IF(NOWF.GT.0)GO TO 7820
C-----RELEASE OLD ROW SPACE
      CALL RIPDEL(MATRIX,FROW1(CUT))
C-----COMPRESS NEW ROW
      DO 7880 III=1,MMM
      IF(XSCRCH(III).LE.FTOL.AND.XSCRCH(III).GE.-FTOL)GO TO
     *   7880
      ZCOEF=XSCRCH(III)
      CALL RIPADD(MATRIX,FROW1(CUT),III,ZCOEF)
 7880 CONTINUE
C-----END CUT ROW LOOP
 7900 CONTINUE
C
C-----RELEASE RRR MATRIX AREA
      DO 7920 III=1,MMM
      CALL RIPDEL(MATRIX,RROW1(III))
 7920 CONTINUE
C
C        ***SECTION TO COMPUTE CUT RHS  (1-PHI)***
C
C-----GET TRUE XBAR VALUES
 8200 DO 8300 PSN=1,MMM
      JJJ=BBB(PSN)
      XSCRCH(PSN)=XBAR(JJJ)
      XSCRCH(PSN)=XSCRCH(PSN)/DELTM
 8300 CONTINUE
C
C-----COMPUTE 1-PHI = 1-FRACPT(LLL*XBAR)
      DO 8400 CUT=1,MAXCUT
      RWORK=RIPDOT(MATRIX,-1,LROW1(CUT),XSCRCH,MMM)
      WORK=RWORK
      IF(RWORK.LT.0.)WORK=WORK-1
      RHS(MMM+CUT)=1.-RWORK+WORK
 8400 CONTINUE
C
      IF(LVLPRT.LT.4)GO TO 9000
C-----PRINT CUT RIGHT-HAND-SIDES
 8800 FORMAT(/20H ***CUT RHS 1-PHI***/
     *   (5X,3(E12.5,7H IN ROW,I4,1H/)))
      WRITE(LUPR,8800)(RHS(III+MMM),III,III=1,MAXCUT)
C-----PRINT LLL AND FFF
      CALL RIPVPR(1,MATRIX,FROW1,MMM,SCRCH1,1,MAXCUT,3,
     *   LTFFF)
      IF(LVLPRT.GE.5)CALL RIPVPR(1,MATRIX,LROW1,MMM,
     *   SCRCH1,1,MAXCUT,3,LTLLL)
```

```
C
C        ***EXIT SECTION***
C
 9100 CONTINUE
      RETURN
      END




C
C-------------------------------------------------------------
C
C-----SUBROUTINE RIPPH2 (MATRIX,ACOL1,NU,XBAR,XSTAR,BBB,PPP,
C     *    QQQ,OMEGA,VBLTYP,ROWTYP,DELQ,DELX,SCRCH1,SCRCH2,
C     *    DELB,DELTA,DDD,RHS,LIVX,LIVQ,SSS,TTT)
C-----SUBROUTINE TO EXECUTE PHASE II OF THE RIP RANDOM
C     INTEGER PROGRAM GENERATOR. PARAMETERS NOT DEFINED
C     IN RIPPH OR IN THE RANDIN AND LIN PAPER ARE AS
C        FOLLOWS--
C        SCRCH1,SCRCH2=WORK VECTORS USED FOR SEVERAL
C           TEMPORARY QUANTITIES
C        DELQ=THE VECTOR OF NET CHANGE IN SOLUTION FROM
C           THE BAR VECTOR FOR QQQ TO THE STAR VECTOR
C        DELX=THE VECTOR OF NET CHANGE IN SOLUTION FROM
C           THE BAR VECTOR FOR PPP (EXCEPT SLACKS) AND
C           QQQ TAKEN TOGETHER
C        DELB=THE VECTOR OF NET RHS IN MOVING FROM THE
C           BAR TO THE STAR SOLUTION IN THE BBB SUBMATRIX
C        LIVX=A LIST CORRESPONDING TO DELX OF COLUMN
C           SUBSCRIPTS STILL ELIGIBLE FOR NONZERO ENTRIES
C        LIVQ=A LIST CORRESPONDING TO DELQ OF QQQ COLUMN
C           SUBSCRIPTS STILL ELIGIBLE FOR NONZERO ENTRIES
C
      SUBROUTINE RIPPH2 (MATRIX,ACOL1,NU,XBAR,XSTAR,BBB,PPP,
     *    QQQ,OMEGA,VBLTYP,ROWTYP,DELQ,DELX,SCRCH1,SCRCH2,
     *    DELB,DELTA,DDD,RHS,LIVX,LIVQ,SSS,TTT)
C
      IMPLICIT INTEGER (A-Y)
C
      COMMON /RIPCOM/SOLBAR,SOLSTR,TOLMOV,TOLOFL,TOLOPT,
     *    TOLSTP,TOLVER,BCONT,BDCONT,BDINT,BBSLK,BFRAC,
     *    BINT,BSLK,BTITST,CORBOT,CORTOP,DATTYP,DELTM,
     *    DENSA,DENSF,DENSP,DENSPQ,DENSQ,DETERM,
     *    DIFBEG,DIFERM,DISRAT,ECONT,EFRAC,EINT,ESLK,
     *    FRAC1,INFIN,LINA,LINALP,LIPD,LIMF,LIMR,LIMSLK,
     *    LINVAR,LINVMX,LIMMIN,LIMMAX,LOWR,LVLPRT,
     *    MAXCON,MAXCUT,MAXINT,MAX 1,MMM,MU,
     *    NAN,NNNP,NNNQ,NNNS,NNNT,NUMERR,NXTMAT,PRECN,PSLK,
     *    RIPSLK,SEED(2,7),SSLK,TOLMU,VERIFY,VERSN
C
      DOUBLE PRECISION SOLBAR,SOLSTR,TOLMOV,TOLOFL,TOLOPT,
     *    TOLSTP,TOLVER
C
      DIMENSION MATRIX(3,1),ACOL1(1),NU(1),XBAR(1),XSTAR(1),
     *    BBB(1),
     *    PPP(1),QQQ(1),OMEGA(1),VBLTYP(1),ROWTYP(1),
     *    DELQ(1),DELX(1),SCRCH1(1),SCRCH2(1),DELB(1),
     *    DELTA(1),DDD(1),RHS(1),LIVX(1),LIVQ(1),SSS(1),
     *       TTT(1)
C
      DOUBLE PRECISION RHS
C
      DATA LTPPP,LTQQQ,LTSSS,LTTTT/3HPPP,3HQQQ,3HSSS,3HTTT/
C
C        ***SECTION TO CALCULATE RHS DIFFERENCE OF BAR
C           AND STAR SOLUTIONS***
C
C-----CLEAR DELTA BBB AND RHS
      DO 2200 III=1,MMM
      RHS(III)=0
      DELB(III)=0
 2200 CONTINUE
C-----LOOP THROUGH FRACTIONAL XXXB
      DO 2400 PSN=FRAC1,MMM
      JJJ=BBB(PSN)
C-----CALCULATE XXX CHANGE
      DEL=XBAR(JJJ)-XSTAR(JJJ)
      STAR=XSTAR(JJJ)/DELTA(MMM)
C-----MULTIPLY BY CORRESPONDING VECTOR OF BBB
      NOW=ACOL1(JJJ)
 2350 III=MATRIX(1,NOW)
      DELB(III)=DELB(III)-DEL*MATRIX(2,NOW)
      RHS(III)=RHS(III)+MATRIX(2,NOW)*STAR
      NOW=MATRIX(3,NOW)
      IF(NOW.GT.0)GO TO 2350
 2400 CONTINUE
C-----DIVIDE BY DELTA(MMM) TO REDUCE TO TRUE VALUE
      DO 2500 III=1,MMM
      DELB(III)=DELB(III)/DELTA(MMM)
 2500 CONTINUE
C-----LOOP THROUGH WHOLE INTEGER XXXB
      MAX=FRAC1-1
      DO 2600 PSN=1,MAX
      JJJ=BBB(PSN)
C-----CALCULATE XXX CHANGE
      DEL=(XBAR(JJJ)-XSTAR(JJJ))/DELTA(MMM)
      STAR=XSTAR(JJJ)/DELTA(MMM)
```

```
C-----MULTIPLY BY CORRESPONDING VECTOR OF BBB
      NOW=ACOL1(JJJ)
 2550 III=MATRIX(1,NOW)
      DELB(III)=DELB(III)-DEL*MATRIX(2,NOW)
      RHS(III)=RHS(III)+MATRIX(2,NOW)*STAR
      NOW=MATRIX(3,NOW)
      IF(NOW.GT.0)GO TO 2550
 2600 CONTINUE
C-----PRINT DELB
      IF(LVLPRT.LT.5)GO TO 3100
 2800 FORMAT(/21H ***DELB***,5I22/(11X,5I22))
      WRITE(LUPR,2800)(DELB(III),III=1,MM)
C
C        ***SECTION TO GENERATE UPPER BOUNDS AND
C           SOLUTION VALUES FOR XXXP AND XXXQ***
C
 3100 KKK=0
C-----PPP PORTION
      DO 3300 PSN=1,NUMP
      JJJ=PPP(PSN)
      XBAR(JJJ)=0
      TYP=VBLTYP(JJJ)
      GO TO (3300,3300,3210,3230,3250),TYP
C-----CONTINUOUS VARIABLE
 3210 CALL RIPND(XSTAR(JJJ),DDD(JJJ),NU(JJJ),0,3)
      GO TO 329
C-----GENERAL INTEGER VARIABLE
 3230 DDD(JJJ)=RIPUNF(1,LIMD,3)
      XSTAR(JJJ)=RIPOST(DDD(JJJ),3)
      GO TO 329.
C-----ZERO-ONE INTEGER VARIABLE
 3250 DDD(JJJ)=1
      XSTAR(JJJ)=1
C-----COMPLETE PPP CASES
 3290 XSTAR(JJJ)=MAX0(1,XSTAR(JJJ))
      KKK=KKK+1
      LIVX(KKK)=JJJ
      DELX(KKK)=-XSTAR(JJJ)
 3300 CONTINUE
C-----ASSURE ONE PPP HAS A UNIT DELX
      XSTAR(JJJ)=1
      DELX(KKK)=-1
C
C-----QQQ PORTION
      DO 3500 PSN=1,NNNQ
      JJJ=QQQ(PSN)
      TYP=VBLTYP(JJJ)
      GO TO (3500,3500,3410,3430,3450),TYP
C-----CONTINUOUS VARIABLE
 3410 CALL RIPND(XSTAR(JJJ),DDD(JJJ),NU(JJJ),1,3)
      GO TO 349.
C-----GENERAL INTEGER VARIABLE
 3430 DDD(JJJ)=RIPUNF(1,LIMD,3)
      XSTAR(JJJ)=DDD(JJJ)-RIPOST(DDD(JJJ),3)
      GO TO 349.
C-----ZERO-ONE INTEGER VARIABLE
 3450 DDD(JJJ)=1
      XSTAR(JJJ)=0
C-----FINISH QQQ CASES
 3490 XBAR(JJJ)=DDD(JJJ)
      XSTAR(JJJ)=MING(XSTAR(JJJ),DDD(JJJ)-1)
      DELQ(PSN)=XBAR(JJJ)-XSTAR(JJJ)
      KKK=KKK+1
      LIVX(KKK)=JJJ
      LIVQ(PSN)=JJJ
      DELX(KKK)=XBAR(JJJ)-XSTAR(JJJ)
 3500 CONTINUE
C-----ASSURE ONE QQQ HAS A UNIT DELX
      XSTAR(JJJ)=DDD(JJJ)-1
      DELX(KKK)=1
      DELQ(NNNQ)=1
      MAXKKK=KKK
C
C        ***SECTION TO CREATE ROWS OF PPP AND QQQ***
C
C-----LOOP THROUGH ROWS
      PPPPSN=0
      MAXPPP=NNNP-PSLK
      MAXQQQ=NNNQ
      OKP=0
      OKQ=0
      DO 7990 ICOMPL=1,MMM
      III=1+MMM-ICOMPL
C-----SEPARATE OUT INTEGER ROWS
      IF(III.GT.ESLK.AND.III.LE.ESLK+EINT)GO TO 5500
      IF(III.GT.MMM-EFRAC) GO TO 5500
C-----PROCESS A FREE ROW
      BRK=DATTYP*MAXKKK
      CALL RIPSPL(SCRCH1,DELX,DELB(III),DELB(III),LIMA,
     *   MAXKKK,SCRCH2,5,DENSPQ,BRK)
      IF(PSLK.EQ.0)GO TO 5310
C-----ADJUST FOR SLACK OR SURPLUS IF APPROPRIATE
      IF(III.LE.ESLK.OR.ROWTYP(III).EQ.0)GO TO 5300
C-----CREAT SLACK/SURPLUS COLUMN
      PPPPSN=PPPPSN+1
      JJJ=PPP(PPPPSN)
      IF(ROWTYP(III).EQ.1)VBLTYP(JJJ)=2
      CALL RIPADD(MATRIX,ACOL1(JJJ),III,ROWTYP(III))
C
C-----SHIFT SOME COEFFICIENT WEIGHT TO THE SLACK/SURPLUS
```

```
      DENS=DELSP
 5610 CALL RIPSPL(SCRCH1,DELX,DELB(III),DELB(III),LIMA,
     *    MAXPPP,SCRCH2,5,DENS,MAXPPP)
      IF(III.LT.MMM-1)GO TO 563
      DO 5620 KKK=1,MAXPPP
      IF(DELX(KKK).EQ.-1.AND.SCRCH1(KKK).NE.0)
     *    GO TO 5630
 5620 CONTINUE
      DENS=1000
      GO TO 5610
 5630 DO 5650 KKK=1,MAXPPP
      COEF=SCRCH1(KKK)
      IF(COEF.EQ.0)GO TO 5650
      JJJ=LIVX(KKK)
      RHS(III)=RHS(III)+COEF*XSTAR(JJJ)
      CALL RIPADD(MATRIX,ACOL1(JJJ),III,COEF)
 5650 CONTINUE
      GO TO 6000
C
C-----COEFFICIENTS IN QQQ
 5700 DENS=DENSQ
 5710 CALL RIPSPL(SCRCH1,DELQ,DELB(III),DELB(III),LIMA,
     *    MAXQQQ,SCRCH2,5,DENSQ,MAXQQQ)
      IF(III.LT.MMM-1)GO TO 573
      DO 5720 KKK=1,MAXQQQ
      IF(DELQ(KKK).EQ.1.AND.SCRCH1(KKK).NE.0)
     *    GO TO 5730
 5720 CONTINUE
      DENS=1000
      GO TO 5710
 5730 DO 5750 KKK=1,MAXQQQ
      COEF=SCRCH1(KKK)
      IF(COEF.EQ.0) GO TO 5750
      JJJ=LIVQ(KKK)
      RHS(III)=RHS(III)+COEF*XSTAR(JJJ)
      CALL RIPADD(MATRIX,ACOL1(JJJ),III,COEF)
 5750 CONTINUE
C-----END ROW LOOP
C
 6000 IF(ICOMPL.NE.EFROW)GO TO 7950
C
C-----CORRECT VACUOUS OR EMPTY INTEGER ROW
C
C-----PPP PORTION
      XSHIFT=0
      DO 7900 PSN=1,MAXPPP
      JJJ=PPP(PSN+PSLK)
      LIVX(PSN-XSHIFT)=LIVX(PSN)
      DELX(PSN-XSHIFT)=DELX(PSN)
      IF(ACOL1(JJJ).NE.0)GO TO 7900
```

```
      DO 5870 KKK=1,MAXKKK
      IF(ROWTYP(III)*SCRCH1(KKK)*DELX(KKK))5220,5270,5250
C-----OK TO DECREASE COEFFICIENT MAGNITUDE
 5220 LIM=RTPSLK*IABS(SCRCH1(KKK))/1000
      ADJ=RIPUNF(0,LIM,5)
      ADJ=MIN0(ADJ,IABS(SCRCH1(KKK)-1))
      STAR=STAR+ADJ*IABS(DELX(KKK))
      IF(SCRCH1(KKK).LT.0)ADJ=-ADJ
      SCRCH1(KKK)=SCRCH1(KKK)-ADJ
      GO TO 5270
C-----OK TO INCREASE COEFFICIENT MAGNITUDE
 5250 LIM=(LIMA-IABS(SCRCH1(KKK)))*RTPSLK/1000
      ADJ=MAX0(ADJ,1)
      STAR=STAR+ADJ*IABS(DELX(KKK))
      IF(SCRCH1(KKK).LT.0)ADJ=-ADJ
      SCRCH1(KKK)=SCRCH1(KKK)+ADJ
 5270 CONTINUE
C
C-----SET SLACK/SURPLUS VALUE
      XSTAR(JJJ)=STAR
C
C-----ADD NEW ROW COEFFICIENTS TO AAA MATRIX
 5300 DO 5360 KKK=1,MAXKKK
      COEF=SCRCH1(KKK)
      IF(COEF.EQ.0) GO TO 5380
      JJJ=LIVX(KKK)
C-----STORE COEFFICIENT
 5330 CALL RIPADD(MATRIX,ACOL1(JJJ),III,COEF)
      RHS(III)=RHS(III)+COEF*XSTAR(JJJ)
 5360 CONTINUE
      GO TO 6000
C
C-----PROCESS AN INTEGER ROW
 5500 IF(DATTYP.EQ.1)GO TO 5600
C-----MIXED SIGNS IN AAA MATRIX
      BRK=-MAXPPP*OMEGA(III)
      DENS=DENSPQ
 5550 CALL RIPSPL(SCRCH1,DELX,DELB(III),DELB(III),LIMA,
     *    MAXKKK,SCRCH2,5,DENS,BRK)
      IF(III.LT.MMM-1)GO TO 5300
      DO 5590 KKK=1,MAXKKK
      IF(DELX(KKK).EQ.-1.AND.SCRCH1(KKK).NE.0)OKP=1
      IF(DELX(KKK).EQ.1.AND.SCRCH1(KKK).NE.0)OKQ=1
      IF(OKP+OKQ.GE.ICOMPL)GO TO 5300
 5590 CONTINUE
      DENS=1000
      GO TO 5550
C-----NONNEGATIVE AAA MATRIX CASES
 5600 IF(OMEGA(III).GT.0)GO TO 5700
C-----COEFFICIENTS IN PPP
```

```
      IF(CATTYP.EQ.1.AND.PSN.EQ.MAXPPP)GO TO 7500
      XSHIFT=XSHIFT+1
      XSTAR(JJJ)=XBAR(JJJ)
 7500 CONTINUE
      MAXPPP=MAXPPP-XSHIFT
      QSHIFT=0
C
C-----QQQ PORTION
      DO 7900 PSN=1,NNNQ
      JJJ=QQQ(PSN)
      KKK=PSN+NNNP-PSLK
      LIVX(KKK-XSHIFT)=LIVX(KKK)
      DELX(KKK-XSHIFT)=DELX(KKK)
      LIVQ(PSN-QSHIFT)=LIVQ(PSN)
      DELQ(PSN-QSHIFT)=DELQ(PSN)
      IF(ACOL1(JJJ).NE.0)GO TO 7900
      IF(CATTYP.EQ.1.AND.PSN.EQ.NNNQ)GO TO 7900
      XSHIFT=XSHIFT+1
      QSHIFT=QSHIFT+1
      XSTAR(JJJ)=0
      XBAR(JJJ)=0
 7900 CONTINUE
C
      MAXQQQ=NNNQ-QSHIFT
      MAXKKK=MAXKKK-XSHIFT
C
 7990 CONTINUE
C
C        ***SECTION TO MOVE VACUOUS COLS TO SSS AND TTT***
C
      PSHIFT=0
      IF(NNNP.EQ.0)GO TO 8400
      DO 8300 PSN=1,NNNP
      JJJ=PPP(PSN)
      IF(ACOL1(JJJ).NE.0)GO TO 8250
      PSHIFT=PSHIFT+1
      NU(JJJ)=1
      NNNS=NNNS+1
      SSS(NNNS)=PPP(PSN)
      GO TO 8300
 8250 PPP(PSN-PSHIFT)=PPP(PSN)
 8300 CONTINUE
      NNNP=NNNP-PSHIFT
C
C-----QQQ SECTION
 8400 QSHIFT=0
      IF(NNNQ.EQ.0)GO TO 8600
      DO 8500 PSN=1,NNNQ
      JJJ=QQQ(PSN)
      IF(ACOL1(JJJ).NE.0)GO TO 8550
      QSHIFT=QSHIFT+1
      NU(JJJ)=1
      NNNT=NNNT+1
      TTT(NNNT)=QQQ(PSN)
      GO TO 8500
 8550 QQQ(PSN-QSHIFT)=QQQ(PSN)
 8500 CONTINUE
      NNNQ=NNNQ-QSHIFT
C
 8600 IF(LVLPRT.LT.5)GO TO 8960
C-----PRINT PPP AND QQQ
      CALL RIPVPR(2,MATRIX,ACOL1,MMM,PPP,1,NNNP,2,LTPPP)
      CALL RIPVPR(2,MATRIX,ACOL1,MMM,QQQ,1,NNNQ,2,LTQQQ)
 8950 FORMAT(/25H COLUMNS NOW IN SUBMATRIX,1X,A3/
     *    (5X,12I5))
 8960 IF(LVLPRT.LT.4)GO TO 9000
      IF(NNNP.GT.0)
     *    WRITE(LUPR,8950)LTPPP,(PPP(PSN),PSN=1,NNNP)
      IF(NNNQ.GT.0)
     *    WRITE(LUPR,8950)LTQQQ,(QQQ(PSN),PSN=1,NNNQ)
      WRITE(LUPR,8950)LTSSS,(SSS(PSN),PSN=1,NNNS)
      WRITE(LUPR,8950)LTTTT,(TTT(PSN),PSN=1,NNNT)
C
C        ***EXIT SECTION***
C
 9000 CONTINUE
      RETURN
      END
```

```
C-----------------------------------------------------------
C
C-----SUBROUTINE RIPPH3(MATRIX,ACOL1,ACOL1,BBB,PPP,QQQ,
C   OMEGA,
C     *   ROWTYP,FROW1,LROW1,CCC,UBAR,USTAR,VSTAR,LDELXB,
C     *   LXBSTR,MOVV,MOVUST,MOVUBR,MOVWP,WWWP,MOVWQ,
C     *   WWWQ,SCRCH,RBAR,RSTAR,DELTA,XBAR,XSTAR,RHS)
C
C-----SUBROUTINE TO EXECUTE PHASE III OF THE RIP RANDOM
C     INTEGER PROGRAM GENERATOR.  PARAMETERS NOT DEFINED
C     IN RIPPH OR IN THE PAPER BY RARDIN AND LIN ARE
C     AS FOLLOWS---
C        LDELXB=LLL*(XBBAR-XBSTAR)
C        LXBSTR=LLL*XBSTAR
C        MOVV=THE DIRECTION OF CHANGE IN VSTAR (THIS VECTOR
C           IS EQUIVALENCED THROUGH THE CALL WITH RBAR)
```

```
C        MOVUST=THE DIRECTION OF CHANGE IN USTAR
C        MOVUBR=THE DIRECTION OF CHANGE IN UBAR
C        WWWP=THE VECTOR OF OPTIMALITY CONDITIONS FOR
C           PPP VARIABLES
C        MOVWP=THE DIRECTION OF CHANGE IN WWWP
C        WWWQ=THE VECTOR OF OPTIMALITY CONDITIONS FOR
C           QQQ VARIABLES
C        MOVWQ=THE DIRECTION OF CHANGE IN WWWQ
C        SCRCH=A WORK VECTOR FOR TEMPORARY VALUES
C
      SUBROUTINE RIPPM3(MATRIX,ACOL1,ACOT1,BBB,PPP,QQQ,
     *   OMEGA,
     *   ROWTYP,FROW1,LROW1,CCC,UBAR,USTAR,VSTAR,LDELXB,
     *   LXBSTR,MOVV,MOVUST,MOVUBR,MOVWP,WWWP,MOVWQ,
     *   WWWQ,SCRCH,RBAR,RSTAR,DELTA,XBAR,XSTAR,RHS)
C
      IMPLICIT INTEGER (A-Y)
C
      COMMON /RIPCOM/SOLBAR,SOLSTR,TOLMOV,TOLOFL,TOLOPT,
     *   TOLSTP,TOLVER,BCONT,BDCONT,BDINT,BDSLK,BFRAC,
     *   BINT,BSLK,BTIIST,CORBOT,CURTOP,DATTYP,DELTM,
     *   DENBA,DENBF,DENBP,DENSPQ,DENSQ,DETERM,
     *   DIFBSG,DIMERM,DISPRT,FCONT,FFRAC,FINT,FSLK,
     *   FRAC1,INFIN,LIMA,LIMALP,LIPD,LIMF,LIPR,LIMSLK,
     *   LIMVMN,LIMVMX,LIMWMN,LIMWMX,LUPR,LVLPRT,
     *   MAXCON,MAXCUT,MAXINT,MAX 1,MMM,MU,
     *   NNN,NNNP,NNNQ,NNNS,NANT,NUMERR,NXTMAT,PRECN,PSLK,
     *   RTPSLK,SEED(2,7),SSLK,TOLMU,VERIFY,VERSN
C
      DOUBLE PRECISION SOLBAR,SOLSTR,TOLMOV,TOLOFL,TOLOPT,
     *   TOLSTP,TOLVER
      DOUBLE PRECISION CCC,UBAR,USTAR,VSTAR,LDELXB,LXBSTR,
     *   MOVV,MOVUST,
     *   MOVVAR,MOVUBR,MOVWP,WWWP,MOVWQ,WWWQ,VAR,
     *   EPS,RTSMU,MOVSST,STEP,SCRCH,RIPROT,
     *   FQ,RHS,RCOEF,DEL,RWORK,RAWMU2
C
      EQUIVALENCE (ZRELNO,INTGNO)
C
      DIMENSION MATRIX(3,1),ACOL1(1),ACOT1(1),BBB(1),PPP(1),
     *   QQQ(1),
     *   OMEGA(1),ROWTYP(1),FROW1(1),LROW1(1),CCC(1),
     *   UBAR(1),USTAR(1),VSTAR(1),LDELXB(1),
     *   LXBSTR(1),MOVV(1),MOVUST(1),MOVUBR(1),MOVWP(1),
     *   WWWP(1),MOVWQ(1),WWWQ(1),SCRCH(1),RBAR(1),RSTAR(1),
     *   DELTA(1),XBAR(1),XSTAR(1),RHS(1)
C
      DATA LTUBAR,LTUSTR,LTCCC,LTVSTR,LTLXBS,LTLDXB,LTRHS
     *   /4HUBAR,4HUSTR,3HCCC,4HVSTR,4HLXBS,4HLDXB,3HRHS/
C
```

```
C-----SET TOLERANCES
      TOLOPT=2
      TOLOPT=TOLOPT/DELTA(MMM)
      TOLSTP=TOLOPT/LIMWMX
      TOLSTP=TOLSTP/LIMWMN
      TOLMOV=10.*TOLOPT
      IF(TOLMOV.LT.1.)TOLMOV=1.
C
C        ***SECTION TO GENERATE INITIAL DUAL MULTIPLIERS***
C
C-----CLEAR UUU DUAL VECTORS
      DO 2500 III=1,MMM
      UBAR(III)=0
      USTAR(III)=0
 2500 CONTINUE
C-----LOOP THROUGH CUTS
      DO 2900 CUT=1,MAXCUT
C-----INITIALIZE CUT DUALS
      VSTAR(CUT)=RIPUNF(LIMVMN,LIMVMX,6)
C-----RSTAR WEIGHTS
      RSTAR(CUT)=RIPUNF(LIMWMN,LIMWMX,6)
C-----RBAR WEIGHTS
      RBAR(CUT)=RSTAR(CUT)+VSTAR(CUT)
C-----ADD UP UUU DUALS
      BAR=RBAR(CUT)
      STAR=RSTAR(CUT)
      NOW=FROW1(CUT)
 2700 IF(NOW.LE.0)GO TO 2900
      III=MATRIX(1,NOW)
      INTGNO=MATRIX(2,NOW)
      UBAR(III)=UBAR(III)+BAR*ZRELNO
      USTAR(III)=STAR*ZRELNO+USTAR(III)
      NOW=MATRIX(3,NOW)
      GO TO 2700
C-----END CUT LOOP
 2900 CONTINUE
C-----PRINT RBAR AND RSTAR
      IF(LVLPRT.LT.5)GO TO 3100
 2920 FORMAT(/1X,8(3H***)/1X,3HCUT,5X,4HRBAR,5X,5HRSTAR/(1X,
     *   I3,2I10))
      WRITE(LUPR,2920)(CUT,RBAR(CUT),RSTAR(CUT),
     *   CUT=1,MAXCUT)
C
C-----CORRECT UUU VALUES
 3100 DO 3150 III=1,MMM
      IF(III.LE.ESLK)GO TO 3150
      IF(OMEGA(III).NE.0)GO TO 3150
      MULT=-ROWTYP(III)
      IF(MULT.NE.0)GO TO 3120
      MULT=-1
```

```
      IF(XIPUNF(...,6).EQ.0)MULT=1
 3120 UBAR(III)=MULT*XIPUNF(LIMMMN,LIMMMX,6)
      USTAR(III)=UBAR(III)
 3150 CONTINUE
C
      CNUM=1
      STEP=INFIN
      STEP=STEP**2
C
C-----COMPUTE LLL*(XUSTAR-XUBAR) AND LLL*XUSTAR
      DO 3450 COUT=1,MAXCOUT
      LDELXB(COUT)=0.
      LXBSTR(COUT)=0.
      NOW=LROW1(COUT)
 3400 IF(NOW.EQ.0)GO TO 3450
      III=MATRIX(1,NOW)
      JJJ=BBB(III)
      RWORK=XSTAR(JJJ)-XBAR(JJJ)
      RWORK=RWORK/DELTM
      LDELXB(COUT)=LDELXB(COUT)+RWORK*MATRIX(2,NOW)
      RWORK=XSTAR(JJJ)
      RWORK=RWORK/DELTM
      LXBSTR(COUT)=LXBSTR(COUT)+RWORK*MATRIX(2,NOW)
      NOW=MATRIX(3,NOW)
      GO TO 3400
 3450 CONTINUE
      IF(LVLPRT.LT.5)GO TO 4000
      WRITE(LUPR,4420)LTLDXB,(LDELXB(COUT),COUT=1,MAXCOUT)
      WRITE(LUPR,4420)LTLXBS,(LXBSTR(COUT),COUT=1,MAXCOUT)
      WRITE(LUPR,4420)LTRHS,(RHS(III),III=1,MMM)
C
C        ***SECTION TO COMPUTE COST ROW AND TEMPORARY
C           SOLUTION VALUES***
C
C-----BBB PORTION
 4000 SOLSTR=0
      SOLBAR=0
      DO 4320 III=1,MMM
      JJJ=BBB(III)
      CCC(JJJ)=RIPROD(MATRIX,-1,ACOL1(JJJ),UBAR,MMM)
      SOLSTR=SOLSTR+CCC(JJJ)*XSTAR(JJJ)/DELTA(MMM)
      SOLBAR=SOLBAR+CCC(JJJ)*XBAR(JJJ)/DELTA(MMM)
 4320 CONTINUE
C-----PPP PORTION
      IF(NNNP.EQ.0) GO TO 4370
      DO 4360 PSN=1,NNNP
      JJJ=PPP(PSN)
      CCC(JJJ)=RIPROD(MATRIX,-1,ACOL1(JJJ),USTAR,MMM)
      SOLSTR=SOLSTR+CCC(JJJ)*XSTAR(JJJ)
 4360 CONTINUE
      IF(DATTYP.EQ.1)CCC(JJJ)=CCC(JJJ)+.001
C-----QQQ PORTION
 4370 IF(NNNQ.EQ.0) GO TO 4400
      DO 4380 PSN=1,NNNQ
      JJJ=QQQ(PSN)
      CCC(JJJ)=RIPROD(MATRIX,-1,ACOL1(JJJ),USTAR,MMM)
      SOLSTR=SOLSTR+CCC(JJJ)*XSTAR(JJJ)
      SOLBAR=SOLBAR+CCC(JJJ)*XBAR(JJJ)
 4380 CONTINUE
      IF(DATTYP.EQ.1)CCC(JJJ)=CCC(JJJ)-.001
C
 4400 RWORK=SOLSTR
      SGNSTR=1
      IF(RWORK.LT.0.)SGNSTR=-1
      IF(RWORK.NE.0.)RTSTMU=(SOLSTR-SOLBAR)/(RWORK*SGNSTR)
     *    *1.10
      TESTMU=RTSTMU
C
      IF(LVLPRT.LT.5)GO TO 4450
C-----PRINT CURRENT STATUS
 4410 FORMAT(/6H *****,5HCNUM=,I4,8H,SGNSTR=,I2,
     *    8H,RTSTMU=,E15.8,6(5H*****)
     *    /9H TESTMU=,I5,4H,MU=,I5,8H,SOLBAR=,E15.8,
     *    8H,SOLSTR=,E15.8,6H,STEP=,E15.8)
      WRITE(LUPR,4410)CNUM,SGNSTR,RTSTMU,TESTMU,MU,SOLBAR,
     *    SOLSTR,STEP
 4420 FORMAT(/1X,7HVECTOR ,A4/(5X,5E12.5))
      WRITE(LUPR,4420)LTUBAR,(UBAR(III),III=1,MMM)
      WRITE(LUPR,4420)LTUSTR,(USTAR(III),III=1,MMM)
      WRITE(LUPR,4420)LTVSTR,(VSTAR(KKK),KKK=1,MAXCUT)
      WRITE(LUPR,4420)LTCCC,(CCC(JJJ),JJJ=1,NNN)
C
C-----DECIDE WHETHER TO ACCEPT CCC AS IT IS
 4450 IF(CNUM.GT.MMM*DIFDEG)GO TO 9000
      IF(STEP.LT.TOLSTP)GO TO 9000
      IF(SOLSTR)4600,9000,4650
C-----SOLSTR NEGATIVE
 4600 IF(TESTMU.GE.DIFDEG*MU)GO TO 9000
      DIRECT=1
      VAR=DIFDEG*MU-RTSTMU
      VAR=VAR/1000
      IF(SOLSTR.GE.-1-LIMWMN)GO TO 9000
      GO TO 5000
C-----SOLSTR POSITIVE
 4650 IF(TESTMU.GE.MIN.(MU,50).AND.TESTMU.LE.MU)
     *    GO TO 9000
      DIRECT=1
      IF(TESTMU.GT.MU)DIRECT=-1
      VAR=MIN.(MU,50)-RTSTMU
      IF(TESTMU.GT.MU)VAR=MU-RTSTMU
```

```
      VAR=VAR/...
      IF(DIRDOT.GT. .ANG.SOLSTR.LE.1+LIMMIN)GO TO 9..
C
C        ***SECTION FOR FINDING A DIRECTION OF IMPROVEMENT
C           IN
C           THE DUAL MULTIPLIERS AND THE COST RATIO***
C
 5000 MOVVAR=.
      MOVSST=.
      RAWMU2=(SOLSTR-SOLBAS)/SOLSTR**2
C-----COMPUTE USTAR MOVEMENT DIRECTION
      DO 529. CUT=1,MAXCUT
      RWORK=SGNSTR*(LXELXH(CUT)/SOLSTR-LXBSTR(CUT)*RAWMU2)
      MOVV(CUT)=RWORK*DIRECT
      IF(MOVV(CUT).LT. .AND.VSTAR(CUT).LE.TOLMOV)MOVV(CUT)=0
      MOVVAR=MOVVAR+RWORK*MOVV(CUT)
      MOVSST=MOVSST+MOVV(CUT)*LXBSTR(CUT)
 5290 CONTINUE
C-----COMPUTE USTAR DIRECTION OF MOVEMENT
      DO 537. III=1,MMM
      IF(III.GT.ESLK)GO TO 532.
C-----SLACK ROW--DUAL DOES NOT CHANGE
      MOVUST(III)=0
      GO TO 537.
C-----MAIN ROW
 5320 MOVUST(III)=-DIRECT*SGNSTR*RAWMU2*RHS(III)
      IF(RCNTYP(III))533.,537.,5350
C-----A .GE. ROW--DUAL MUST STAY NONNEGATIVE
 5330 IF(MOVUST(III).LT.0.AND.USTAR(III).LT.TOLMOV)
     *    MOVUST(III)=.
      GO TO 537.
C-----A .LE. ROW--DUAL MUST STAY NONPOSITIVE
 5350 IF(MOVUST(III).GT.0.AND.USTAR(III).GT.-TOLMOV)
     *    MOVUST(III)=.
 5370 CONTINUE
C-----ZERO DIRECTIONS FOR USTARS ON SLACK MEMBERS OF PPP
      IF(PSLK.EQ.0)GO TO 5500
      DO 542. PSN=1,PSLK
      JJJ=PPP(PSN)
      NOW=ACOL1(JJJ)
      III=MATRIX(1,NOW)
      MOVUST(III)=0
 5420 CONTINUE
C-----COMPUTE CHANGE IN UBAR--USTAR PORTION
 5500 DO 552. III=1,MMM
      RWORK=-SGNSTR*RAWMU2*RHS(III)
      MOVVAR=MOVVAR+RWORK*MOVUST(III)
      MOVSST=MOVSST+RHS(III)*MOVUST(III)
      MOVUBR(III)=MOVUST(III)
 5520 CONTINUE
C-----USTAR PORTION (F-TRANSPOSE)
      DO 554. CUT=1,MAXCUT
      RCOEF=MOVV(CUT)
      NOW=FROW1(CUT)
 553. IF(NOW.LE.0)GO TO 554
      III=MATRIX(1,NOW)
      INTGNO=MATRIX(2,NOW)
      MOVUBR(III)=MOVUBR(III)+RCOEF*ZRELNO
      NOW=MATRIX(3,NOW)
      GO TO 553.
 554. CONTINUE
C-----COMPUTE CHANGE IN PPP  OPTIMALITY CONDITIONS
      IF(NNNP.LE.0) GO TO 560.
      DO 556. PSN=1,NNNP
      WWWP(PSN)=.
      MOVWP(PSN)=.
 556. CONTINUE
      DO 559. CUT=1,MAXCUT
      STRT=FROW1(CUT)
      DO 559. PSN=1,NNNP
      JJJ=PPP(PSN)
      IF(ACOL1(JJJ).LE.0)GO TO 559.
      FP=RIPROD(MATRIX,STRT,ACOL1(JJJ),SCRCH,MMM)
      STRT=-1
      WWWP(PSN)=WWWP(PSN)-FP*VSTAR(CUT)
      MOVWP(PSN)=MOVWP(PSN)-FP*MOVV(CUT)
 559. CONTINUE
C-----COMPUTE CHANGE IN QQQ OPTIMALITY CONDITIONS
 560. IF(NNNQ.LE.0)GO TO 564.
      DO 561. PSN=1,NNNQ
      WWWQ(PSN)=0
      MOVWQ(PSN)=0
 561. CONTINUE
      DO 563. CUT=1,MAXCUT
      STRT=FROW1(CUT)
      DO 563. PSN=1,NNNQ
      JJJ=QQQ(PSN)
      IF(ACOL1(JJJ).LE.0)GO TO 563.
      FQ=RIPROD(MATRIX,STRT,ACOL1(JJJ),SCRCH,MMM)
      STRT=-1
      WWWQ(PSN)=WWWQ(PSN)+FQ*VSTAR(CUT)
      MOVWQ(PSN)=MOVWQ(PSN)+FQ*MOVV(CUT)
 563. CONTINUE
C
C        ***SECTION TO STEP ALONG THE CHOSEN DIRECTION***
C
C-----COMPUTE STEP SIZE
 564. STEP=INFIN
      STEP=STEP**2
C-----USTAR OR UBAR SIGN LIMITS
```

```
      DO 569. III=1,MMM
      IF(ROWTYP(III))565.,5690,567.
C-----.GE. ROW
 5650 IF(MOVUST(III).GE. ) GO TO 5660
      RWORK=USTAR(III)/(-MOVUST(III))
      IF(RWORK.LT.STEP)STEP=RWORK
 5660 IF(MOVUBR(III).GE. )GO TO 5690
      RCOEF=UBAR(III)-TOLOPT
      IF(RCOEF.LT.UBAR(III)-1.)RCOEF=UBAR(III)-1.
      RWORK=RCOEF/(-MOVUBR(III))
      IF(RWORK.LT.STEP)STEP=RWORK
      GO TO 5690
C-----.LE. ROW
 5670 IF(MOVUST(III).LE. ) GO TO 5680
      RWORK=-USTAR(III)/MOVUST(III)
      IF(RWORK.LT.STEP)STEP=RWORK
 5680 IF(MOVUBR(III).LE. )GO TO 5690
      RCOEF=-UBAR(III)-TOLOPT
      IF(RCOEF.LT.-UBAR(III)-1.)RCOEF=-UBAR(III)-1.
      RWORK=RCOEF/MOVUBR(III)
      IF(RWORK.LT.STEP)STEP=RWORK
 5690 CONTINUE
C-----VSTAR NONNEGATIVITY
      DO 5720 CUT=1,MAXCUT
      IF(MOVV(CUT).GE.)) GO TO 5720
      RWORK=VSTAR(CUT)/(-MOVV(CUT))
      IF(RWORK.LT.STEP)STEP=RWORK
 5720 CONTINUE
C-----PPP OPTIMALITY LIMITS
      IF(NNNP.LE.0)GO TO 5750
      DO 5740 PSN=1,NNNP
      IF(MOVWP(PSN).GE.0)GO TO 5740
      RCOEF=WWWP(PSN)-TOLOPT
      IF(RCOEF.LT.WWWP(PSN)-1.)RCOEF=WWWP(PSN)-1.
      RWORK=-RCOEF/MOVWP(PSN)
      IF(RWORK.LT.STEP)STEP=RWORK
 5740 CONTINUE
C-----QQQ OPTIMALITY LIMITS
 5750 IF(NNNQ.LE.0)GO TO 5900
      DO 5760 PSN=1,NNNQ
      IF(MOVWQ(PSN).GE.0)GO TO 5760
      RCOEF=WWWQ(PSN)-TOLOPT
      IF(RCOEF.LT.WWWQ(PSN)-1.)RCOEF=WWWQ(PSN)-1.
      RWORK=-RCOEF/MOVWQ(PSN)
      IF(RWORK.LT.STEP)STEP=RWORK
 5760 CONTINUE
C-----VARIANCE REACHES ZERO
 5900 IF(MOVVAR.NE. ).)RWORK=VAR/MOVVAR
      IF(MOVVAR.NE. )..AND.RWORK.LT.STEP)STEP=RWORK
C-----SOLSTR REACHES ITS MINIMUM ABSOLUTE VALUE
      IF(MOVSST.NE...)RWORK=(SGNSTR*SOLSTR-LIMMMN)
     *   /MOVSST*SGNSTR
      IF(MOVSST.NE.0..AND.RWORK.LT.0..AND.-RWORK.LT.STEP)
     *   STEP=-RWORK
C-----REVISE USTAR AND INITIALIZE UBAR
      DO 5920 III=1,MMM
      USTAR(III)=USTAR(III)+STEP*MOVUST(III)
      UBAR(III)=0
 5920 CONTINUE
C-----REVISE VSTAR AND COMPUTE VSTAR*FFF TERM OF UBAR
      DO 5940 CUT=1,MAXCUT
      VSTAR(CUT)=VSTAR(CUT)+STEP*MOVV(CUT)
      RCOEF=VSTAR(CUT)
      NOW=FROM1(CUT)
 5930 IF(NOW.LE.0)GO TO 5940
      III=MATRIX(1,NOW)
      INTGNO=MATRIX(2,NOW)
      UBAR(III)=UBAR(III)+RCOEF*ZRELNO
      NOW=MATRIX(3,NOW)
      GO TO 5930
 5940 CONTINUE
C-----COMPLETE UBAR
      DO 5960 III=1,MMM
      UBAR(III)=UBAR(III)+USTAR(III)
 5960 CONTINUE
C
C-----LOOP BACK TO REVISE COSTS
      CNUM=CNUM+1
      GO TO 4000
C
C        ***EXIT SECTION***
C
 9000 CONTINUE
      IF(LVLPRT.NE.4)GO TO 9500
      WRITE(LUPR,4410)CNUM,SGNSTR,RTSTMU,TESTMU,MU,SOLBAR,
     *   SOLSTR,STEP
      WRITE(LUPR,4420)LTUBAR,(UBAR(III),III=1,MMM)
      WRITE(LUPR,4420)LTUSTR,(USTAR(III),III=1,MMM)
      WRITE(LUPR,4420)LTVSTR,(VSTAR(KKK),KKK=1,MAXCUT)
 9500 RETURN
      END




C
C-------------------------------------------------------------
C
```

```
C
C-----SSS PORTION
      IF(NNNS.LE.0)GO TO 2450
      DO 2300 PSN=1,NNNS
      JJJ=SSS(PSN)
      XBAR(JJJ)=0.
      XSTAR(JJJ)=0.
      IF(VBLTYP(JJJ).LE.2)GO TO 2300
      DDD(JJJ)=RIPUNF(1,LIMG,7)
      IF(VBLTYP(JJJ).EQ.5)DDD(JJJ)=1
 2300 CONTINUE
C
C-----TTT PORTION
 2450 IF(NNNT.LE.0)GO TO 3100
      DO 2500 PSN=1,NNNT
      JJJ=TTT(PSN)
      DDD(JJJ)=RIPUNF(1,LIMG,7)
      IF(VBLTYP(JJJ).EQ.5)DDD(JJJ)=1
      XSTAR(JJJ)=DDD(JJJ)
      XBAR(JJJ)=DDD(JJJ)
 2500 CONTINUE
C
C        ***SECTION TO GENERATE SSS AND TTT SUBMATRICES***
C
C-----COUNT NUMBER OF MATRIX CELLS FILLED
 3100 HAVE=0
      COLS=0
      DO 3250 JJJ=1,NNN
      IF(ACOL1(JJJ).EQ.0)COLS=COLS+1
      NOW=ACOL1(JJJ)
 3200 IF(NOW.EQ.0)GO TO 3250
      HAVE=HAVE+1
      NOW=MATRIX(3,NOW)
      GO TO 3200
 3250 CONTINUE
C
C-----DECIDE THE NUMBER LEFT TO FILL
      RWORK=NNN*MMM*DENSA
      DESIRE=RWORK/1000+.5
      NEED=DESIRE-HAVE
      LEFT=MMM*COLS
C
C-----COMPUTE TTT COEFFICIENT BIASES
      DO 3500 III=1,MMM
      XCRCH(III)=USTAR(III)
 3500 CONTINUE
      DO 3540 OUT=1,MAXOUT
      NOW=FROW1(OUT)
 3520 IF(NOW.LE.0)GO TO 3540
      III=MATRIX(1,NOW)
```

```
C-----SUBROUTINE RIPPH4(MATRIX,SSS,TTT,BBB,VSTAR,USTAR,
C     *  UBAR,CCC,RHS,LROW1,FROW1,ACOL1,ACUT1,OMEGA,XBAR,
C     *  XSTAR,DDD,ROWTYP,VBLTYP,XCRCH,TBIAS)
C-----SUBROUTINE TO EXECUTE PHASE IV OF THE RIP RANDOM
C     INTEGER PROGRAM GENERATOR.  ALL VARIABLES AND NAMES
C     ARE AS DEFINED IN THE RANDIN AND LIN PAPER ON
C     RIP OR AS DEFINED IN SUBROUTINE RIPPH.
C        XCRCH=A DOUBLE PRECISION WORK VECTOR
C        TBIAS=A VECTOR OF INTEGERS USED TO BIAS THE
C           TTT SUBMATRIX TOWARD A NONNEGATIVE TTT COST
C
      SUBROUTINE RIPPH4(MATRIX,SSS,TTT,BBB,VSTAR,USTAR,
     *  UBAR,CCC,RHS,LROW1,FROW1,ACOL1,ACUT1,OMEGA,XBAR,
     *  XSTAR,DDD,ROWTYP,VBLTYP,XCRCH,TBIAS)
C
      IMPLICIT INTEGER (A-Y)
C
      COMMON /RIPCOM/SOLBAR,SOLSTR,TOLNOV,TOLOFL,TOLOPT,
     *   TOLSTP,TOLVER,BCONT,BDCONT,BDINT,BUSLK,BFRAC,
     *   BINT,BSLK,BTITST,CORBOT,CORTOP,DATTYP,DELTM,
     *   DENSA,DENSF,DENSP,DENSPQ,DENSQ,DETERM,
     *   DIFDEG,DIMFRM,DISKAT,ECONT,EFRAC,EINT,ESLK,
     *   FRAC1,INFIN,LIMA,LIMALP,LIMD,LIMF,LIMR,LIMSLK,
     *   LIMVMN,LIMVMX,LIMWMN,LIMWMX,LUPR,LVLPRT,
     *   MAXCON,MAXCUT,MAXINT,MAX01,MMM,MU,
     *   NNN,NNNP,NNNQ,NNNS,NNNT,NUMERR,NXTNA1,PRECN,PSLK,
     *   RTPSLK,SEED(2,7),SSLK,TOLMU,VERIFY,VERSN
C
      DOUBLE PRECISION SOLBAR,SOLSTR,TOLNOV,TOLOFL,TOLOPT,
     *   TOLSTP,TOLVER
C
      DIMENSION MATRIX(3,1),SSS(1),TTT(1),BBB(1),VSTAR(1),
     *  USTAR(1),UBAR(1),CCC(1),RHS(1),LROW1(1),
     *  FROW1(1),ACOL1(1),ACUT1(1),XCRCH(1),XBAR(1),
     *  XSTAR(1),DDD(1),ROWTYP(1),VBLTYP(1),OMEGA(1),
     *     TBIAS(1)
C
      EQUIVALENCE(INTGNO,ZRELNO)
C
      DOUBLE PRECISION SIGMA,RIPROT,XCRCH,BAR,STAR,RWORK,
     *   TTTCST,
     *   CCC,VSTAR,USTAR,UBAR,RHS,TGOAL,RMU
     *   ,TTTADJ,TUP,TDOWN
C
      DATA LITBIS,LITSSS,LITTTT,LITCCC,LITDDD,LITXBR,LITXST,
     *   LITCUT/
     *   4HBIAS,3HSSS,3HTTT,3HCCC,3HDDD,4HXBAR,4HXSTR,3HCUT/
C
C        ***SECTION TO SET GOLNOS AND VALUES FOR SSS AND
C           TTT***
```

```
C-----STORE COEFFICIENT
 4350 CALL RIPADD(MATRIX,ACOL1(JJJ),III,COEF)
      NEED=NEED-1
      IF(NEED.LE.COLS)GO TO 450
C-----END COEFFICIENT LOOP
 4400 CONTINUE
C
C-----GENERATE CUT COEFFICIENTS (IF ANY)
 4500 DO 4600 CUT=1,MAXCUT
      SIGMA=RIPROJ(MATRIX,FROM1(CUT),ACOL1(JJJ),XCRCH,MMM)
      IF(SIGMA.LE.-...AND.XBAR(JJJ).EQ.-
     *  .OR.SIGMA.GE.....AND.XBAR(JJJ).EQ.DDD(JJJ))
     *  GO TO 460
      ZIGMA=SIGMA/(1.-RHS(MMM+CUT))
      CALL RIPADD(MATRIX,ACUT1(JJJ),CUT,ZIGMA)
 4600 CONTINUE
C
C-----SET PRELIMINARY COST
      STAR=RIPROJ(MATRIX,-1,ACOL1(JJJ),USTAR,MMM)
     *   +RIPROJ(MATRIX,ACUT1(JJJ),-1,VSTAR,MAXCUT)
      IF(XBAR(JJJ).GT.0) GO TO 47..
C-----LOWER BOUNDED VARIABLE
      CCC(JJJ)=STAR+RIPUNF(1,LIMA,7)
      GO TO 490.
C-----UPPER BOUNDED VARIABLE
 4700 IF(STAR.GE...)CCC(JJJ)=.99*STAR-.01
      IF(STAR.LT...)CCC(JJJ)=1.01*STAR-.01
C
C-----END OF GENERATION LOOP
 4900 CONTINUE
C
C-----PRINT SSS AND TTT
      IF(LVLPRT.LT.5)GO TO 600.
      CALL RIPVPR(2,MATRIX,ACOL1,MMM,SSS,1,NNNS,2,LITSSS)
      CALL RIPVPR(2,MATRIX,ACOL1,MMM,TTT,1,NNNT,2,LITTTT)
 5000 FORMAT(/1X,7HVECTOR ,A4/(5X,5E12.5))
 5010 FORMAT(/1X,7HVECTOR ,A4/(5X,5I12))
      WRITE(LUPR,5000)LITCCC,(CCC(JJJ),JJJ=1,NNN)
      WRITE(LUPR,5010)LITDDD,(DDD(JJJ),JJJ=1,NNN)
      WRITE(LUPR,5010)LITXBR,(XBAR(JJJ),JJJ=1,NNN)
      WRITE(LUPR,5010)LITXST,(XSTAR(JJJ),JJJ=1,NNN)
C
C        ***SECTION TO CORRECT TTT COST CONTRIBUTION TO
C           ZERO***
C
 6000 IF(NNNT.LE.0)GO TO 80..
C-----TOTAL TTT COST CONTRIBUTION
      TSTAR=0
      TTTADJ=0.
      TTTCST=0.
      INTGNO=MATRIX(2,NOW)
      XCRCH(III)=XCRCH(III)+ZRELNO*VSTAR(CUT)
     *   /(1-RHS(CUT+MMM))
      NOW=MATRIX(3,NOW)
      GO TO 3520
 3540 CONTINUE
C
      DO 3560 III=1,MMM
      ZWORK=XCRCH(III)
      ZABS=ABS(ZWORK)
      RWORK=2
      RWORK=ALOG1.(ZABS+1)*RWORK**(DIFDEG-2)
      LOG=RWORK+1.
      TBIAS(III)=LOG
      IF(XCRCH(III).LT.0.)TBIAS(III)=-LOG
 3560 CONTINUE
      IF(LVLPRT.GE.5)WRITE(LUPR,5010)LITBIS,(TBIAS(III)
     *   ,III=1,MMM)
C
C-----MAIN COLUMN GENERATING LOOP
      DO 4900 JJJ=1,NNN
      IF(ACOL1(JJJ).GT.0)GO TO 4900
      COLS=COLS-1
C
      DO 4400 III=1,MMM
C-----DECIDE WHETHER TO PLACE A COEFFICIENT
      DENGEN=MAX0(0,NEED)*1000/LEFT
      LEFT=LEFT-1
      IF(RIPUNF(1,MMM-EFRAC,7).LE.III.AND.ACOL1(JJJ).EQ.0)
     *   GO TO 4150
      IF(RIPUNF(0,1000,7).GT.DENGEN)GO TO 4400
 4150 BIAS=TBIAS(III)
      IF(DATTYP.NE.3)GO TO 4300
C-----DATA TYPE 3 COEFFICIENT
 4200 IF(XSTAR(JJJ).EQ.0)COEF=RIPUNF(-LIMA,LIMA,7)
      IF(XSTAR(JJJ).EQ.DDD(JJJ).AND.BIAS.GT.0)
     *   COEF=RIPUNF(-LIMA/BIAS,LIMA*BIAS,7)
      IF(XSTAR(JJJ).EQ.DDD(JJJ).AND.BIAS.LT.0)
     *   COEF=RIPUNF(LIMA*BIAS,-LIMA/BIAS,7)
      IF(COEF.EQ.0)GO TO 4200
      GO TO 4350
C-----DATA TYPE 1 COEFFICIENT
 4300 IF(XSTAR(JJJ).EQ.0)COEF=RIPUNF(1,LIMA,7)
      IF(XSTAR(JJJ).EQ.DDD(JJJ).AND.BIAS.LE.-3.AND.
     *   ACOL1(JJJ).GT.0.AND.NEED.LE.LEFT)GO TO 4400
      IF(XSTAR(JJJ).EQ.DDD(JJJ).AND.BIAS.GT.0)
     *   COEF=RIPUNF(LIMA/2*BIAS,LIMA*BIAS,7)
      IF(XSTAR(JJJ).EQ.DDD(JJJ).AND.BIAS.LT.0)
     *   COEF=RIPUNF(0.,-LIMA/BIAS,7)
      IF(COEF.LE..)COEF=1
```

```
      DO 6150 PSN=1,NNNT
      JJJ=TTT(PSN)
      TTTCST=TTTCST+CCC(JJJ)*XSTAR(JJJ)
      TSTAR=TSTAR+XSTAR(JJJ)
      IF(CCC(JJJ).LT. .)TTTADJ=TTTADJ-(DDD(JJJ)-1)*CCC(JJJ)
      IF(CCC(JJJ).GT. .)TTTADJ=TTTADJ+DDD(JJJ)*CCC(JJJ)
 6150 CONTINUE
C
C-----SET TTT TOTAL COST GOAL
      RHO=1.00-DU
      RHO=RHO/1.00
      TUP=0.
      IF(SOLBAR.LT. .)TUP=-SOLBAR
      TUP=(RHO*(SOLSTR+TUP)-(SOLBAR+TUP))
     *   /(1.-RHO)+TUP
      TDOWN=0.
      IF(SOLSTR.GT.0.)TDOWN=-SOLSTR
      TDOWN=((RHO-2.)*(SOLSTR+TDOWN)+(SOLBAR+TDOWN))
     *   /(1.-RHO)+TDOWN
      IF(TUP.GT.TTTCST+TTTADJ)GO TO 6150
C-----MU UP OPTION IS POSSIBLE
      TGOAL=TTTCST-TUP
      IF(SOLSTR.GE. .)GO TO 6230
C-----MU DOWN OPTION IS BETTER
 6150 TGOAL=TTTCST-TDOWN
C
C-----PRINT TTTCST STATUS
 6220 FORMAT (/34H ***STARTING TOTAL OF TTT COSTS***,E14.7
     *   /21H ***TTT MAX ADJUST***,E14.7
     *   /20H ***TTT COST GOAL***,E14.7
     *   /18H ***TTT UP DIST***,E14.7
     *   /20H ***TTT DOWN DIST***,E14.7)
 6230 IF(LVLPRT.GE.4)WRITE(LUPR,6220)TTTCST,TTTADJ,TGOAL,
     *   TUP,TDOWN
C
      IF(TGOAL.LT.0.)GO TO 6500
C-----TTT COST GOAL IS NONNEGATIVE (SPREAD POSITIVE PART TO
C   GET 0)
 6250 DO 6300 PSN=1,NNNT
      JJJ=TTT(PSN)
      GOAL=2*TGOAL/TSTAR
      RHORK=RIPUNF( ,GOAL,7)
      IF(TGOAL.LT.RHORK*XSTAR(JJJ))RHORK=TGOAL/XSTAR(JJJ)
      CCC(JJJ)=CCC(JJJ)-RHORK
      TSTAR=TSTAR-XSTAR(JJJ)
      TGOAL=TGOAL-RHORK*XSTAR(JJJ)
 6300 CONTINUE
      CCC(JJJ)=CCC(JJJ)-TGOAL/XSTAR(JJJ)
      TGOAL=0.
      GO TO 6000
C-----TTT GOAL IS NEGATIVE (TRY TO MODIFY DDD TO GET 0)
 6500 DO 6600 PSN=1,NNNT
      JJJ=TTT(PSN)
      RHORK=CCC(JJJ)*XSTAR(JJJ)
      IF(CCC(JJJ).GT. .AND.VBLTYP(JJJ).NE.5)GO TO 6700
C-----CCC IS NEGATIVE OR VBL IS U-1
      TSTAR=TSTAR-DDD(JJJ)+1
      DDD(JJJ)=1
      XSTAR(JJJ)=1
      XBAR(JJJ)=1
      TGOAL=TGOAL-RHORK+CCC(JJJ)
      GO TO 6750
C-----CCC IS POSITIVE
 6700 TSTAR=TSTAR+DDD(JJJ)
      DDD(JJJ)=DDD(JJJ)*2
      XSTAR(JJJ)=DDD(JJJ)
      XBAR(JJJ)=DDD(JJJ)
      TGOAL=TGOAL-RHORK+CCC(JJJ)*XSTAR(JJJ)
C-----CHECK IF NOW NONNEGATIVE
 6750 IF(TGOAL.GE.0.)GO TO 6250
 6600 CONTINUE
C
C         ***SECTION TO WRAPUP HOUSEKEEPING***
C
 8000 IF(LVLPRT.LT.5)GO TO 8100
 8010 FORMAT(/31H ***FINAL DEVN FROM TTT GOAL***,E14.7)
      WRITE(LUPR,8010)TGOAL
      WRITE(LUPR,5000)LITCCC,(CCC(JJJ),JJJ=1,NNN)
      WRITE(LUPR,5010)LITDDD,(DDD(JJJ),JJJ=1,NNN)
      WRITE(LUPR,5010)LITXBR,(XBAR(JJJ),JJJ=1,NNN)
      WRITE(LUPR,5010)LITXST,(XSTAR(JJJ),JJJ=1,NNN)
C
C-----CORRECT FFF AND LLL STORAGE
 8100 DO 8300 CUT=1,MAXCUT
C-----DELETE FFF ROW
      CALL RIPDEL(MATRIX,FROW1(CUT))
C-----MOVE LLL TO CUT AREA BENEATH BBB
      NOW=LROW1(CUT)
 8200 IF(NOW.LE.0)GO TO 8250
      III=MATRIX(1,NOW)
      JJJ=BBB(III)
      ZCOEF=MATRIX(2,NOW)
      CALL RIPADD(MATRIX,ACUT1(JJJ),CUT,ZCOEF)
      NOW=MATRIX(3,NOW)
      GO TO 8200
C-----DELETE LLL ROW
 8250 CALL RIPDEL(MATRIX,LROW1(CUT))
 8300 CONTINUE
C
C-----PRINT CUTS
```

```
     *    FRAC1,INFIN,LIMA,LIMALP,LIMD,LIMF,LIMR,LIMSLK,
     *    LIMVAR,LIMVMX,LIMWRN,LIMWMX,LUPR,LVLPRT,
     *    MAXCUT,MAXCUT,MAXINT,MAX1,MM1,MU,
     *    NNN,NNNP,NNNQ,NNNS,NNNT,NUMERR,NXTMAT,PRECN,PSLK,
     *    RIPSLK,SEED(2,7),SSLK,TOLMU,VERIFY,VERSN
C
      DOUBLE PRECISION SOLBAR,SOLSTR,TOLMOV,TOLOFL,TOLOPT,
     *    TOLSTP,TOLVER
C
      DIMENSION MATRIX (3,1)
C
C-----ADVANCE MATRIX 3-TUPLE POINTER
      PSN=NXTMAT
      NXTMAT=MATRIX(3,PSN)
      IF(NXTMAT.GT.0)GO TO 150
C-----MATRIX AREA OVERFLOW
 120  FORMAT(1H1,49H***RUN TERMINATED DUE TO INADEQUATE AAA
     *    DIMENSION)
      WRITE(LUPR,1200)
      STOP 1
C-----PLACE NEW ENTRY AT HEAD OF THE LIST OF COL1
 150  MATRIX(3,PSN)=COL1
      MATRIX(2,PSN)=COEF
      MATRIX(1,PSN)=III
      COL1=PSN
C
C-----EXIT
 900  CONTINUE
      RETURN
      END
C
C-------------------------------------------------------------
C
C-----SUBROUTINE RIPDEL(MATRIX,COL1)
C-----SUBROUTINE TO FREE (DELINK) ALL MATRIX ENTRIES OF THE
C     RIP 3-TUPLE AREA THAT CORRESPOND TO A GIVEN COLUMN
C     OR ROW.  PARAMETERS ARE AS FOLLOWS--
C        MATRIX=THE GENERAL STORAGE AREA
C        COL1=THE 3-TUPLE IN MATRIX WITH THE FIRST DATA FOR
C           THIS COLUMN OR ROW
C
      SUBROUTINE RIPDEL(MATRIX,COL1)
C
      IMPLICIT INTEGER (A-Y)
```

```
      IF(LVLPRT.LT.4) GO TO 841
      CALL RIPVPR(2,MATRIX,-COL1,MAXCUT,ACOL1,1,NNN,3,
     *    LITOUT)
C
C-----COMPUTE SOLUTION VALUES AND RIGHT-HAND-SIDES
 841  IF(NNNT.LE.0)GO TO 900
      DO 850 I=1,NNNT
      JJJ=ITT(PSN)
      SOLBAR=SOLBAR+COC(JJJ)*XBAR(JJJ)
      SOLSTR=SOLSTR+COC(JJJ)*XSTAR(JJJ)
      NOW=ACOL1(JJJ)
 845  IF(NOW.EQ.0)GO TO 850
      III=MATRIX(1,NOW)
      RHS(III)=RHS(III)+MATRIX(2,NOW)*XSTAR(JJJ)
      NOW=MATRIX(3,NOW)
      GO TO 845
 850  CONTINUE
C
C        ***SECTION TO EXIT***
C
 900  CONTINUE
      RETURN
      END
```

```
C-----SUBROUTINE RIPADD(MATRIX,COL1,III,COEF)
C-----SUBROUTINE TO ADD ONE 3-TUPLE ENTRY IN THE GENERAL
C    MATRIX
C      AREA OF THE THE RIP GENERATOR. PARAMETERS ARE AS
C         FOLLOWS--
C         MATRIX=THE GENERAL STORAGE AREA
C         COL1=THE 3-TUPLE IN MATRIX WITH THE FIRST DATA FOR
C            THIS COLUMN OR ROW
C         III=THE INDEX WITHIN THE COLUMN OR ROW AT WHICH WE
C            WISH TO ADD A COEFFICIENT
C         COEF=THE COEFFICIENT
C
      SUBROUTINE RIPADD(MATRIX,COL1,III,COEF)
C
      IMPLICIT INTEGER (A-Y)
C
      COMMON /RIPCOM/SOLBAR,SOLSTR,TOLMOV,TOLOFL,TOLOPT,
     *    TOLSTP,TOLVER,BDCNT,BDCONT,BDINT,BDSLK,BFRAC,
     *    BINT,BSLK,BTITST,CORBOT,CORTOP,DATTYP,DELTM,
     *    DENSA,DENSF,DENSP,DENSPQ,DENSQ,DETERM,
     *    DIFDEG,DIMPRM,DISMAT,ECONT,EFRAC,EINT,ESLK,
```

```
C         STRT2=THE STARTING 3-TUPLE OF THE SECOND VECTOR
C            (=-1 IF THE VECTOR ALREADY EXIST AS EXPAND)
C         EXPAND=THE WORK AREA INTO WHICH ONE VECTOR WILL
C            BE EXPANDED IF BOTH BELONG TO MATRIX
C         LENGTH=THE LENGTH OF THE VECTORS INVOLVED
C
      INTEGER FUNCTION RIPDOT(MATRIX,STRT1,STRT2,EXPAND,
     *   LENGTH)
C
      IMPLICIT INTEGER (A-Y)
C
      COMMON /RIPCOM/SOLBAR,SOLSTR,TOLMOV,TOLOFL,TOLOPT,
     *   TOLSTP,TOLVER,BCONT,BDCONT,BDINT,BDSLK,BFRAC,
     *   BINT,BSLK,BTITST,CORDOT,CORTOP,DATTYP,DELTM,
     *   DENSA,DENSF,DENSP,DENSPQ,DENSQ,DETERM,
     *   DIFDEG,DIMPRM,DISRAT,ECONT,EFRAC,EINT,ESLK,
     *   FRAC1,INFIN,LIMA,LIMALP,LIMO,LIMF,LIMR,LIMSLK,
     *   LIMVMN,LIMVMX,LIMWMN,LIMWMX,LUPR,LVLPRT,
     *   MAXCON,MAXCUT,MAXINT,MAX1,MMM,MU,
     *   NNN,NNNP,NNNQ,NNNS,NNNT,NUMERR,NXTMAT,PRECN,PSLK,
     *   RTPSLK,SEED(2,7),SSLK,TOLMU,VERIFY,VERSN
C
      DOUBLE PRECISION SOLBAR,SOLSTR,TOLMOV,TOLOFL,TOLOPT,
     *   TOLSTP,TOLVER
C
      DIMENSION EXPAND(1),MATRIX(3,1)
C
C-----CHECK FOR VACUOUS VECTOR 1
      IF(STRT1)2200,2500,2500
C-----1ST VECTOR IS PRE-EXPANDED
 2200 NOW=STRT2
      GO TO 4000
C-----CHECK FOR VACUOUS VECTOR 2
 2500 IF(STRT2)2600,3000,3600
C-----2ND VECTOR IS PRE-EXPANDED
 2600 NOW=STRT1
      GO TO 4000
C-----EXPAND VECTOR 1 IF NEITHER PRE-EXPANDED
 3000 DO 3200 KKK=1,LENGTH
      EXPAND(KKK)=0
 3200 CONTINUE
      NOW=STRT1
 3500 IF(NOW.LE.0)GO TO 3900
      KKK=MATRIX(1,NOW)
      EXPAND(KKK)=MATRIX(2,NOW)
      NOW=MATRIX(3,NOW)
      GO TO 3500
C-----MULTIPLY THE UNEXPANDED BY THE EXPANDED
 3900 NOW=STRT2
 4000 RIPDOT=0
C
      COMMON /RIPCOM/SOLBAR,SOLSTR,TOLMOV,TOLOFL,TOLOPT,
     *   TOLSTP,TOLVER,BCONT,BDCONT,BDINT,BDSLK,BFRAC,
     *   BINT,BSLK,BTITST,CORDOT,CORTOP,DATTYP,DELTM,
     *   DENSA,DENSF,DENSP,DENSPQ,DENSQ,DETERM,
     *   DIFDEG,DIMPRM,DISRAT,ECONT,EFRAC,EINT,ESLK,
     *   FRAC1,INFIN,LIMA,LIMALP,LIMO,LIMF,LIMR,LIMSLK,
     *   LIMVMN,LIMVMX,LIMWMN,LIMWMX,LUPR,LVLPRT,
     *   MAXCON,MAXCUT,MAXINT,MAX1,MMM,MU,
     *   NNN,NNNP,NNNQ,NNNS,NNNT,NUMERR,NXTMAT,PRECN,PSLK,
     *   RTPSLK,SEED(2,7),SSLK,TOLMU,VERIFY,VERSN
C
      DOUBLE PRECISION SOLBAR,SOLSTR,TOLMOV,TOLOFL,TOLOPT,
     *   TOLSTP,TOLVER
C
      DIMENSION MATRIX(3,1)
C
C-----CHECK FOR VACUOUS CHAIN
      NXT=COL1
      IF(NXT.EQ.0)GO TO 9000
C-----LOCATE END OF A CHAIN
 2000 NOW=NXT
      NXT=MATRIX(3,NOW)
      IF(NXT.GT.0)GO TO 2000
C-----MOVE ENTIRE CHAIN TO CHAIN OF AVAILABLE 3-TUPLES
      MATRIX(3,NOW)=NXTMAT
      NXTMAT=COL1
      COL1=0
C-----EXIT
 9000 CONTINUE
      RETURN
      END
C
C-------------------------------------------------------------
C
C-----INTEGER FUNCTION RIPDOT(MATRIX,STRT1,STRT2,EXPAND,
C   LENGTH)
C-----FUNCTION TO RETURN THE DOT PRODUCT OF TWO VECTORS,
C     AT LEAST ONE OF WHICH IS STORED IN THE GENERAL
C     RIP 3-TUPLE STORAGE AREA.  BOTH VECTORS ARE INTEGER.
C     PARAMETERS ARE AS FOLLOWS--
C        MATRIX=THE GENERAL MATRIX STORAGE AREA
C        STRT1=THE STARTING 3-TUPLE OF THE FIRST VECTOR
C           (=-1 IF THE VECTOR ALREADY EXISTS AS EXPAND)
```

```
      RIPDST=
      IF(LIMIT.LE. )GO TO 9
C-----CHECK UNIFORM DISTRIBUTION CASE
      IF(DISRAT.NE.10 )GO TO 2
      RIPDST=RIPUNF(,LIMIT,GENR)
      GO TO 90
C-----COMPUTE TOTAL PROBABILITY WEIGHT
 200  WEIGHT=1.
      TERM=1.
      RATIO=DISRAT
      RATIO=RATIO/10
      DO 22 VAL=1,LIMIT
      TERM=TERM*RATIO
      WEIGHT=WEIGHT+TERM
 220  CONTINUE
C-----GENERATE RANDOM CUMMULATIVE POINT
      CUMM=WEIGHT/1 0 *RIPUNF(,1 ,GENR)
C-----DETERMINE CORRESPONDING RIPOST
      WEIGHT=1.
      TERM=1.0
      DO 25 VAL=1,LIMIT
      IF(WEIGHT.GE.CUMM)GO TO 30
      TERM=TERM*RATIO
      WEIGHT=WEIGHT+TERM
 250  CONTINUE
      RIPDST=LIMIT
      GO TO 90
C
 300  RIPDST=VAL-1
C-----EXIT
 900  CONTINUE
      RETURN
      END
```

```
C
C-------------------------------------------------------------
C
C-----SUBROUTINE RIPERR(NUM)
C-----SUBROUTINE TO REPORT PARAMETER ERRORS IN THE CALL
C     TO THE RIP RANDOM INTEGER PROGRAM GENERATOR.
C        NUM=THE ERROR NUMBER DETECTED EXCEPT THAT NUM=0
C           IMPLIES SUM UP
C
      SUBROUTINE RIPERR(NUM)
C
```

```
 420  IF(NOW.LE. )GO TO 9
      KKK=MATRIX(1,NOW)
      RIPOUT=RIPOUT*EXPAND(KKK)*MATRIX(2,NOW)
      NOW=MATRIX(3,NOW)
      GO TO 420
C
C-----EXIT
 900  CONTINUE
      RETURN
      END
```

```
C
C-------------------------------------------------------------
C
C-----INTEGER FUNCTION RIPOST(LIMIT,GENR)
C-----FUNCTION TO RETURN A OBSERVATION FROM THE SPECIAL RIP
C     RANDOM X-SPACE DISTANCE GENERATOR.  THE VALUE RETURNED
C     FALLS BETWEEN 0 AND THE PARAMETER LIMIT.  EACH UNIT OF
C     DISTANCE AWAY FROM 1 HAS DISRAT (A USER-SPECIFIED
C     PARAMETER EXPRESSED IN 1/10 PERCENT) TIMES THE
C     PROBABILITY OF THE PREVIOUS UNIT. GENR PICKS THE
C        RANDOM
C   . SEED PAIR USED.
C
      INTEGER FUNCTION RIPOST(LIMIT,GENR)
C
      IMPLICIT INTEGER (A-Y)
C
      COMMON /RIPCOM/SOLBAR,SOLSTR,TOLMOV,TOLOFL,TOLOPT,
     *   TOLSTP,TOLVER,BCONT,BDCONT,BDINT,BDSLK,BFRAC,
     *   BINT,BSLK,BTITST,COREOT,CORTOP,DATTYP,DELTM,
     *   DENSA,DENSF,DENSP,DENSPQ,DENSQ,DETERM,
     *   DIFDEG,DIMFRM,DISRAT,ECONT,EFRAC,EINT,ESLK,
     *   FRAC1,INFIN,LIMA,LIMALP,LIMD,LIMF,LIMR,LIMSLK,
     *   LIMVMN,LIMVMX,LIMWMN,LIMWMX,LUPR,LVLFRT,
     *   MAXCON,MAXCUT,MAXINT,MAXL1,MMM,MU,
     *   NNN,NNNP,NNNQ,NNNS,NNNT,NUMERR,NXTMAT,PRECN,PSLK,
     *   RIPSLK,SEED(2,7),SSLK,TOLMU,VERIFY,VERSN
C
      DOUBLE PRECISION SOLBAR,SOLSTR,TOLMOV,TOLOFL,TOLOPT,
     *   TOLSTP,TOLVER
      DOUBLE PRECISION WEIGHT,TERM,RATIO,CUMM
C
C
C-----CHECK IF THERE ARE OPTIONS
```

```
C
C-----INTEGER FUNCTION RIPEQ(COEF0,COEF1,COEF2,SOL2,GCD)
C-----FUNCTION TO SOLVE IN INTEGERS THE EQUATION
C         COEF0=COEF1*SOL1+COEF2*SOL2
C     USING THE EUCLIDEAN ALGORITHM FOR THE GREATEST
C     COMMON DIVISOR (GCD) OF COEF1 AND COEF2.  AFTER
C     A SPECIFIC SOLUTION IS OBTAINED, THE ONE WITH SMALLEST
C     POSITIVE SOL1 IS DETERMINED.  THE SOL1 OF THAT
C     SOLUTION IS RETURNED AS RIPEQ.
C
      INTEGER FUNCTION RIPEQG(COEF0,COEF1,COEF2,SOL2,GCD)
C
      IMPLICIT INTEGER (A-Y)
C
      COMMON /RIPCOM/SOLBAR,SOLSTR,TOLMOV,TOLCFL,TOLOPT,
     *   TOLSTP,TOLVER,BCONT,BDCONT,BDINT,BDSLK,BFRAC,
     *   BINT,BSLK,BTITST,CORBOT,CORTOP,DATTYP,DELTM,
     *   DENSA,DENSF,DENSP,DENSPQ,DENSQ,DETERM,
     *   DIFDEG,DIMERM,DISMAT,ECONT,EFRAC,EINT,ESLK,
     *   FRAC1,INFIN,LINA,LIMALP,LIMD,LIMF,LIMR,LIMSLK,
     *   LIMVAR,LIMVMX,LIMWAR,LIMWMX,LUPR,LVLPRT,
     *   MAXCON,MAXCUT,MAXINT,MAXT1,MMM,MU,
     *   NNN,NNNP,NNNQ,NNNS,NNNT,NUMERR,NXTMAT,PRECN,PSLK,
     *   RTPSLK,SEED(2,7),SSLK,TOLMU,VERIFY,VERSN
C
      DOUBLE PRECISION SOLBAR,SOLSTR,TOLMOV,TOLCFL,TOLOPT,
     *   TOLSTP,TOLVER
C
      DIMENSION EUCMAT(2,3)
C
C-----LOAD WORK MATRIX
      DO 1500 ROW=1,2
      DO 1500 COL=2,3
      EUCMAT(ROW,COL)=0
      IF(ROW+1.EQ.COL)EUCMAT(ROW,COL)=1
 1500 CONTINUE
      EUCMAT(1,1)=IABS(COEF1)
      EUCMAT(2,1)=IABS(COEF2)
C-----PICK THE OPERATOR ROW
 1700 OPRTOR=1
      IF(EUCMAT(OPRTOR,1).EQ.0)OPRTOR=OPRTOR+1
      IF(EUCMAT(OPRTOR,1).GT.EUCMAT(2,1).AND.EUCMAT(2,1)
     *   .NE.0)
     *   OPRTOR=2
C-----PICK OPERAND ROW
      OPRAND=OPRTOR+1
      IF(OPRAND.GT.2)OPRAND=1
C-----CHECK FOR TERMINATION
      IF(EUCMAT(OPRAND,1).EQ.0)GO TO 3000
C
      IMPLICIT INTEGER (A-Y)
C
      COMMON /RIPCOM/SOLBAR,SOLSTR,TOLMOV,TOLCFL,TOLOPT,
     *   TOLSTP,TOLVER,BCONT,BDCONT,BDINT,BDSLK,BFRAC,
     *   BINT,BSLK,BTITST,CORBOT,CORTOP,DATTYP,DELTM,
     *   DENSA,DENSF,DENSP,DENSPQ,DENSQ,DETERM,
     *   DIFDEG,DIMERM,DISMAT,ECONT,EFRAC,EINT,ESLK,
     *   FRAC1,INFIN,LINA,LIMALP,LIMD,LIMF,LIMR,LIMSLK,
     *   LIMVAR,LIMVMX,LIMWAR,LIMWMX,LUPR,LVLPRT,
     *   MAXCON,MAXCUT,MAXINT,MAXT1,MMM,MU,
     *   NNN,NNNP,NNNQ,NNNS,NNNT,NUMERR,NXTMAT,PRECN,PSLK,
     *   RTPSLK,SEED(2,7),SSLK,TOLMU,VERIFY,VERSN
C
      DOUBLE PRECISION SOLBAR,SOLSTR,TOLMOV,TOLCFL,TOLOPT,
     *   TOLSTP,TOLVER
C
C
C-----DETERMINE WHETHER IS THE SUM UP CALL
      IF(NUM.GT.0) GO TO 2000
      IF(NUMERR.GE.0)GO TO 9010
 1500 FORMAT(//25H ***RUN TERMINATED DUE TO ,I5,17H
     *   PARAMETER ERRORS/
     *   1H1)
      WRITE(LUPR,1500)NUMERR
      STOP 2
C-----CASE OF A PARAMETER RANGE ERROR
 2000 IF(NUM.GE.900)GO TO 3000
      NUMERR=NUMERR+1
 2100 FORMAT(18H RIP PARAMETER NO.,I4,24H OUTSIDE ALLOWABLE
     *   RANGE)
      WRITE(LUPR,2100)NUM
      GO TO 9000
C-----CASE OF A PARAMETER RELATIONSHIP ERROR
 3000 NUMERR=NUMERR+1
 3100 FORMAT(27H RIP PARAMETER PROPERTY NO.,I5,
     *   15H DOES NOT CHECK)
      WRITE(LUPR,3100)NUM
C
C-----EXIT
 9000 CONTINUE
      RETURN
      END
C
C-------------------------------------------------------------
```

```
C     VALUE IS NOT.  GENR SELECTS THE RANDOM SEED PAIR
C     USED
C
      SUBROUTINE RIPNU(XSTAR,DDD,NU,BNDWAS,GENR)
C
      IMPLICIT INTEGER (A-Y)
C
      COMMON /RIPCOM/SOLWAS,ZCLSTR,TOLMOV,TOLOFL,TOLOPT,
     *   TOLSTP,TOLVER,BCONT,BGCONT,BDINT,BDSLK,BFRAC,
     *   BINT,BSLK,RTITST,CURBOT,CURTOP,DATTYP,DELTM,
     *   DENSA,DENSF,DENSP,DENSPQ,DENSQ,DETERM,
     *   DIFDEG,DIFFRM,DISRAT,ECONT,EFRAC,EINT,ESLK,
     *   FRACI,INFIN,LIMA,LIMALP,LIMD,LIMF,LIMR,LIMSLK,
     *   LIMVMN,LIMVMX,LIMWMN,LIMWMX,LUPR,LVLPRT,
     *   MAXCON,MAXCUT,MAXINT,MAX 1,MMM,MU,
     *   NNN,NNNP,NNNQ,NNNS,NNNT,NUMERR,NXTMAT,PRECN,PSLK,
     *   RIPSLK,SEED(2,7),SSLK,TOLMU,VERIFY,VERSN
C
      DOUBLE PRECISION SOLBAR,SOLSTR,TOLMOV,TOLOFL,TOLOPT,
     *   TOLSTP,TOLVER
C
C-----COMPUTE PRELIMINARY DDD AND XSTAR
      DDD=RIPUNF(1,LIMD,GENR)
      DST=RIPDST(1,DDD-1,GENR)
      DST=MAXO(1,DST)
      XSTAR=DST
      IF(BNDWAS.EQ.1)XSTAR=DDD-DST
C-----SET NU IF DDD IS DIVISIBLE
      NU=RIPUNF(1,DDD/2+1,GENR)
      IF(NU.EQ.1)GO TO 9000
C-----MAKE DDD A MULTIPLE OF NU
      DDD=DDD*NU
C-----MAKE SURE XSTAR IS NOT A MULTIPLE OF NU
      XSTAR=XSTAR*NU+RIPUNF(1,NU-1,GENR)
      IF(XSTAR.GE.DDD)XSTAR=XSTAR-NU
C
C-----EXIT
 9000 RETURN
      END
C
C-------------------------------------------------------------
C
C-----SUBROUTINE RIPORG(MAT#3,MAT#2,ACOL,ACUT)
C-----SUBROUTINE TO REORGANIZE MEMORY AT THE END OF THE
C-----PERFORM A MAIN ITERATION OF THE EUCLIDEAN ALGORITHM
      QUOT=EUCMAT(OPRAND,1)/EUCMAT(OPRTOR,1)
      DO 2500 COL=1,3
      EUCMAT(OPRAND,COL)=EUCMAT(OPRAND,COL)-
     *   QUOT*EUCMAT(OPRTOR,COL)
 2500 CONTINUE
C
      GO TO 1700
C
C-----SET THE FIRST SPECIFIC SOLUTION
 3000 GCD=EUCMAT(OPRTOR,1)
      SOL1=EUCMAT(OPRTOR,2)*COEF1/IABS(COEF1)
      SOL2=EUCMAT(OPRTOR,3)*COEF2/IABS(COEF2)
C-----ADJUST SOLUTION FOR RHS=A MULTIPLE OF GCD
      SOL1=SOL1*COEF3/GCD
      SOL2=SOL2*COEF3/GCD
C
C-----REVISE TO THE SOLUTION WITH SMALLEST 1ST COMPONENT
C    ,GT.,
      TMULT=SOL1/COEF2
      IF(SOL1)3500,3600,3700
 3500 TMULT=TMULT-COEF2/IABS(COEF2)
      GO TO 4000
 3600 TMULT=-COEF2/IABS(COEF2)
      GO TO 4000
 3700 IF(TMULT*COEF2.EQ.SOL1)TMULT=TMULT-COEF2/IABS(COEF2)
C
 4000 SOL1=SOL1-TMULT*COEF2
      SOL2=SOL2+TMULT*COEF1
      RIPEUC=SOL1
C
C-----EXIT
 9000 CONTINUE
      RETURN
      END
C
C-------------------------------------------------------------
C    -
C
C-----SUBROUTINE RIPNU(XSTAR,DDD,NU,BNDWAS,GENR)
C
C-----SUBROUTINE TO GENERATE XSTAR,DDD AND NU VALUES ON
C     CONTINUOUS VARIABLES. BNDWAS=0 IF THE 0 VALUE
C     IS NOT ALLOWED FOR XSTAR, AND BNDWAS=1 IF THE DDD
```

```
      IF(NXT.GE.0)GO TO 43.
C-----NOW 3-TUPLE WAS PREVIOUSLY MOVED DOWN
      NOW=-NXT
      GO TO 42.
C-----NOW ENTRY ACTUALLY EXISTS
 43.0 III=MATR3(1,NOW)
      COEF=MATR3(2,NOW)
C-----CHECK IF RELOCATION SPACE IS FREE
      LAST=LAST+1
      IF(MATR3(3,LAST).EQ.INFIN)GO TO 4410
C-----SPACE NOT EMPTY--MOVE ENTRIES TO NOW
      MATR3(1,NOW)=MATR3(1,LAST)
      MATR3(2,NOW)=MATR3(2,LAST)
      MATR3(3,NOW)=MATR3(3,LAST)
      MATR3(3,LAST)=-NOW
      GO TO 441.
C-----COPY NOW INTO LAST
 44.0 MATR3(3,NOW)=INFIN
 4410 MATR3(1,LAST)=III
      MATR3(2,LAST)=COEF
C-----ADVANCE TO NEXT COLUMN ENTRY
      IF(NXT.EQ.0)GO TO 4600
      NOW=NXT
      GO TO 4200
C-----SET ACOL TO LAST ENTRY
 4600 ACOL(JJJ)=LAST
C-----END COLUMN LOOP
 4900 CONTINUE
C
C-----CUT STORAGE REORGANIZATION LOOP
      DO 5900 JJJ=1,NNN
      NOW=ACUT(JJJ)
      IF(NOW.EQ.0)GO TO 5800
C-----LOOP THROUGH COLUMN ENTRIES
 5200 NXT=MATR3(3,NOW)
      IF(NXT.GE.0)GO TO 5300
C-----NOW 3-TUPLE WAS PREVIOUSLY MOVED DOWN
      NOW=-NXT
      GO TO 520.
C-----NOW ENTRY ACTUALLY EXISTS
 5300 III=MATR3(1,NOW)
      COEF=MATR3(2,NOW)
C-----CHECK IF RELOCATION SPACE IS FREE
      LAST=LAST+1
      IF(MATR3(3,LAST).EQ.INFIN)GO TO 5400
C-----SPACE NOT EMPTY--MOVE ENTRIES TO NOW
      MATR3(1,NOW)=MATR3(1,LAST)
      MATR3(2,NOW)=MATR3(2,LAST)
      MATR3(3,NOW)=MATR3(3,LAST)
      MATR3(3,LAST)=-NOW
```

```
C     SET THREE PHASES OF RIP RANDOM INTEGER PROGRAM
C     GENERATION.  MATR2 AND MATR3 ARE THE 2-TUPLE AND
C     3-TUPLE (I.E. LINKED AND UNLINKED) VERSIONS OF
C     MAIN MATRIX STORAGE, RESPECTIVELY.
C     ALL OTHER VARIABLE NAMES  ARE AS IN THE RARDIN AND
C     LIN PAPER ON RIP OR AS DEFINED IN RIPPH.
C
      SUBROUTINE RIPORG(MATR3,MATR2,ACOL,ACUT)
C
      IMPLICIT INTEGER (A-Y)
C
      COMMON /RIPCOM/SOLBAR,SOLSTR,TOLMOV,TOLOFL,TOLOPT,
     *   TOLSTP,TOLVER,BDCONT,BDCONT,BDINT,BDSLK,BFRAC,
     *   BINT,BSLK,BTITST,CORBOT,CORTOP,DATTYP,DELTM,
     *   DENSA,DENSF,DENSP,DENSPQ,DENSQ,DETERM,
     *   DIFDEG,DIMFRM,DISRAT,ECONT,EFRAC,EINT,ESLK,
     *   FRAC1,INFIN,LIMA,LIMALP,LIMD,LIMF,LIMR,LIMSLK,
     *   LIMVMN,LIMVMX,LIMWMN,LIMWMX,LUPR,LVLPRT,
     *   MAXCON,MAXCUT,MAXINT,MAXL1,MMM,MU,
     *   NAN,NNNP,NNN2,NNNS,NNNT,NUMERR,NXTMAT,PRECN,PSLK,
     *   RIPSLK,SEED(2,7),SSLK,TOLMU,VERIFY,VERSN
C
      DOUBLE PRECISION SOLBAR,SOLSTR,TOLMOV,TOLOFL,TOLOPT,
     *   TOLSTP,TOLVER
C
      DIMENSION MATR3(3,1),MATR2(2,1),ACOL(1),ACUT(1)
C
C        ***SECTION TO REORGANIZE MEMORY INTO UNLINKED
C           COLUMN
C              LISTS***
C
C-----TAG EMPTY 3-TUPLES IN THE MATRIX AREA WITH INFINITY
      MAX=CORTOP/3
      DO 3650 PSN=1,MAX
      MATR3(3,PSN)=INFIN
 3650 CONTINUE
      NXT=NXTMAT
 3100 NOW=NXT
      IF(NOW.EQ.0)GO TO 4050
      NXT=MATR3(3,NOW)
      MATR3(3,NOW)=INFIN
      GO TO 3100
C
C-----MAIN COLUMN REORGANIZATION LOOP
 4000 LAST=0
      DO 4900 JJJ=1,NNN
      NOW=ACOL(JJJ)
      IF(NOW.EQ.0)GO TO 4800
C-----LOOP THROUGH COLUMN ENTRIES
 4200 NXT=MATR3(3,NOW)
```

```
      GO TO 541
C-----COPY NOW INTO LAST
 5400 MATR3(3,NOW)=INFIN
 5410 MATR3(1,LAST)=III
      MATR3(2,LAST)=COEF
C-----ADVANCE TO NEXT COLUMN ENTRY
      IF(NXT.EQ.0)GO TO 5800
      NOW=NXT
      GO TO 5200
C-----SET  ACUT TO LAST ENTRY
 5800 ACUT(JJJ)=LAST
C-----END OUT COLUMN LOOP
 5900 CONTINUE
C
C-----PACK 3-TUPLES INTO 2-TUPLES
      DO 6900 PSN=1,LAST
      MATR2(1,PSN)=MATR3(1,PSN)
      MATR2(2,PSN)=MATR3(2,PSN)
 6900 CONTINUE
C
C-----SET CORTOP
      CORTOP=2*LAST
C
C         ***EXIT SECTION***
C
 9000 CONTINUE
      RETURN
      END



C
C-------------------------------------------------------------
C
C-----INTEGER FUNCTION RIPPRM(NUM)
C-----FUNCTION TO RETURN PRIME NUMBER (NUM) OF THE 1ST 100
C
      INTEGER FUNCTION RIPPRM(NUM)
C
      IMPLICIT INTEGER (A-Y)
C
      COMMON /RIPCOM/SOLHAR,SOLSTR,TOLMOV,TOLOFL,TOLOPT,
     *   TOLSTP,TOLVER,BCONT,BDCONT,BDINT,BDSLK,BFRAC,
     *   BINT,BSLK,BTITST,COREOF,CORTOP,DATTYP,DELTM,
     *   DENSA,DENSF,DENSP,DENSPQ,DENSQ,DETERM,
     *   DIFDEG,DIMPRM,DISRAT,FCONT,EFRAC,EINT,ESLK,
     *   FRAC1,INFIN,LIMA,LIMALP,LIMD,LIMF,LIMR,LIMSLK,
     *   LIMVMN,LIMVMX,LIMXMN,LIMXMX,LUPR,LVLERT,
     *   MAXCOL,MAXCUT,MAXINT,MAX1,MMM,MU,
     *   NRU,NUMP,NUMQ,NUMS,NUMT,NUMERR,NXTNAM,PRECN,PSLK,
     *   RIPSLK,SEED(2,7),SSLK,TOLMU,VERIFY,VERSN
C
      DOUBLE PRECISION SOLHAR,SOLSTR,TOLMOV,TOLOFL,TOLOPT,
     *   TOLSTP,TOLVER
C
      DIMENSION PRIME(100)
      DATA PRIME/
     * 2   ,3   ,5   ,7   ,11  ,13  ,17  ,19  ,23  ,29  ,
     * 31  ,37  ,41  ,43  ,47  ,53  ,59  ,61  ,67  ,71  ,
     * 73  ,79  ,83  ,89  ,97  ,101 ,103 ,107 ,109 ,113 ,
     * 127 ,131 ,137 ,139 ,149 ,151 ,157 ,163 ,167 ,173 ,
     * 179 ,181 ,191 ,193 ,197 ,199 ,211 ,223 ,227 ,229 ,
     * 233 ,239 ,241 ,251 ,257 ,263 ,269 ,271 ,277 ,281 ,
     * 283 ,293 ,307 ,311 ,313 ,317 ,331 ,337 ,347 ,349 ,
     * 353 ,359 ,367 ,373 ,379 ,383 ,389 ,397 ,401 ,409 ,
     * 419 ,421 ,431 ,433 ,439 ,443 ,449 ,457 ,461 ,463 ,
     * 467 ,479 ,487 ,491 ,499 ,503 ,509 ,521 ,523 ,541 /
C
C-----EXTRACT TABLE ENTRY
      RIPPRM=PRIME(NUM)
C-----EXIT
 9000 CONTINUE
      RETURN
      END



C
C-------------------------------------------------------------
C
C-----DOUBLE PRECISION FUNCTION RIPROT(MATRIX,STRT1,STRT2,
C    EXPAND,LENGTH)
C-----FUNCTION TO RETURN THE DOT PRODUCT OF TWO VECTORS,
C     AT LEAST ONE OF WHICH IS STORED IN THE GENERAL
C     RIP 3-TUPLE STROAGE AREA.  THE FIRST IS FLOATING POINT
C     THE 2ND IF INTEGER AND THE RESULT IS FLOATING POINT.
C     HOWEVER, THE 2ND MAY ALSO BE FULL PRECICION
C     DOUBLE PRECISION IF IT IS PRE-EXPANDED.
C     PARAMETERS ARE AS FOLLOWS--
C        MATRIX=THE GENERAL MATRIX STAORAGE AREA
C        STRT1=THE STARTING 3-TUPLE OF THE FIRST VECTOR
C           (=-1 IF THE VECTOR ALREADY EXISTS AS EXPAND)
C        STRT2=THE STARTING 3-TUPLE OF THE SECOND VECTOR
C           (=-1 IF THE VECTOR ALREADY EXIST AS EXPAND)
```

```
C        EXPAND=TH  WORK AR A INTO WHICH ONE VECTOR WILL
C           BE  XPAND D IF BOTH BELONG TO MATRIX
C        LENGTH=THE LENGTH OF THE VECTORS INVOLVED
C
      DOUBLE PR ISION FUNCTION RIPRDT(MATRIX,STRT1,STRT2,
     *   EXPAND,LENGTH)
C
      IMPLICIT INTEGER (A-Y)
C
      COMMON /RIPCOM/SOLBAR,SOLSTR,TOLMOV,TOLOFL,TOLOPT,
     *   TOLSTP,TOLVER,BCONT,BDCONT,BDINT,BDSLK,BFRAC,
     *   BINT,BSLK,BTITST,CORBOT,CORTOP,DATTYP,DELTM,
     *   DENSA,DENSF,DENSP,DENSPQ,DENSQ,DETERM,
     *   DIFDEG,DIMFRM,DISRAT,ECONT,EFRAC,EINT,ESLK,
     *   FRACI,INFIN,LIMA,LIMALP,LIMD,LIMF,LIMR,LIMSLK,
     *   LIMVMN,LIMVMX,LIMWMN,LIMWMX,LUPR,LVLPRT,
     *   MAXCON,MAXCUT,MAXINT,MAXTRY,MMM,MU,
     *   NNN,NNNP,NNNQ,NNNS,NNNT,NUMERR,NXTMAT,PRECN,PSLK,
     *   RTPSLK,SEED(2,7),SSLK,TOLMU,VERIFY,VERSN
C
      DOUBLE PRECISION SOLBAR,SOLSTR,TOLMOV,TOLOFL,TOLOPT,
     *   TOLSTP,TOLVER
C
      DOUBLE PRECISION EXPAND,RCOEF
      EQUIVALENCE (ZRELNO,INTGNO)
C
      DIMENSION EXPAND(1),MATRIX(3,1)
C
C-----CHECK FOR VACUOUS VECTOR 1
      IF(STRT2)2200,2900,2500
C-----1ST VECTOR IS PRE-EXPANDED
 2200 NOW=STRT2
      WORKER=2
      GO TO 4000
C-----CHECK FOR VACUOUS VECTOR 2
 2500 IF(STRT2)2600,3100,3000
C-----2ND VECTOR IS PRE-EXPANDED
 2600 NOW=STRT1
      WORKER=1
      GO TO 4000
C-----EXPAND VECTOR 1 IF NEITHER PRE-EXPANDED
 3000 DO 3200 KKK=1,LENGTH
      EXPAND(KKK)=0
 3200 CONTINUE
      NOW=STRT1
 3500 IF(NOW.LE.0)GO TO 3900
      KKK=MATRIX(1,NOW)
      INTGNO=MATRIX(2,NOW)
      EXPAND(KKK)=ZRELNO
      NOW=MATRIX(3,NOW)
      GO TO 3500
C-----MULTIPLY THE UNEXPANDED BY THE EXPANDED
 3900 NOW=STRT2
      WORKER=2
 4000 RIPRDT=0
 4200 IF(NOW.LE.0)GO TO 9000
      KKK=MATRIX(1,NOW)
      INTGNO=MATRIX(2,NOW)
      IF(WORKER.EQ.1)RCOEF=ZRELNO
      IF(WORKER.EQ.2)RCOEF=INTGNO
      RIPRDT=RIPRDT+EXPAND(KKK)*RCOEF
      NOW=MATRIX(3,NOW)
      GO TO 4200
C
C-----EXIT
 9000 CONTINUE
      RETURN
      END
C
C-------------------------------------------------------------
C
C-----SUBROUTINE RIPSEQ(LIST,MAXPSN,GENNUM,MAXPT)
C-----SUBROUTINE TO RANDOMIZE THE NUMBERS IN LIST.
C        LIST=THE VECTOR OF NUMBERS TO RANDOMIZE
C        MAXPSN=THE NUMBER OF ENTRIES IN LIST
C        GENNUM=THE RIP RANDOM NUMBER GENERATOR TO EMPLOY
C        MAXPT=THE VALUE OF THE LAST ENTRY IN THE LIST IF
C           THE SUBROUTINE IS TO CREATE THE INITIAL LIST.
C           IF MAXPT=0, THE LIST IS PRECREATED.
C
      SUBROUTINE RIPSEQ(LIST,MAXPSN,GENNUM,MAXPT)
C
      IMPLICIT INTEGER (A-Y)
C
      COMMON /RIPCOM/SOLBAR,SOLSTR,TOLMOV,TOLOFL,TOLOPT,
     *   TOLSTP,TOLVER,BCONT,BDCONT,BDINT,BDSLK,BFRAC,
     *   BINT,BSLK,BTITST,CORBOT,CORTOP,DATTYP,DELTM,
     *   DENSA,DENSF,DENSP,DENSPQ,DENSQ,DETERM,
     *   DIFDEG,DIMFRM,DISRAT,ECONT,EFRAC,EINT,ESLK,
     *   FRACI,INFIN,LIMA,LIMALP,LIMD,LIMF,LIMR,LIMSLK,
     *   LIMVMN,LIMVMX,LIMWMN,LIMWMX,LUPR,LVLPRT,
     *   MAXCON,MAXCUT,MAXINT,MAXTRY,MMM,MU,
     *   NNN,NNNP,NNNQ,NNNS,NNNT,NUMERR,NXTMAT,PRECN,PSLK,
     *   RTPSLK,SEED(2,7),SSLK,TOLMU,VERIFY,VERSN
```

```
C
      DOUBLE PRECISION SOLBAR,SOLSTR,TOLMOV,TCLOFL,TOLOPT,
     *   TOLSTP,TOLVER
C
      DIMENSION LIST(1)
C-----CHECK IF LIST PRELOADED
      IF(MAXPT.EQ..)GO TO 1...
C-----LOAD THE LIST WITH VALUES UP TO MAXPT
      OFFSET=MAXPT-MAXPSN
      DO 19.. PSN=1,MAXPSN
      LIST(PSN)=PSN+OFFSET
 19.. CONTINUE
C-----PERFORM MAXPSN RANDOM PAIR INTERCHANGES
 1...  DO 2... PSN=1,MAXPSN
      SWAP=RIPUNF(1,MAXPSN,GENNUM)
      HOLD=LIST(PSN)
      LIST(PSN)=LIST(SWAP)
      LIST(SWAP)=HOLD
 2... CONTINUE
C-----EXIT
 9... CONTINUE
      RETURN
      END
```

```
C
C-------------------------------------------------------------
C
C-----SUBROUTINE RIPSPL(SPLIT,MULTS,MINTOT,MAXTOT,MAXVAL,
C         NUMSPL,RNDSEQ,GENNUM,DENS,BREAK)
C-----SUBROUTINE TO RANDOMLY PLACE IN THE VECTOR SPLIT
C   INTEGERS
C      WHICH, WHEN WEIGHTED BY THE VECTOR MULTS, HAVE
C      A TOTAL BETWEEN MINVAL AND MAXVAL. PARAMETERS ARE AS
C         FOLLOWS--
C         SPLIT=THE OUTPUT VECTOR
C         MULTS=THE INPUT WEIGHT VECTOR
C         MINTOT=THE MINIMUM ALLOWABLE TOTAL
C         MAXTOT=THE MAXIMUM ALLOWABLE TOTAL
C         NUMSPL=THE LENGTH OF THE SPLIT VECTOR
C         MAXVAL=THE MAXIMUM ABSOLUTE VALUE OF GENERATED
C            INTEGERS
C         RNDSEQ=A WORK VECTOR AREA OF SAME LENGTH AS OTHER
C            VECTORS
C         GENNUM=THE RIP RANDOM NUMBER GENERATOR TO BE
C            EMPLOYED
C         DENS=THE DESIRED DENSITY ON NONZERO ENTRIES IN THE
C            OUTPUT
C            (.1. PERCENT)
C         BREAK=THE POINT AT WHICH OUTPUT ENTRIES CHANGE
C            SIGN.
C            IF BREAK=. ALL ENTRIES ARE OF MIXED SIGN
C            IF BREAK.LT.. ENTRIES UP TO IABS(BREAK) ARE
C               NONPOSITIVE
C               AND THOSE AFTERWARD ARE NONNEGATIVE
C            IF BREAK.GT.. ENTRIES UP TO BREAK ARE
C               NONNEGATIVE
C               AND THOSE AFTERWARD ARE NONPOSITIVE
C
      SUBROUTINE RIPSPL(SPLIT,MULTS,MINTOT,MAXTOT,MAXVAL,
     *   NUMSPL,RNDSEQ,GENNUM,DENS,BREAK)
C
      IMPLICIT INTEGER (A-Y)
C
      COMMON /RIPCOM/SOLBAR,SOLSTR,TOLMOV,TOLOFL,TOLOPT,
     *   TOLSTP,TOLVER,BCONT,BDCONT,BOINT,BDSLK,BFRAC,
     *   BINT,BSLK,BTITST,CORBOT,CORTOP,DATTYP,DELTM,
     *   DENSA,DENSF,DENSP,DENSPQ,DENSQ,DETERM,
     *   DIFDEG,DIMERM,DISKAT,ECONT,EFRAC,EINT,ESLK,
     *   FRAC1,INFIN,LIMA,LIMALP,LIMD,LIMF,LIPR,LIMSLK,
     *   LIMVMN,LIMVMX,LIMWMN,LIMWMX,LUPR,LVLPRT,
     *   MAXCON,MAXCUT,MAXINT,MAX.1,NMM,MU,
     *   NNN,NNNP,NNNQ,NNNS,NNNT,NUMERR,NXTMAT,PRECN,PSLK,
     *   RIPSLK,SEED(2,7),SSLK,TOLMU,VERIFY,VERSN
C
      DOUBLE PRECISION SOLBAR,SOLSTR,TOLMOV,TCLOFL,TOLOPT,
     *   TOLSTP,TOLVER
      DOUBLE PRECISION RWORK,MINREM,MAXREM,ABSVAL,MINV,MAXV,
     *   TEMMIN,
     *   TEMMAX,ADJ
C
      DIMENSION SPLIT(1),MULTS(1),RNDSEQ(1)
C
C-----INITIALIZE CONTROL VALUES
      ABSBRK=IABS(BREAK)
      ABSVAL=MAXVAL
      NONZER=0
C-----CALCULATE MINIMUM AND MAXIMUM POSSIBLE TOTALS
 2... IF(BREAK)2..1.,2..2.,2..3.
C-----NEGATIVES 1ST
 2..1. MINV=-ABSVAL
      MAXV=.
      ADJ=ABSVAL
      GO TO 2..5.
C-----MIXED SIGNS OK
 2..2. MINV=-ABSVAL
```

```
      SPL=RNDSEQ(PSN)
      TEMMAX=MAXV
      TEMMIN=MINV
      IF(SPL.LE.ABSBRK.OR.ABSBRK.EQ.0)GO TO 3090
      TEMMAX=TEMMAX+ADJ
      TEMMIN=TEMMIN+ADJ
C-----REMOVE VARIABLE FROM RUNNING TOTALS
 3090 IF(MULTS(SPL))3100,3300,3200
C-----NEGATIVE MULTIPLIER
 3100 MINREM=MINREM-MULTS(SPL)*TEMMAX
      MAXREM=MAXREM-MULTS(SPL)*TEMMIN
      CURMAX=(MAXTOT-CURTOT-MAXREM)/MULTS(SPL)
      CURMIN=(MINTOT-CURTOT-MINREM)/MULTS(SPL)
      GO TO 3500
C-----POSITIVE MULTIPLIER
 3200 MINREM=MINREM-MULTS(SPL)*TEMMIN
      MAXREM=MAXREM-MULTS(SPL)*TEMMAX
      CURMIN=(MAXTOT-CURTOT-MAXREM)/MULTS(SPL)
      CURMAX=(MINTOT-CURTOT-MINREM)/MULTS(SPL)
      GO TO 3500
C-----ZERO MULTIPLIER
 3300 IF(RIPUNF(0,1.0,GENNUM).LE.DENS.AND.NONZER.EQ.1)GO
     *    TO 3400
C-----CREATE A ZERO ENTRY
 3350 SPLIT(SPL)=0
      GO TO 3900
C-----CREATE A NONZERO ENTRY
 3400 SPLIT(SPL)=RIPUNF(TEMMIN,TEMMAX,GENNUM)
      IF(SPLIT(SPL).EQ.0)GO TO 3400
      NONZER=1
      GO TO 3900
C-----CLOSE IN LIMITS AND SET VALUE ON NONZERO MULT
 3500 FIX=TEMMIN
      CURMIN=MAX0(CURMIN,FIX)
      FIX=TEMMAX
      CURMAX=MIN0(CURMAX,FIX)
C-----DECIDE IF ENTRY SHOULD BE ZERO
      IF(CURMIN.LE. .AND.CURMAX.GE.0.AND.RIPUNF(0,1.0,
     *    GENNUM).GT.DENS
     *    .AND.NONZER.EQ.1)GO TO 3350
 3600 SPLIT(SPL)=RIPUNF(CURMIN,CURMAX,GENNUM)
      IF(CURMIN.NE.CURMAX.AND.SPLIT(SPL).EQ.0)
     *    GO TO 3600
      IF(SPLIT(SPL).NE.0)NONZER=1
      CURTOT=CURTOT+SPLIT(SPL)*MULTS(SPL)
C-----END LOOP
 3900 CONTINUE
 3990 CONTINUE
C-----RESET LAST ENTRY TO HAVE SMALLEST FEASIBLE ABSOLUTE
C    VALUE
      MAXV=ABSVAL
      ADJ=0
      GO TO 2050
C-----POSITIVES 1ST
 2030 MINV=0
      MAXV=ABSVAL
      ADJ=-ABSVAL
C-----CALCULATE MINIMUM AND MAXIMUM POSSIBLE TOTALS
 2050 MINREM=0
      MAXREM=0
      DO 2060 SPL=1,NUMSPL
C-----CHANGE SIGNS IF NECESSARY
      IF(SPL.NE.ABSBRK+1.OR.ABSBRK.EQ.0)GO TO 2100
      MINV=MINV+ADJ
      MAXV=MAXV+ADJ
C-----DIVIDE ON MULTIPLIER SIGN
 2100 IF(MULTS(SPL))2150,2500,2200
C-----NEGATIVE MULTIPLIER
 2150 MINREM=MINREM+MULTS(SPL)*MAXV
      MAXREM=MAXREM+MULTS(SPL)*MINV
      GO TO 2500
C-----POSITIVE MULTIPLIER
 2200 MINREM=MINREM+MULTS(SPL)*MINV
      MAXREM=MAXREM+MULTS(SPL)*MAXV
 2500 CONTINUE
C-----CHECK IF FEASIBLE
      IF(MINREM.LE.MAXTOT.AND.MAXREM.GE.MINTOT)GO TO 2900
C-----EXPAND RANGE
      ABSVAL=2*ABSVAL
      GO TO 2000
C-----RANDOMIZE ASSIGNMENT SEQUENCE
 2900 CALL RIPSEQ(RNDSEQ,NUMSPL,GENNUM,NUMSPL)
C-----ASSURE LAST ENTRY IS UNITY OR POSITIVE
      LAST=RNDSEQ(NUMSPL)
      IF(IABS(MULTS(LAST)).EQ.1)GO TO 3000
      DO 2950 PSN=1,NUMSPL
      SPL=RNDSEQ(PSN)
      IF(IABS(MULTS(SPL)).LT.1.OR.IABS(MULTS(SPL)).GT.
     *    IABS(MULTS(LAST)))GO TO 2950
      RNDSEQ(PSN)=RNDSEQ(NUMSPL)
      RNDSEQ(NUMSPL)=SPL
      IF(IABS(MULTS(SPL)).EQ.1)GO TO 3000
      LAST=SPL
 2950 CONTINUE
C-----MAIN LOOP TO SPLIT THE TOTAL
 3000 CURTOT=0
      IF(NUMSPL.LE.ABSBRK.OR.ABSBRK.EQ.0)GO TO 3050
      MINV=MINV-ADJ
      MAXV=MAXV-ADJ
 3050 DO 3990 PSN=1,NUMSPL
```

```
      DOUBLE PRECISION SOLBAR,SOLSTR,TOLMOV,TOLOFL,TOLOPT,
     *   TOLSTP,TOLVER
C
      DIMENSION VBLTYP(1),JLIST(1),SUBM(1),AVAIL(3)
C
C-----LOOP THROUGH SUBMATRIX POSITIONS
      DO 390 PSN=MINPSN,MAXPSN
C-----ASSIGN A JJJ TO THE SUBMATRIX POSITION
      JPSN=JPSN+1
      JJJ=JLIST(JPSN)
      SUBM(PSN)=JJJ
      NOWAV=0
      IF(NUMCON.GE.MAXCON)GO TO 292
C-----SOME CONTINUOUS VARIABLES ARE AVAILABLE FOR ASSIGNMENT
      NOWAV=NOWAV+1
      AVAIL(NOWAV)=3
 292  IF(NUMINT.GE.MAXINT)GO TO 294
C-----SOME GENERAL INTEGER VARIABLES ARE AVAILABLE FOR
C   ASSIGNMENT
      NOWAV=NOWAV+1
      AVAIL(NOWAV)=4
 294  IF(NUM01.GE.MAX 1)GO TO 300
C-----SOME 0-1 VARIABLES ARE AVAILABLE FOR ASSIGNMENT
      NOWAV=NOWAV+1
      AVAIL(NOWAV)=5
C-----PICK THE TYPE FOR THIS POSITION
 300  SUB=IPUNF(1,NOWAV,1)
      TYP=AVAIL(SUB)
      VBLTYP(JJJ)=TYP
      GO TO (390,390,310,320,330),TYP
 310  IF(NUMCON.NE.INFIN)NUMCON=NUMCON+1
      GO TO 390
 320  IF(NUMINT.NE.INFIN)NUMINT=NUMINT+1
      GO TO 390
 330  IF(NUM01.NE.INFIN)NUM01=NUM01+1
 390  CONTINUE
C
C-----EXIT
 900  CONTINUE
      RETURN
      END
```

```
C
C-------------------------------------------------------------
C
```

```
      CURTOT=CURTOT-SPLIT(SPL)*MULTS(SPL)
      IF(CURMAX.GT.0) GO TO 420
      SPLIT(SPL)=CURMAX
      GO TO 450
 420  IF(CURMIN.LT.0) GO TO 430
      SPLIT(SPL)=CURMIN
      GO TO 450
 430  SPLIT(SPL)=0
 450  CURTOT=CURTOT+SPLIT(SPL)*MULTS(SPL)
C-----EXIT
 900  CONTINUE
      RETURN
      END
```

```
C
C-------------------------------------------------------------
C
C-----SUBROUTINE RIFTYP(VBLTYP,JLIST,JPSN,SUBM,MINPSN,
C    MAXPSN,
C     *   NUM01,NUMINT,NUMCON)
C-----SUBROUTINE TO ALLOCATE COLUMNS RANDOMLY ARRANGED IN
C    THE
C      LIST JLIST INTO COMPONENTS MINPSN TO MAXPSN OF
C         SUBMATRIX
C      LIST SUBM.  NUM01,NUMINT AND NUMCON ARE RESPECTIVELY
C      THE NUMBERS OF 0-1, GENERAL INTEGER AND CONTINUOUS
C      VARIABLES ALREADY ASSIGNED.
C
      SUBROUTINE RIFTYP(VBLTYP,JLIST,JPSN,SUBM,MINPSN,
     *   MAXPSN,
     *   NUM01,NUMINT,NUMCON)
C
      IMPLICIT INTEGER (A-Y)
C
      COMMON /RIPCOM/SOLBAR,SOLSTR,TOLMOV,TOLOFL,TOLOPT,
     *   TOLSTP,TOLVER,BCONT,EDCONT,BDINT,BDSLK,BFRAC,
     *   BINT,BSLK,BTITST,CORBOT,CORTOP,DATTYP,DELTM,
     *   DENSA,DENSF,DENSP,DENSPQ,DENSQ,DETERM,
     *   DIFOLG,DIMPRM,DISRAT,ECONT,EFRAC,EINT,ESLK,
     *   FRAC1,INFIN,LIMA,LIMALP,LIMD,LIMF,LIMR,LIMSLK,
     *   LIMVMN,LIMVMX,LIMWMN,LIMWMX,LUPR,LVLPRT,
     *   MAXCON,MAXCUT,MAXINT,MAX 1,MMM,MU,
     *   NNN,NNNP,NNNQ,NNNS,NNNT,NUMERR,NXTMAT,PRECN,PSLK,
     *   RIPSLK,SEED(2,7),SSLK,TOLMU,VERIFY,VERSN
C
```

```
C-----INTEGER FUNCTION RIPUNF(LEFT,RIGHT,GENNUM)
C-----FUNCTION TO GENERATE A UNIFORM (LEFT,RIGHT) INTEGER
C     USING RIP RANDOM NUMBER SEED PAIR NUMBER GENNUM
C
C-----THE METHOD USES A MULTIPLICATIVE CONGRUENTIAL APPROACH
C     WITH MULTIPLIER 69069 AND MOD=2**32.  ARITHMETIC ON
C     SEEDS IS DONE BASE 2**16=65536 TO AVOID OVERFLOW AND
C     MAKE THE PROCESS PORTABLE BETWEEN SCIENTIFIC MACHINES.
C     SEED(1,GENNUM) AND SEED(2,GENNUM) ARE THE LEFT AND
C        RIGHT
C     BASE 2**16 DIGITS.
C
      INTEGER FUNCTION RIPUNF(LEFT,RIGHT,GENNUM)
C
      IMPLICIT INTEGER (A-Y)
C
      COMMON /RIPCOM/SOLBAR,SOLSTR,TOLMOV,TOLOFL,TOLOPT,
     *   TOLSTP,TOLVER,BCONT,BDCONT,BDINT,BDSLK,BFRAC,
     *   BINT,BSLK,BTITST,CORBOT,CORTOP,DATTYP,DELTM,
     *   DENBAR,DENSF,DENSP,DENSPQ,DENSQ,DETERM,
     *   DIFDEG,DIMPRM,DISRAT,ECONT,EFRAC,EINT,ESLK,
     *   FRAC1,INFIN,LIMA,LIMALP,LIMD,LIMF,LIMR,LIMSLK,
     *   LIMVMN,LIMVMX,LIMWMN,LIMWMX,LUPR,LVLPRT,
     *   MAXCON,MAXCUT,MAXINT,MAX1,MMM,MU,
     *   NNN,NNNP,NNNQ,NNNS,NNNT,NUMERR,NXTMAT,PRECN,PSLK,
     *   RIPSLK,SEED(2,7),SSLK,TOLMU,VERIFY,VERSN
C
      DOUBLE PRECISION SOLBAR,SOLSTR,TOLMOV,TOLOFL,TOLOPT,
     *   TOLSTP,TOLVER
      DOUBLE PRECISION RANGE
C
C-----MULTIPLY SEED BY 69069 = 1*65536 + 3533*1 AND MOD
C   2**32
      WORK1=3533*SEED(1,GENNUM)
      WORK2=3533*SEED(2,GENNUM)
      SEED(1,GENNUM)=MOD(WORK2/65536+SEED(2,GENNUM)+MOD(WORK1,
     *   65536)
      SEED(1,GENNUM)=MOD(SEED(1,GENNUM),65536)
      SEED(2,GENNUM)=MOD(WORK2,65536)
C-----SET UNIFORM VALUE USING ONLY THE LEFT 16 BITS OF SEED
      RANGE=RIGHT-LEFT+1
      RIPUNF=SEED(1,GENNUM)*RANGE/65536
      RIPUNF=RIPUNF+LEFT
C-----EXIT
 9000 CONTINUE
      RETURN
      END
C
C-----------------------------------------------------------
C
C-----SUBROUTINE RIPVER(MATRIX,ZATRIX,RHS,CCC,DDD,UBAR,
C    USTAR,XBAR,
C     *   XSTAR,ROWTYP,VBLTYP,VSTAR,ACOL,ACUT,BBB,NU,SCRCH,
C     *   DENBAR,DENSTR)
C-----SUBROUTINE TO VERIFY KUHN-TUCKER OPTIMALITY CONDITIONS
C     FOR THE LINEAR RELAXATION AND THE INTEGER SOLUTIONS OF
C     PROBLEMS GENERATED BY THE RIP RANDOM INTEGER PROGRAM
C     GENERATOR.  ALL PARAMETER NAMES  ARE AS IN THE RARDIN
C     AND LIN PAPER ON THE RIP GENERATOR, OR AS DEFINED IN
C     SUBROUTINE RIPPH.
C
      SUBROUTINE RIPVER(MATRIX,ZATRIX,RHS,CCC,DDD,UBAR,
     *   USTAR,XBAR,
     *   XSTAR,ROWTYP,VBLTYP,VSTAR,ACOL,ACUT,BBB,NU,SCRCH,
     *   DENBAR,DENSTR)
C
      IMPLICIT INTEGER (A-Y)
C
      COMMON /RIPCOM/SOLBAR,SOLSTR,TOLMOV,TOLOFL,TOLOPT,
     *   TOLSTP,TOLVER,BCONT,BDCONT,BDINT,BDSLK,BFRAC,
     *   BINT,BSLK,BTITST,CORBOT,CORTOP,DATTYP,DELTM,
     *   DENBAR,DENSF,DENSP,DENSPQ,DENSQ,DETERM,
     *   DIFDEG,DIMPRM,DISRAT,ECONT,EFRAC,EINT,ESLK,
     *   FRAC1,INFIN,LIMA,LIMALP,LIMD,LIMF,LIMR,LIMSLK,
     *   LIMVMN,LIMVMX,LIMWMN,LIMWMX,LUPR,LVLPRT,
     *   MAXCON,MAXCUT,MAXINT,MAX1,MMM,MU,
     *   NNN,NNNP,NNNQ,NNNS,NNNT,NUMERR,NXTMAT,PRECN,PSLK,
     *   RIPSLK,SEED(2,7),SSLK,TOLMU,VERIFY,VERSN
C
      DOUBLE PRECISION SOLBAR,SOLSTR,TOLMOV,TOLOFL,TOLOPT,
     *   TOLSTP,TOLVER
C
      DIMENSION MATRIX(2,1),RHS(1),CCC(1),DDD(1),UBAR(1),
     *   USTAR(1),XBAR(1),XSTAR(1),ROWTYP(1),VBLTYP(1),
     *   ACOL(1),BBB(1),ACUT(1),SCRCH(1),VSTAR(1),NU(1),
     *      ZATRIX(2,1)
C
      DOUBLE PRECISION CCC,UBAR,USTAR,VSTAR,ADJCST,RELXB,
     *   RCOEF,RHS,
     *   SOLVAL,SCRCH,TOLSUM,RELXS
C
C-----SET TOLERANCES
      TOLVER=.01
      TOLSUM=100*TOLVER
      IF(TOLSUM.GT.NNN*TOLVER)TOLSUM=NNN*TOLVER
C
C-----PRINT VERIFICATION HEADER
```

```
 15..  FORMAT(1H1/2..X,4(7H*******)
     *    /2.X,2H* ,34H-IP PROBLEM VERIFICATION,2H *
     *    /2.X,4(7H*******)/1X)
       WRITE(LUPR,15..)
C
C         ***SECTION TO VERIFY THE LINEAR RELAXATION (BAR)
C            SOLUTION***
C
       SOLVAL=.
C-----CLEAR RONDUM VECTOR
       DO 22.. III=1,MMM
       SCRCH(III)=-RHS(III)
 22.. CONTINUE
C
C-----LOOP THROUGH COLUMNS
       DO 39.. JJJ=1,NNN
C-----CHECK UPPER BOUND VALUE
       IF(VBLTYP(JJJ).NE.5.AND.DDD(JJJ).GT.0)GO TO 31..
       IF(DDD(JJJ).EQ.1)GO TO 31..
 3..  FORMAT(/23H **DDD  VERIF FAILURE**,8HVARIABLE,
     *    I5,7H BOUND=,I15/1..X,24HWRONG FOR VARIABLE TYPE=,
     *    I3)
       WRITE(LUPR,3..)JJJ,DDD(JJJ),VBLTYP(JJJ)
C-----CLEAR ADJUSTED COST
 31.. ADJCST=.
       SOLVAL=SOLVAL+COC(JJJ)*XBAR(JJJ)/DENBAR
C-----LOOP THROUGH COLUMN ELEMENTS
       MIN=1
       IF(JJJ.GT.1)MIN=ACOL(JJJ-1)+1
       MAX=ACOL(JJJ)
       IF(MAX.GE.MIN)GO TO 315.
 311.  FORMAT(/23H **AAA  VERIF FAILURE**,7H COLUMN,I5,
     *    11H IS VACUOUS)
       WRITE(LUPR,3110)JJJ
       GO TO 3..3
 315.  DO 3..2 PSN=MIN,MAX
       III=MATRIX(1,PSN)
       COEF=MATRIX(2,PSN)
       SCRCH(III)=SCRCH(III)+COEF*XBAR(JJJ)/DENBAR
       ADJCST=ADJCST-UBAR(III)*COEF
 3..2 CONTINUE
 39.. CONTINUE
C
C-----CHECK ADJUSTED COST
       IF(ADJCST.GT.1..OR.ADJCST.LT.-1..AND.DATTYP.EQ.0)
     *    GO TO 35..
 345.  FORMAT(/23H **AAA  VERIF FAILURE**,12HCOEFFICIENT=,
     *    I15,6H AT A(,I5,1H,,I5,1H)/1.X,
     *    24HWRONG FOR DATA TYPE=,I2)
       WRITE(LUPR,345.)COEF,III,JJJ,DATTYP
 35.. RCOEF=COEF
       SCRCH(III)=SCRCH(III)+RCOEF*XBAR(JJJ)/DENBAR
       ADJCST=ADJCST-UBAR(III)*COEF
 3..2 CONTINUE
C
```

```
 42.0  IF(UBAR(III).GE.-TOLVER)GO TO 425
 4220  WRITE(LUPR,4230)III,UBAR(III),ROWTYP(III)
 4230  FORMAT(/23H **BAR  VERIF FAILURE**,4H ROW,I5,
     *    8H DUAL OF,E12.5,12H WRONG FOR ROW TYPE=,I3)
       GO TO 490
C-----CHECK COMPLEMENTARITY
 4250  IF(SCRCH(III)*UBAR(III).GE.-TOLVER.AND.
     *    SCRCH(III)*UBAR(III).LE.TOLVER)GO TO 490
       WRITE(LUPR,4260)III,UBAR(III),SCRCH(III)
 4260  FORMAT(/23H **BAR  VERIF FAILURE**,4H ROW,I5,
     *    8H DUAL OF,E12.5,12H AND ROW SUM,E12.5
     *    /1X,19H NOT COMPLEMENTARY)
       GO TO 490
C-----.EQ. ROW
 4300  IF(SCRCH(III).LT.-TOLSUM.OR.SCRCH(III).GT.TOLSUM)GO
     *    TO 415
       GO TO 490
C-----.LE. ROW
 4500  IF(SCRCH(III).GT.TOLSUM)GO TO 415
       IF(UBAR(III).GT.TOLVER)GO TO 4220
       GO TO 425
C-----END ROW LOOP
 4900  CONTINUE
C
C-----CHECK SOLBAR VALUE
       IF(SOLVAL.LE.SOLBAR+TOLSUM.AND.SOLVAL.GE.
     *    SOLBAR-TOLSUM)GO TO 5000
 4950  FORMAT(/23H **BAR  VERIF FAILURE**,17HSOLUTION VALUE
     *    IS,
     *    E12.5,12H NOT SOLBAR=,E12.5)
       WRITE(LUPR,4950)SOLVAL,SOLBAR
C
C         ***SECTION TO VERIFY THE INTEGER (STAR) SOLUTION***
C
C-----CLEAR ROWSUM VECTOR
 5000  MAX=MMM+MAXCUT
       DO 5200 III=1,MAX
       SCRCH(III)=-RHS(III)
 5200  CONTINUE
C
C-----LOOP THROUGH COLUMNS
       SOLVAL=0
       DO 6900 JJJ=1,NNN
       RELXS=XSTAR(JJJ)
       RELXS=RELXS/DENSTR
C-----CLEAR ADJUSTED COST
       SOLVAL=SOLVAL+CCC(JJJ)*RELXS
       ADJCST=CCC(JJJ)
C-----LOOP THROUGH COLUMN ELEMENTS
 6100  MIN=1
C-----CHECK COLUMN OPTIMALITY CONDITIONS
 3630  ADJCST=ADJCST+CCC(JJJ)
       DO 3650 PSN=1,M1M
       JJJ=BBB(PSN)
       IF(JJJ0.EQ.JJJ)GO TO 3700
 3650  CONTINUE
       IF(ADJCST.GE.TOLVER/10.OR.ADJCST.LE.-TOLVER/10)
     *    GO TO 3710
 3670  FORMAT(/23H **BAR  VERIF FAILURE**,8HVARIABLE,
     *    I5/10X,37HIS NOT IN BBB BUT HAS 0 ADJUSTED COST)
       WRITE(LUPR,3670)JJJ
 3710  IF(XBAR(JJJ))3715,3750,3800
C-----NEGATIVE XBAR
 3715  RELXB=XBAR(JJJ)
       RELXB=RELXB/DENBAR
       WRITE(LUPR,3715)JJJ,RELXB,DDD(JJJ)
 3715  FORMAT(/23H **BAR  VERIF FAILURE**,9H VARIABLE,I5,
     *    3H AT,E12.5/10X,21H OUTSIDE BOUNDS= -0 TO,I15)
       GO TO 3900
C-----ZERO XBAR
 3750  IF(ADJCST.GE.-TOLVER)GO TO 3900
 3755  RELXB=XBAR(JJJ)
       RELXB=RELXB/DENBAR
       WRITE(LUPR,3760)ADJCST,JJJ,RELXB,DDD(JJJ)
 3760  FORMAT(/23H **BAR  VERIF FAILURE**,
     *    14H ADJUSTED COST,E12.5,19H WRONG FOR VARIABLE,
     *    I5/1 X,3H AT,E12.5,18H WITH BOUNDS= I TO,I15)
       GO TO 3900
C-----POSITIVE XBAR
 3800  IF(XBAR(JJJ)-DDD(JJJ)*DENBAR)3820,3840,3710
C-----STRICTLY WITHIN BOUNDS
 3820  IF(ADJCST.GT.TOLVER.OR.ADJCST.LT.-TOLVER)
     *    GO TO 3755
       GO TO 3900
C-----UPPER BOUNDED XBAR
 3840  IF(ADJCST.GT.TOLVER)GO TO 3755
C-----END LOOP
 3900  CONTINUE
C
C-----LOOP THROUGH ROWS TO CHECK ROW CONDITIONS
       DO 4900 III=1,MMM
       IF(ROWTYP(III))4100,4300,4500
C-----.GE. ROW
 4100  IF(SCRCH(III).GE.-TOLSUM)GO TO 4200
 4150  WRITE(LUPR,4160)III,SCRCH(III),ROWTYP(III)
 4160  FORMAT(/23H **BAR  VERIF FAILURE**,4H ROW,I5,
     *    11H ROW SUM OF,E12.5/10X,20H WRONG FOR ROW TYPE=,
     *       I3)
       GO TO 4900
C-----CHECK DUAL FEASIBILITY
```

```
      IF(JJJ.GT.1)MIN=ACOL(JJJ-1)+1
      MAX=ACOL(JJJ)
      IF(MAX.GE.MIN)GO TO 6190
      GO TO 6820
 6190 DO 6800 PSN=MIN,MAX
      III=MATRIX(1,PSN)
      COEF=MATRIX(2,PSN)
      SCRCH(III)=SCRCH(III)+COEF*RELXS
      ADJCST=ADJCST-USTAR(III)*COEF
 6800 CONTINUE
C
C-----ADJUST XBAR FOR THE DENOMINATOR
 6820 RELXB=XBAR(JJJ)
      RELXB=RELXB/DENBAR
C-----LOOP THROUGH CUT ELEMENTS
 6830 MIN=ACOL(MMM)+1
      IF(JJJ.GT.1)MIN=ACUT(JJJ-1)+1
      MAX=ACUT(JJJ)
      IF(MIN.GT.MAX)GO TO 6870
      DO 6850 PSN=MIN,MAX
      III=MATRIX(1,PSN)+MMM
      RCOEF=MATRIX(2,PSN)
      SCRCH(III)=SCRCH(III)+RCOEF*(RELXS-RELXB)
      ADJCST=ADJCST-RCOEF*USTAR(III+MMM)
 6850 CONTINUE
C-----CHECK COLUMN OPTIMALITY CONDITIONS
 6870 IF(RELXS)6710,6750,6810
C-----NEGATIVE XSTAR
 6710 WRITE(LUPR,6715)JJJ,RELXS,DOD(JJJ)
 6715 FORMAT(/23H **STAR VERIF FAILURE**,9H VARIABLE,I5,
     *    3H AT,E12.5/10X,21H OUTSIDE BOUNDS= 0 TO,I15)
      GO TO 6900
C-----ZERO XSTAR
 6750 IF(ADJCST.GE.-TOLVER)GO TO 6900
 6755 WRITE(LUPR,6760)ADJCST,JJJ,RELXS,DOD(JJJ)
 6760 FORMAT(/23H **STAR VERIF FAILURE**,
     *    14H ADJUST 0 COST,E12.5,19H WRONG FOR VARIABLE,
     *    I5/1X,3H AT,E12.5,18H WITH BOUNDS= 0 TO,I15)
      GO TO 6900
C-----POSITIVE XSTAR
 6810 IF(RELXS-DOD(JJJ))6820,6840,6710
C-----STRICTLY WITHIN BOUNDS
 6820 IF(ADJCST.GT.TOLVER.OR.ADJCST.LT.-TOLVER)
     *    GO TO 6755
      GO TO 6900
C-----UPPER BOUNDED XSTAR
 6840 IF(ADJCST.GE.TOLVER)GO TO 6755
C-----END LOOP
 6900 CONTINUE
C
C-----LOOP THROUGH ROWS TO CHECK ROW CONDITIONS
      DO 7900 III=1,MMM
      IF(ROWTYP(III))7100,7300,7500
C-----.GE. ROW
 7100 IF(SCRCH(III).GE.-TOLSUM)GO TO 7200
 7150 WRITE(LUPR,7160)III,SCRCH(III),ROWTYP(III)
 7160 FORMAT(/23H **STAR VERIF FAILURE**,4H ROW,I5,
     *    11H ROW SUM OF,E12.5/1X,20H WRONG FOR ROW TYPE=,
     *       I3)
      GO TO 7900
C-----CHECK DUAL FEASIBILITY
 7200 IF(USTAR(III).GE.-TOLVER)GO TO 7250
 7220 WRITE(LUPR,7230)III,USTAR(III),ROWTYP(III)
 7230 FORMAT(/23H **STAR VERIF FAILURE**,4H ROW,I5,
     *    8H DUAL OF,E12.5,21H WRONG FOR ROW TYPE=,I3)
      GO TO 7900
C-----CHECK COMPLEMENTARITY
 7250 IF(SCRCH(III)*USTAR(III).GE.-TOLVER.AND.
     *    SCRCH(III)*USTAR(III).LE.TOLVER)GO TO 7900
      WRITE(LUPR,7260)III,USTAR(III),SCRCH(III)
 7260 FORMAT(/23H **STAR VERIF FAILURE**,4H ROW,I5,
     *    8H DUAL OF,E12.5,12H AND ROW SUM,E12.5
     *    /10X,18H NOT COMPLEMENTARY)
      GO TO 7900
C-----.EQ. ROW
 7300 IF(SCRCH(III).LT.-TOLSUM.OR.SCRCH(III).GT.TOLSUM)GO
     *    TO 7150
      GO TO 7250
C-----.LE. ROW
 7500 IF(SCRCH(III).GT.TOLSUM)GO TO 7150
      IF(USTAR(III).GT.TOLVER)GO TO 7220
      GO TO 7250
C-----END ROW LOOP
 7900 CONTINUE
C
C-----CHECK CUT ROW OPTIMALITY
      DO 7930 CUT=1,MAXCUT
      ROW=CUT+MMM
      IF(USTAR(CUT).LT.-TOLVER)WRITE(LUPR,7230)ROW,
     *    USTAR(CUT),ROWTYP(ROW)
      IF(SCRCH(ROW).GE.-TOLSUM.AND.SCRCH(ROW).LE.TOLSUM)
     *    GO TO 7930
 7920 FORMAT(/23H **STAR VERIF FAILURE**,4H CUT,I5,
     *    11H ROW SUM OF,E12.5/1X,9HNOT TIGHT)
      WRITE(LUPR,7920)CUT,SCRCH(ROW)
 7930 CONTINUE
C
C-----CHECK SOLSTR VALUE
      IF(SOLVAL.LE.SOLSTR+TOLSUM.AND.SOLVAL.GE.
     *    SOLSTR-TOLSUM)GO TO 9000
```

```
C
      SUBROUTINE RIPVPR(ROC,MATRIX,COL1,LENGTH,LIST,MINSUB,
     *   MAXSUB,
     *   OPTN,NAME)
C
      IMPLICIT INTEGER (A-Y)
C
      COMMON /CIPCOM/ SOLBAR,SOLSTR,TOLMOV,TOLOFL,TOLOPT,
     *   TOLSTP,TOLVER,BCONT,BDCONT,BDINT,BDSLK,BFRAC,
     *   BINT,BSLK,CHITST,CONBOT,CONTOP,DATTYP,DELTM,
     *   DENSA,DENSF,DENSP,DENSPQ,DENSQ,DETERM,
     *   DIFDLG,DIMERR,DISRAT,ECONT,EFRAC,EINT,ESLK,
     *   FRAC1,INFIN,LIMA,LIMALP,LIMD,LIMF,LIMR,LIMSLK,
     *   LIMVMN,LIMVMX,LIMWRN,LIMWMX,LUPR,LVLPRT,
     *   MAXCON,MAXCUT,MAXINT,MAX 1,MMM,MU,
     *   NAN,NNNP,NNNQ,NNNS,NNNT,NUMERR,NXTMAT,PRECN,PSLK,
     *   RIPSLK,SEED(2,7),SSLK,TOLMU,VERIFY,VERSN
C
      DOUBLE PRECISION SOLBAR,SOLSTR,TOLMOV,TOLOFL,TOLOPT,
     *   TOLSTP,TOLVER
C
      DIMENSION MATRIX(3,1),COL1(1),LIST(1),EXPAND(50
     *   ,RNOCL(2),ZXPAND(1)
C
      EQUIVALENCE (EXPAND,ZXPAND),(ZRELNO,INTGNO)
C
      DATA RNOCL/4H ROW,4H COL/
C
C-----PRINT HEADER
 1500 FORMAT(///11H ***MATRIX ,A4,4H ***)
      WRITE(LUPR,1500)NAME
C-----LOOP THROUGH THE RANGE OF SUBSCRIPTS
      DO 3000 SUB=MINSUB,MAXSUB
C-----CLEAR OUTPUT AREA
      DO 2200 PSN=1,LENGTH
      EXPAND(PSN)=0
      IF(OPTN.EQ.3.OR.OPTN.EQ.4)ZXPAND(PSN)=0
 2200 CONTINUE
C-----ESTABLISH COL1 SUBSCRIPT
      SUB1=SUB
      IF(OPTN.EQ.2.OR.OPTN.EQ.4)SUB1=LIST(SUB)
C-----SEE IF VECTOR IS VACUOUS
      NOW=COL1(SUB1)
      IF(NOW.GT.0)GO TO 2600
 2300 FORMAT(/A4,1X ,I5, 8H VACUOUS)
      WRITE(LUPR,2300)RNOCL(ROC),SUB1
      GO TO 3000
C-----LOOP THROUGH A NON-VACUOUS VECTOR
 2600 PSN=MATRIX(1,NOW)
      INTGNO=MATRIX(2,NOW)
 7950 FORMAT(/23H **STAT VERIF FAILURE**,17HSOLUTION VALUE
     *   IS,
     *   E12.5,12H NOT SOLSTR=,E12.5)
      WRITE(LUPR,7950)SOLVAL,SOLSTR
C
C        ***EXIT SECTION***
C
 9000 CONTINUE
C-----PRINT TRAILER
 9010 FORMAT(//26H ***END OF VERIFICATION***//)
      WRITE(LUPR,9010)
      RETURN
      END



C
C-------------------------------------------------------------
C
C-----SUBROUTINE RIPVPR(ROC,MATRIX,COL1,LENGTH,LIST,MINSUB,
C   MAXSUB,
C        ,OPTN,NAME)
C-----SUBROUTINE TO PRINT ON UNIT LUPR A SEQUENCE OF VECTORS
C     STORED IN THE GENERAL RIP 3-TUPLE MATRIX
C     AREA.  PARAMETERS ARE AS FOLLOWS--
C        ROC=1 IF THE MATRIX IS STORED BY ROWS
C           =2 IF THE MATRIX IS STORED BY COLUMNS
C        MATRIX=THE GENERAL MATRIX AREA
C        COL1=THE VECTOR OF POINTERS TO VECTOR STARTING 3-
C           TUPLES
C        LENGTH=THE LENGTH OF THE VECTORS BEING PRINTED
C        LIST=A LIST GIVING THE COLUMN POSITIONS IN COL1
C           DESIRED
C           (USED ONLY WHEN OPTN=2)
C        MINSUB=THE SUBSCRIPT OF THE 1ST COLUMN TO PRINT
C        MAXSUB=THE SUBSCRIPT OF THE LAST COLUMN TO PRINT
C        OPTN=1 IF MINSUB AND MAXSUB REFER TO A RANGE OF
C           COL1
C           =2 IF MINSUB AND MAXSUB REFER TO A RANGE OF THE
C           THE INDIRECT REFERENCE LIST.
C           =3 IF MINSUB AND MAXSUB ARE AS IN 1 BUT MATRIX
C              IS
C              FLOATING POINT
C           =4 IF MINSUB AND MAXSUB ARE AS IN 2 BUT MATRIX
C              IS
C              FLOATING POINT
C        NAME=THE (A4) NAME OF THE PRINT OUT
```

```
      IF(OPTN.EQ.1.OR.OPTN.EQ.2)EXPAND(PSN)=INTGNO
      IF(OPTN.EQ.3.OR.OPTN.EQ.4)ZXPAND(PSN)=ZKELNO
      NOW=MAT-IX(3,NOW)
      IF(NOW.GT.0)GO TO 26
C-----PRINT ONE VECTOR
 2900 FORMAT(/A4,1X,I5,1H=,6(I8,1H )/(11X,6(I8,1H )))
 2910 FORMAT(/A4,1X,I5,1H=,6(E12.5,1H )/(11X,4(E12.5,1H )))
      IF(OPTN.EQ.1.OR.OPTN.EQ.2)
     *   WRITE(LUPR,2900)RWORD(ROW),SUBS,(EXPAND(PSN),PSN=1,
     *      LENGTH)
      IF(OPTN.EQ.3.OR.OPTN.EQ.4)
     *   WRITE(LUPR,2910)RWORD(ROW),SUBS,(ZXPAND(PSN),PSN=1,
     *      LENGTH)
 3000 CONTINUE
C
C-----EXIT
 9000 CONTINUE
      RETURN
      END
```