A SIMPLICIAL ALGORITHM

FOR CONCAVE PROGRAMMING

Robert B. Wilson

A thesis submitted in partial fulfillment of
the requirements for the degree of
Doctor of Business Administration

Graduate School of Business Administration
George F. Baker Foundation
Harvard University
June, 1963

Author
g

Abstract

A SIMPLICIAL ALGORITHM FOR CONCAVE PROGRAMMING

Robert B. Wilson

Mathematical programming is a quantitative technique that has been gaining widespread acceptance in business practice as an analytical aid in complicated decision problems. The decision problem, for example, might be to specify a program of production operations and inventory stocking levels for a multi-product, multi-plant firm which must cope with capacity limitations, and which must decide what to produce, and how, before market demands can be known. In another type of problem an allocation of effort must be made among activities using fixed and variable factors of production in different amounts. In substance, the technique of mathematical programming consists of using mathematical methods to find a program which yields the maximum feasible value of some index of performance.

This thesis is a contribution to the body of programming theory and method. The main result is a computational method for solving problems satisfying certain restrictions. Although the Simplex Method for the special case of linear programming has been widely used for a decade, general methods have been developed only recently. The method proposed here extends the rationale of the Simplex Method to more general problems.

The new method has been implemented in computer routines for solving three special types of programming problems: quadratic programming, linear programming under uncertainty, and variable-factor programming. These routines are available for general use, and complete descriptions are included in the thesis along with instructions for using them.

Preface

The research reported in this thesis stems from my investigations in 1961 with Professor John Bishop, into methods of solving a problem of linear programming under uncertainty (cf. Section 5). The problem had been introduced to me by my long time mentor, Professor Howard Raiffa. Although originally proposed as a topic in statistical decision theory (my field of major interest) the problem became one of developing computational methods for mathematical programming: the terminal decision problem, let alone the sampling problem, was far from solved.

Following the lead of Professor Bishop's formulation and structural analysis of the problem, I presented a paper to his seminar on the topic in the fall of 1961. That paper contained in a crude form the computational method which in its matured form is the subject of this thesis. In a revised form the paper was delivered at the Chicago Symposium on Mathematical Programming in June, 1962. Since then, the method has been generalized (and also simplified for exposition), and implemented in computer routines.

Professors Bishop (chairman), Raiffa, and Robert Dorfman have served as the supervising committee for the thesis. I am deeply indebted to each of these men for his efforts in reviewing and helping me with my work. Professor Raiffa deserves special thanks for forcing me to clarify and simplify my ideas in the exposition, and for valiantly trying to teach me how to write. It is a certain fact that the strengths of this thesis are a reflection of these men, and that the weaknesses are my own, since any one of them could, without effort, have written this thesis better than I have.

I would like to add that Drs. Philip Wolfe and Albert Madansky of the RAND Corporation contributed substantially at a crucial point in the development of my ideas by offering a reformulation of the method which bypasses the duality theory of programming.

The Ford Foundation provided financial support for my doctoral studies, for which I am very grateful. It is not, however, responsible

for the views expressed in the thesis. Also, the Division of Research of Harvard Business School made it possible for me to attend the Chicago Symposium.

Table of Contents

1. Introduction

This paper reports my research on a method of solving mathematical programming problems. Although my investigations covered studies in programming theory as well as the development of new computational procedures, my main concern here is the computational work. I will describe a programming method, which I have developed in the course of my research, demonstrate its application on sample problems, and show how it is implemented in computer routines.

The research was motivated by a need to find a general algorithm for solving the constrained extrema problems of mathematical programming. This need arises primarily from industrial problems, but extremal problems are also important in managerial economics. In both the applied and theoretical contexts, a feasible and efficient computational procedure is necessary in order to determine specific, quantified recommendations for maximizing behavior. In addition, an algorithm can be used to explore structural relationships in the decision environment, and to do sensitivity analyses to find the relative magnitude of environmental effects. Even if a decision maker does not use its full potential, a practical computational procedure is an essential companion to normative prescriptions for maximizing behavior.

To fill this need, a programming method must be feasible and efficient on modern computers, and it must also be efficient in terms of human resources. Because man hours are as important as machine time, the method should be widely applicable with little need for modification, easily understood, and easily used.

Experience with the method will finally determine the value of the results reported here. Initial evidence indicates that it is feasible and efficient, and later discussions will show the method to be quite flexible. Although not trivial, it seems to be about as simple conceptually as one can expect. Later discussions provide comparisons with other methods, but technical difficulties prevent a firm conclusion about relative merits.

1.1 Results Obtained

The main result is a computational procedure for solving concave programming problems which satisfy certain differentiability conditions. Developed in a general form in Sections 2 and 4, this procedure is specialized to solve three important types of concave programming problems in Sections 3, 5, and 6. Tangible evidence that the procedure works is a set of three computer routines for solving these types of problems. With little modification in the computer routines one can solve virtually any differentiable concave programming problem.

Two theoretical developments make possible the construction of the computational procedure. The first development is a "Simplicial Algorithm" for quadratic programming, obtained as a special case of a generalization to concave programming of the Simplex Method used in linear programming. The general Simplicial Algorithm is described in Section 2, and then in Section 3 it is specialized to quadratic programming. The quadratic programming algorithm introduces the principle of the floating pivot, which besides making possible a simple tableau representation of the problem, contributes greatly to an efficient algorithm. In more general problems than quadratic programming

the Simplicial Algorithm encounters burdensome computations, but this difficulty is averted by the second development.

The second theoretical development extends the power of the Simplicial Algorithm from quadratic programming into a wider domain of problems. Newton's method of solving simultaneous equations is combined with the simplicial procedures to construct an algorithm which is feasible, and apparently efficient, for general concave programming problems satisfying certain differentiability conditions. No radical departure is involved here; yet, the combination of these two techniques is surprisingly effective. The basic principle at work is that quadratic approximations to the objective function and the constraint functions of the programming problem permit (approximate) determination of the optimal "basis" (defined in Section 2) by the simplicial procedures. Once the basis is known, Newton's method converges to a solution to whatever degree of approximation is desired.

Section 4 develops the programming method in the general case, and then Sections 5 and 6 describe its application in two special types of problems.

1.2 The Computer Routines

Among the concrete accomplishments of the research are computer routines for solving three important types of programming problems. The three problems are the following:

1. Quadratic Programming;
2. Linear Programming under Uncertainty;
3. Variable-Factor (or Blend) Programming.

The formulations of these problems are elaborated in Sections 3, 5, and 6, respectively.

Besides their individual significance, these problems were chosen as applications because they exhibit the programming method from different vantage points. Thus, the Simplicial Algorithm is represented in the quadratic programming routine without any of the complications of simultaneous nonlinear equations. The routine for linear programming under uncertainty exhibits the method for the special case of a nonlinear objective and linear constraints. The complementary situation with a linear objective and nonlinear constraints is demonstrated in the blending routine. (A problem with both a nonlinear objective and nonlinear constraints presents no difficulties beyond those encountered in the three types of problems treated.)

Each of the computer routines has been thoroughly tested. First tested on small problems for which the answers could feasibly be checked in detail by hand, each routine was then used to solve larger problems which were designed to be representative of realistic situations. Finally, each routine was applied to solve a host of problems for which the data was provided by a random number generator. Further details of these tests, examples of problems solved, and data on the computational characteristics of the routines, are included in later discussions.

Anyone with these types of problems to solve can use the computer routines. Complete instructions are included here, and copies of these instructions, plus copies of the program card decks, are deposited with Professor John Bishop. The object-program card

decks are to be used on an IBM 709 or 7090 machine; but recompilations for the FORTRAN source-program card decks suffice to create object-programs for any of a wide variety of machines, ranging from the IBM 1401 to the CDC 1604. In most cases, use of the computer routines requires nothing more than preparing the input data on punch-cards in appropriate fashion.

The practical potential of the research is evidenced in the computer routines. Besides showing that the method works, they are the first members of a programming package. By a programming package we mean a set of computer routines for solving the more important practical programming problems. Each of the three routines that I have developed is built around several core subroutines which accomplish the basic optimizing procedures. With these core subroutines already available, the development of a computer routine for any other differentiable, concave programming problem requires merely the writing of additional subroutines for (1.) data input, (2.) calculations of values of the functions involved, and (3.) output of the solution. Subroutines for these purposes are, of course, unique to each type of problem. Now that the basic theory and the core subroutines have been developed, further development of a rather complete programming package should be relatively easy and economical.

1.3 Relation to Other Work

The ideas in Kuhn and Tucker's original paper lead naturally to the general simplicial procedure developed here. Although my theoretical justification differs somewhat from Beale's justification, the simplicial algorithm in the special case of quadratic programming

with linear constraints is closely akin to his method for that problem. In this context the floating pivot technique can be construed as an extension of Beale's method. A comparison is made in Section 3.

Surveys of the several other general programming methods that have been proposed are given by Wolfe (1962) and by Dorn. The most prominent methods are the gradient projection method of Rosen, the cutting plane method of Kelley, and the method of feasible directions of Zoutendijk. Computer routines employing these methods are not available for direct comparisons, making it difficult to evaluate them in relation to the method proposed here.

No other computer routines are known for solving the particular problems treated in Sections 5 and 6.

2. The Simplicial Algorithm

Dantzig's Simplex Algorithm (1949) for linear programming is generalized in this section to solve any concave programming problem. By a simplicial algorithm, a terminology originating with Wolfe (1962), I mean a programming algorithm which uses the same method of solution as the Simplex Algorithm. The method can be characterized in this way: the search for a solution among all the possible candidates is confined to those candidates which are feasible, and which satisfy the complementary slackness condition (elaborated below). In the nomenclature of linear programming one would say that the search is confined to basic candidates, and this will be true here, after the concept of a basis is extended appropriately. An important feature of simplicial algorithms is that they find a solution after examining only a limited number of candidates. In addition, the Simplex Algorithm, as well as the general Simplicial Algorithm to be described here, is easy to understand, and to use, and possesses a strong rationale in programming theory.

The fundamental theorems of concave programming are presented without proofs in this section. From the theorems I construct a general Simplicial Algorithm. The algorithm uses only the intuitive ideas inherent in these theorems.

In Section 3 I will apply the Simplicial Algorithm to the special case of quadratic programming, for which a substantial refinement of the computational technique is possible. Newton's

method is introduced in Section 4 to construct a variant of the Simplicial Algorithm which is more efficient for nonquadratic programming problems.

2.1 Formulation of the Concave Programming Problem, and Notational Conventions

Our subject matter, as formulated originally by H. W. Kuhn and A. W. Tucker, is the

> Concave Programming Problem: Find a value of the program vector $x = (x_j)$, in the J-dimensional real domain, yielding the maximal value of the objective function $g(x)$ subject to the nonnegativity constraints $x_j \geq 0$ for $j = 1, \ldots, J$, on the program vector and the N additional constraints $f_n(x) \geq 0$, for $n = 1, \ldots, N$, where g and each f_n is real-valued and concave.

For our purposes it suffices that g and f_n be defined only over the feasible subspace:

$$F = (x | x \geq 0, \quad f_n(x) \geq 0, \quad n = 1, \ldots, N) \quad . \tag{2-1}$$

A function such as g, or any f_n, is concave if linear interpolation between the values at any two points of its range yields a value not greater than its actual value at the point of interpolation; viz.,

$$a g(x^1) + [1-a]\, g(x^2) \leq g(a x^1 + [1-a]\, x^2) \quad , \quad 0 \leq a \leq 1 \quad , \tag{2-2}$$

for every x^1 and x^2 in the domain. Among the many references on

concavity, and the related concept of convexity, one can consult Fenchel, Eggleston, or Appendix B of Karlin, where the properties of concave functions can be found.

A concave function possesses partial derivatives almost everywhere, so we can use the gradient operator ∇ defined by

$$\nabla g(x) \equiv (\partial g(x)/\partial x_1, \ \ldots, \ \partial g(x)/\partial x_J)^t \tag{2-3}$$

to transform the scalar valued function g into the vector valued function ∇_g of the same argument. At a point where a concave function does not have some partial derivative the left and right partial derivatives will be defined, and we will assume that at the time required some proportional combination of left and right partial derivatives will be specified appropriate to the circumstances. It should be noted that gradient vectors are column vectors, as are all the other vectors used in this paper.

Extending the notation further, we define the gradient matrix of a vector valued function, say $f(x) \equiv (f_n(x))$, as the J by N matrix

$$\nabla f(x) \equiv (\nabla f_n(x)) \equiv (\partial f_n(x)/\partial x_j) \ , \tag{2-4}$$

where $n = 1, \ \ldots, \ N$ indexes the component gradient vectors across the columns.

Using the gradient notation, a scalar or vector valued function f is concave if and only if

$$f(x^2) \leq f(x^1) + \nabla f(x^1)^t [x^2 - x^1] \tag{2-5}$$

for all x^1 and x^2. If f in (2-5) is interpreted as a production function, then this relation expresses the proposition that, in economic terms, a concave function exhibits nonincreasing marginal productivity.

2.2 Conditions for a Solution

H. W. Kuhn and A. W. Tucker proved the fundamental theorems of concave programming 1950. Although much work has been done since then, it is not necessary here to recount the later developments. A compendium of work in the area is given by Vajda, while Karlin displays the elegance and rigor of the present theory. The relevant theorems, all due to Kuhn and Tucker, are stated here without proofs. To these theorems I will add an important lemma in a later discussion.

The substantive proposition of the Kuhn-Tucker theory demonstrates an equivalence between the concave programming problem and a related saddle problem.

Definition: The saddle problem with objective function $G(x; u)$ is to find vectors $x^\circ \geq 0$ and $u^\circ \geq 0$ for which

$$G(x; u^\circ) \leq G(x^\circ; u^\circ) \leq G(x^\circ; u) \quad \text{for all } x \geq 0 \text{ and } u \geq 0 .$$

For the economic interpretations of the saddle problem see, for example, the comments of Kuhn and Tucker, or in limited contexts Samuelson, Dorfman, and Dorfman, Samuelson, and Solow.

The main accomplishment of the theory is to identify a form of the objective function in the saddle problem such that the programming problem and the saddle problem are equivalent. From this equivalence one can then deduce conditions for a solution. That any solution $(x^\circ; u^\circ)$ to the saddle problem yields x° as a solution to the programming problem

is immediate, even without the concavity assumptions. The more difficult task is an existence theorem showing that under certain conditions there will exist a solution $(x^o; u^o)$ to the saddle problem for every solution x^o to the programming problem. The concavity assumption is needed for this theorem; in addition, it is necessary to impose a regularity condition on the constraints in the form of a

Constraint Qualification: Let $H(x)$ be the matrix formed by appending the J-dimensional identity matrix to $-\nabla f(x)$, as

$$H(x) \equiv [-\nabla f(x), I] ,$$

and define $\hat{H}(x)$ to be any J by J matrix obtained from $H(x)$ by deleting any N columns; then it is assumed that every $\hat{H}(x)$ is non-singular for all x in the feasible subspace F.

The role of the constraint qualification is demonstrated by Kuhn and Tucker who formulate it in different terms, and by Wilde. Always satisfied by independent linear constraints, the constraint qualification will also be satisfied whenever there exists a program vector for which no constraints are binding; viz., there exists $\bar{x} \geq 0$ for which $f(\bar{x}) > 0$. The constraint qualification is only mildly restrictive, and is assumed to be satisfied in all the work in this paper.

The following are the fundamental theorems of concave programming proved by Kuhn and Tucker.

Equivalence Theorem: x^o is a solution to the concave programming problem if and only if there exists a solution $(x^o; u^o)$ to the saddle problem with objective function

$$G(x; u) \equiv g(x) + u^t f(x) .$$

This theorem establishes the desired equivalence relation, so that an investigation of the programming problem can be conducted via the saddle problem. Due to the concavity assumptions, the differential conditions for a solution to the saddle problem are sufficient as well as necessary, as stated in the next

Theorem: $(x^o; u^o)$ is a solution to the saddle problem of concave programming if and only if

a) $x^o \geq 0$, $\nabla g(x^o) + \nabla f(x^o)u^o \leq 0$, $[\nabla g(x^o) + \nabla f(x^o)u^o]^t x^o = 0$;

b) $u^o \geq 0$, $f(x^o) \geq 0$, $u^{ot} f(x^o) = 0$.

This theorem states a mutual exclusiveness between a variable and the corresponding slack in the inequalities of the differential conditions for a solution. Such a relation between variables is formalized in the

Complementary Slackness Condition: Two variables x_j and v_j satisfy the complementary slackness condition when one of them is constrained identically to zero.

Two vectors x and v satisfy the complementary slackness condition when each component pair x_j and v_j satisfies it. Introducing slack vectors v and s into the right inequalities of (a) and (b) above, respectively, the previous results combine to yield the

Major Theorem: x^o is a solution to the concave programming problem if and only if there exist N-dimensional vectors s^o and u^o, and a J-dimensional vector v^o, such that

(i) x^o, s^o, v^o, and u^o are all nonnegative;

(ii) $\begin{bmatrix} x^o \\ s^o \end{bmatrix}$ and $\begin{bmatrix} v^o \\ u^o \end{bmatrix}$ satisfy the complementary slackness condition;

(iii) x^o, s^o, v^o, u^o provide a solution to the following set of J + N equations in 2(J + N) variables,

$$\nabla g(x) + v + \nabla f(x)u = 0$$
$$-f(x) + s = 0 \qquad (2\text{-}6)$$

The operational aspects of the Kuhn-Tucker results are summarized in this Major Theorem. To put it briefly, it states that the computational task is to find a solution to the simultaneous equations (2-6), which satisfies as well (i) the nonnegativity condition and (ii) the complementary slackness condition.

Were it not for the nonnegativity condition (i) and its derivative, the complementary slackness condition (ii), the Kuhn-Tucker results would be the means as well as the end of concave programming. As it is, however, the conditions for a solution cannot be resolved directly, due to combinatorial complexities. Under the circumstances, a systematic search procedure is needed.

2.3. Nature of the Computational Task

Two techniques are needed for solving a concave programming problem: one is a way of solving the simultaneous equations (2-6), and the other is a procedure for searching for a solution to the programming problem among the solutions to (2-6). My immediate concern in this section is the search procedure. In order to concentrate on the subject at hand, assume that the algebraic techniques required are all accomplished by a "black box". I will specify what goes on inside this black box in Sections 3 and 4.

The nature of the computational task is implicit in various ways in the Major Theorem, at least three approaches being possible. One way is to ignore, for the most part, the conditions for a solution except as a check for optimality at the end.. Most of the algorithms

(cf. Wolfe (1962)) based on gradient procedures do this in effect, relying on feasible directions of steepest ascent as their path of search, and as the means of solution.

A second way is to identify all solutions to (2-6) and then to search among them for an optimal candidate; unfortunately, very little work has been done in this direction.

The third way, the approach of the Simplicial Algorithm, is to satisfy part of the conditions for a solution, and then systematically to search for an optimal candidate by making adjustments in the remaining conditions until they are all satisfied. In practice, the conditions (ii) and (iii) of the Major Theorem are easy to satisfy initially, and it is usually relatively easy to satisfy part of condition (i) also. I shall discuss the Simplicial Algorithm for only one way of satisfying part of the nonnegativity condition (i); viz., when an initial candidate is available for which $x \geq 0$, $s \geq 0$.

2.4. The Notion of a Basis, and Some Terminology

In order to obtain a solution to (2-6) which satisfies the complementary slackness condition (ii) of the Major Theorem, one needs merely to specify for each pair of complementary variables x_j and v_j, and s_n and u_n, which member of the pair is to be constrained to zero. There must then be $J + N$ unconstrained variables in terms of which the $J + N$ simultaneous equation (2-6) can be solved.

The unconstrained variables will be called basic variables, and the others, nonbasic variables. A basis is then defined as a set of indices of the vector $(x^t, s^t, v^t, u^t)^t$. When it is unqualified, the

term basis presumes that the complementary slackness condition is satisfied, so that if the index $\ell \leq J + N$ is in the basis then $\ell + J + N$ is not in the basis, and vice versa. For a semibasis the complementary slackness condition need not be satisfied, but otherwise the definition is the same.

Given some semibasis, the values of the basic variables will be denoted by a superimposed carat. Then a basic candidate is a quadruplet $(\hat{x}; \hat{s}; \hat{v}; \hat{u})$ of vectors of values of basic variables. Being constrained to zero, the nonbasic variables are omitted from the formulation.

The values of the basic variables are determined by solution of (2-6). The Major Theorem assures that a solution to these equations will exist for the optimal choice of a basis, provided a finite solution exists to the programming problem. No such assurance is given for a nonoptimal basis, but this presents no difficulties in practice since the search procedure moves from any one basis consistent with (2-6) to some other which, by construction, is also consistent.

We shall call the x and s variables primal variables, and the u and v variables dual variables. The technical terminology calls x and u the primal and dual program variables, respectively; and s and v, the primal and dual slack variables, respectively. Because the differentiation between program and slack variables is not always necessary, we shall often use $y \equiv (x^t, s^t)^t$ and $w \equiv (v^t, u^t)^t$ to denote the vectors of primal and dual variables, respectively.

A feasible candidate will be taken to be a primal-feasible candidate; i.e., a candidate $(y; w)$ for which $y \geq 0$.

2.5. The Significance of Dual Variables; A Lemma

More information is needed in order to make the Kuhn-Tucker results operational. The Major Theorem provides a test for accepting or rejecting candidates for a solution, but it does not indicate a criterion for improving one's choice. In particular, one needs to know which nonbasic variables can beneficially be made basic and/or which basic variables might be made nonbasic.

Partially formulated by Dreyfus and Freimer and proved by Wilde, a simple lemma yields the required information.

Lemma: Given a basis for which y_ℓ is a nonbasic primal variable,

$$\partial g(\hat{y})/\partial y_\ell = -\hat{w}_\ell .$$

The hypothesis requires that the complementary slackness condition be satisfied by $(\hat{y}; \hat{w})$ in the sense that (1) the nonbasic variables are all specified as constants, or parameters, in (2-6), and (2) of two complementary variables, y_ℓ and w_ℓ, one must be basic and the other nonbasic. The values of the basic variables are assumed to be determined by (2-6). The result says that as the nonbasic primal variable y_ℓ is increased, all the while keeping the other nonbasic variables constrained to zero, and keeping the basic variables so as to satisfy (2-6) given y_ℓ as a parameter of the system, that the rate of change in the objective function is the negative of the complementary basic dual variable. Specifically, if the constraint ℓ were to be tightened by a small increment Δy_ℓ, then to a first approximation the change in the value of the objective function would be $-\hat{w}_\ell \, \Delta y_\ell$. Hence, it pays to increase a nonbasic primal variable for which the corresponding basic dual variable is negative.

2.6. The General Simplicial Procedure

Let me outline the algorithm in terms of the ideas that have already been discussed. We will start from an initial basis for which solution of (2-6) yields a feasible candidate, $\hat{y} \geq 0$. (Methods of obtaining an initial feasible basis are described by Wolfe (1962).) Ifthe candidate is also dual feasible, $\hat{w} \geq 0$, then the basis is optimal, according to the Major Theorem. Otherwise, we will make basis changes one at a time to eliminate dual infeasibility, while retaining primal feasibility, until an optimal basis is obtained. The key to the algorithm is the lemma, since it indicates how to change the basis so as to improve the attained value of the objective function.

The procedure can be outlined roughly as follows: Suppose $(\hat{y}, \hat{w})$ is a basic feasible candidate, $\hat{y} \geq 0$ but $\hat{w}$ is not nonnegative. In this case there is no loss of generality if we choose the labeling so that $\hat{w}_1 < 0$. According to the lemma, increasing the nonbasic primal variable y_1 will increase the objective function. Hence we proceed by releasing the constraint $y_1 = 0$ and drive, $\hat{w}_1$, so to speak, towards zero assuming that $(\hat{y}, \hat{w})$ is kept in balance to satisfy (2-6), and that all nonbasic variables other than y_1 remain at zero. As we push $\hat{w}_1$ towards zero $\hat{y}$ changes continuously. If we can push $\hat{w}_1$ all the way to zero keeping $\hat{y} \geq 0$ then we arrive at a new basis where $\hat{y}_1 > 0$, and $\hat{w}_1 = 0$. If, however, we cannot push $\hat{w}_1$ all the way towards zero maintaining feasibility of the $\hat{y}$ then we can assume that the labeling is chosen so that $\hat{y}_2$ is the first basic primal variable to hit zero. We can now continue driving $\hat{w}_1$ towards zero by holding fast on $y_2 = 0$ but now releasing w_2. In

this way we can continue until all the basic dual variables, as well as the basic primal variables, are nonnegative and a solution has been found. Now I shall elaborate the details.

Select a basic dual variable which is negative, say $\hat{w}_1 < 0$. According to the lemma, increasing the complementary nonbasic primal variable, y_1, will increase the objective function, so we want to increase y_1 as long as it is beneficial and feasible. Our task, then, is to ascertain two critical values of y_1; viz.,

1. $\bar{\bar{y}}_1$, the critical value at which it is no longer beneficial to increase y_1; and

2. $\bar{\bar{y}}_1$, the critical value at which it is no longer feasible to increase y_1.

The smaller of these two critical values will dictate the change of basis to be made. The first critical value is easily determined from the lemma, but the second critical value introduces some complications.

The first critical value will be that value of y_1 for which a further increase would not increase the objective function. Consequently, applying the lemma again it is that value for which $\hat{w}_1$ becomes zero. If no positive value of y_1 will drive $\hat{w}_1$ to zero, then $\bar{\bar{y}}_1$ is unbounded. When $\bar{\bar{y}}_1$ is less than $\bar{\bar{y}}_1$ it is clear that we will make w_1 nonbasic and replace it by making y_1 basic: unless this yields an optimal basis, the entire procedure will then be repeated.

To find $\bar{\bar{y}}_1$, we increase y_1 in the same manner as above until some basic primal variable, say $\hat{y}_2$, passes zero to become negative. That is, we want to increase y_1 as far as it is feasible to do so. It will no longer be feasible to increase y_1 further when some basic primal variable

becomes negative. By identifying $\bar{\bar{y}}_1$ as the smallest positive value of y_1 for which some basic primal variable, $\hat{y}_2$, becomes zero, we are specifying the maximum amount that y_1 can be increased. This much is clear; what is not so clear is how the basis is to be reconstituted. The situation is such that we will find it desirable to proceed, not directly from basis to basis, but from basis to basis with intermediate stops at semibases.

The reasoning is as follows. If $\bar{\bar{y}}_1$ is less than $\bar{y}_1$, then we know that $\hat{w}_1$ is still negative at $y_1 = \bar{\bar{y}}_1$, and the attained value of the objective function can still be improved by increasing y_1 beyond $\bar{\bar{y}}_1$. To increase y_1 further, however, would necessitate appending the constraint $\hat{y}_2 = 0$ to (2-6) in order to avoid becoming infeasible, complicating the technique drastically. A simple trick, however, alleviates the difficulty. Specify the constraint on y_2 by making it a nonbasic variable and replace it with y_1 as a basic variable. Although this yields a semibasis, it is not a proper basis because y_1 and w_1 are both basic variables, and y_2 and w_2 are both nonbasic variables. But since y_2 is nonbasic we are now free to increase w_2. This allows us to increase y_1 and w_1 further, not by increasing y_1 directly since it is now basic and is a dependent variable determined by (2-6), but by increasing w_2 as the controllable parameter in the system. (The fact that we want to increase w_2, rather than to decrease it, in order to drive $\hat{w}_1$ to zero is seen by the fact that if $\hat{w}_1$ were forced to zero by making w_1 nonbasic and w_2 basic, then the attained value of w_2 would be $\hat{w}_2 = -\partial g/\partial y_2$, which must be positive since we have just previously found that a beneficial increase in y_1 was prevented by $\hat{y}_2$

becoming negative.) Hence, we must do for w_2 what we did for y_1; viz., calculate the two critical values $\bar{w}_2$ (at which $\hat{w}_1$ is driven to zero) and $\bar{\bar{w}}_2$ (at which some basic primal variable, say $\hat{y}_3$, goes to zero) and repeat the analysis. In this way we can go from a basis through a sequence of semibases until finally $\hat{w}_1$ can feasibly be driven to zero. At that stage there is some nonbasic dual variable, say w_4, which is the paramster in (2-6), and we can reconstitute the basis by making w_4 basic and w_1 nonbasic. From there the entire procedure can be repeated until an optimal basis is found.

If at any stage either critical value is negative or is inconsistent with (2-6), then it is taken to be infinite. If a critical value is zero then the problem is possibly degenerate (in the same sense as in linear programming (Gass)) and it is well to perturbate the data of the problem slightly to remove the indeterminacy.

This is the substance of the simplicial search procedure, although a good deal remains to be said about the algebraic technique required to implement it. Sections 3-6 are concerned mainly with the algebraic technique. Omitting the details, we can formalize the search procedure in the flow chart shown in Figure 2-1.

Figure 2-1

1. Start by choosing a feasible initial basis.
2. Calculate values of basic variables from (2-6).
3. Are basic variables all nonnegative? —Yes→ Stop
   |
   No
   ↓
4. Choose a negative basic dual variable, and identify its co entary (nonbasic primal) variable as the parametric variable.
5. Increase the parametric variable in (2-6) to find its two critical values (Stop if both are unbounded).
6. Is the first critical value the smaller? —Yes→ Step 8
   |
   No
   ↓
7. Make the parametric variable a basic variable to replace the basic primal variable which became zero, identify the latter's complementary (nonbasic dual) variable as the new parametric variable and return to 5.
8. Make the parametric variable a basic variable to replace the negative basic dual variable, and return to 2.

2.7. A Geometrical Description

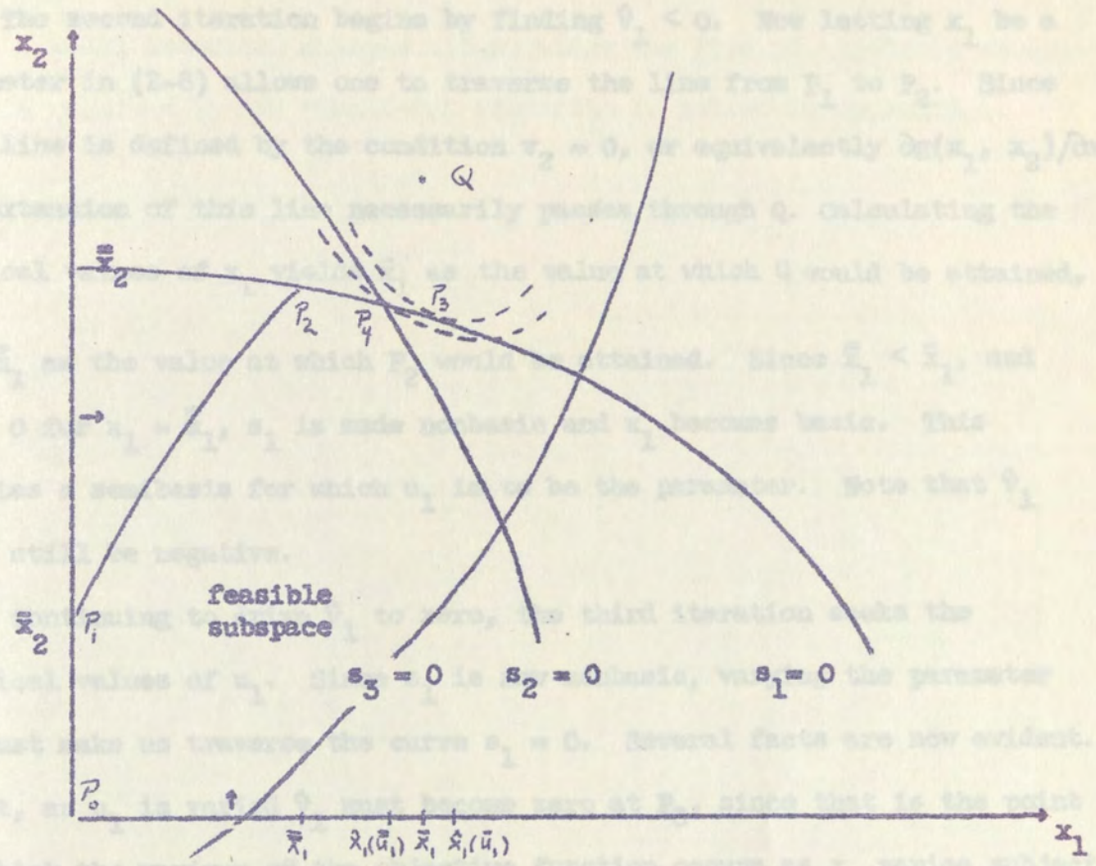
A geometrical description of the search procedure highlights some of its important features.

Figure 2-2 depicts a representative feasible subspace for $J = 2$, $N = 3$. The boundaries of the various constraints are represented by the curves for which $x_1 = 0$, $x_2 = 0$, $s_1 = 0$, $s_2 = 0$, and $s_3 = 0$, respectively, given that the primal variables are related by the system,

$$-f_1(x) + s_1 = 0 ,$$
$$-f_2(x) + s_2 = 0 ,$$
$$-f_3(x) + s_3 = 0 .$$

The feasible subspace is the set of primal program vectors (points in Figure 2-2) for which all the primal variables are nonnegative. The objective function attains an absolute maximum at P_4, and at P_3 when constrained only by $f_1(x) = 0$. The dotted curves in Figure 2-2 are isoquants of the objective function. The segmented curve $P_0P_1P_2P_4$ will be the path to the solution from the initial candidate P_0.

Figure 2-2



The Simplicial Algorithm operates as follows. For the initial basis,

$$B_0 = (3, 4, 5, 6, 7) ,$$

the slack variables are all basic, and the initial candidate is the origin, P_0, in the space of primal program variables. This basis is feasible because $\hat{s}_1 > 0$, $\hat{s}_2 > 0$, and $\hat{s}_3 > 0$. It is not optimal, however, since $\hat{v}_2 < 0$. Letting x_2 be a parameter in the system (2-6), which here takes the form

$$\nabla g(x_1 = 0 , x_2) + \hat{v} = 0 ,$$

$$-f(x_1 = 0 , x_2) + \hat{s} = 0 ,$$

yields $\bar{x}_2 < \bar{\bar{x}}_2$. Hence, v_2 is made nonbasic and x_2 becomes basic to attain the point P_1 at the end of the first iteration.

The second iteration begins by finding $\hat{v}_1 < 0$. Now letting x_1 be a parameter in (2-6) allows one to traverse the line from P_1 to P_2. Since this line is defined by the condition $v_2 = 0$, or equivalently $\partial g(x_1, x_2)/\partial x_2 = 0$, the extension of this line necessarily passes through Q. Calculating the critical values of x_1 yields $\bar{\bar{x}}_1$ as the value at which Q would be attained, and $\bar{\bar{x}}_1$ as the value at which P_2 would be attained. Since $\bar{\bar{x}}_1 < \bar{\bar{x}}_1$, and $\hat{s}_1 = 0$ for $x_1 = \bar{\bar{x}}_1$, s_1 is made nonbasic and x_1 becomes basic. This creates a semibasis for which u_1 is to be the parameter. Note that $\hat{v}_1$ must still be negative.

Continuing to drive $\hat{v}_1$ to zero, the third iteration seeks the critical values of u_1. Since s_1 is now nonbasic, varying the parameter u_1 must make us traverse the curve $s_1 = 0$. Several facts are now evident. First, as u_1 is varied $\hat{v}_1$ must become zero at P_3, since that is the point at which the maximum of the objective function occurs as x_1 varies subject to $s_1 = 0$. This identifies the value of $\bar{\bar{u}}_1$. Second, it must always be that $\bar{\bar{u}}_1 > 0$, since if the point P_3 were attained by making v_1 nonbasic and u_1 basic, then one would have $\partial g/\partial s_1 = -\hat{u}_1$ which would have to be negative. Increasing u_1 as a parameter in (2-6), then, we obtain $\bar{\bar{u}}_1$ yielding P_4 for which $\hat{s}_2 = 0$, and $\bar{\bar{u}}_1$ yielding P_3. Because $\bar{\bar{u}}_1 < \bar{\bar{u}}_1$ we make s_2 nonbasic and u_1 basic.

In this example we know that the final iteration will consist of making u_2 basic and v_1 nonbasic. This is because P_4 is a vertex of the feasible subspace, so that there are no degrees of freedom allowing the

primal variables to vary as u_2 is varied; hence $\bar{\bar{u}}_2$ is unbounded and it must be that $\tilde{u}_2 < \bar{\bar{u}}_2$.

A small numerical example illustrating the type of algebraic manipulations required by the Simplicial Algorithm is solved in Appendix A.

3. Quadratic Programming

The Simplicial Algorithm for quadratic programming is described in this section. My main concern here are the details of the algebraic technique, since the general algorithm was described in Section 2. In this special case linear algebra suffices, so that a simple and efficient computational scheme is possible.

Among the reported problems to which quadratic programming has been applied are portfolio analysis (Markowitz), pricing and activity analysis when demand curves are linear (Dorfman), and allocation of a strategic material (Karreman). Quadratic programming has a special role in this paper: in Section 4 I will present a method of solving more general concave programming problems, in effect, by solving a succession of approximating quadratic problems.

3.1. Introduction

The quadratic programming problem is a special case of the concave programming problem defined in Section 2, in which the objective is a quadratic function, and the constraints are linear functions. The notation we shall use is specified in the following statement of the

Quadratic Programming Problem: solve the concave programming problem when the objective function is

$$g(x) \equiv p^t x + x^t A x ,$$

and the constraint function is

$$f(x) \equiv c - Dx .$$

The quadratic form must be negative semidefinite over the feasible subspace in order to satisfy the concavity assumptions.

Recall from the formulation of the Simplicial Algorithm in Section 2 (summarized in the flow chart in Figure 2-1), that there are two algebraic operations required by the algorithm. In each iteration one must do the following:

1. Given a nonbasic parametric variable which is to be increased, determine the two critical values at which increasing the parametric variable further is no longer beneficial, and feasible, respectively;

2. Given the basic variable which is to be made nonbasic as determined by the smaller of the two critical values, make the parametric variable basic and determine the new values of all the basic variables.

The first operation is called the search procedure, and the second is called the pivoting operation. In the search procedure we are searching for the basic variable to be made nonbasic; the name of the pivoting operation comes to us from linear algebra. Both operations are familiar from the Simplex Algorithm for linear programming. They are described below in sequence after a preliminary discussion in which the computational task is cast in terms of linear algebra, and some appropriate notation is established.

The discussions below will, in effect, be specializing the flow chart of the general algorithm in Figure 2-1 to the complete flow chart for quadratic programming in Figure 3-1. The reader might find an occasional look at these flow charts helpful in recalling how the pieces discussed below fit together.

3.2. The Basic Representation

The algebraic operations of the Simplicial Algorithm are mostly concerned with finding and analyzing basic candidates determined as solutions

to the simultaneous equations (2-6) in the Major Theorem. For the special case of quadratic programming these equations are linear, taking the following form:

$$\begin{bmatrix} A + A^t & 0 & I_J & -D^t \\ D & I_N & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ s \\ v \\ u \end{bmatrix} = \begin{bmatrix} -p \\ c \end{bmatrix}. \tag{3-1}$$

If we consider the matrix on the left as a collection of column vectors, then we can regard (3-1) as requiring us to find a representation of the vector on the right as a linear combination of the vectors in the matrix, or just of the basic vectors when a semibasis is given for the programming problem. Thus, the basic vectors form a basis for a vector space over which the representation is to be obtained, where the term basis is understood to indicate that it is interpreted in the sense of linear algebra (cf. Orden).

To see where these ideas lead us, let us adjoin the vector on the right to the matrix in (3-1) and call the resulting $(J + N)$ by $2(J + N) + 1$ matrix $R \equiv (r_{ij})$; viz.,

$$R \equiv \begin{bmatrix} A + A^t & 0 & I_J & -D^t & -p \\ D & I_N & 0 & 0 & c \end{bmatrix}. \tag{3-2}$$

For notational convenience we will reorder the columns of R so that the basic vectors occur on the left, as

$$R \equiv \begin{bmatrix} R_B , & R_N , & r \end{bmatrix}, \tag{3-3}$$

where R_B is the matrix of basic vectors in some order, R_N is the matrix of nonbasic vectors in the same order, and r is the last column of (3-2). Now R can be represented formally as a collection of column vectors represented over the unit vectors,

$$R = IR \,, \tag{3-4}$$

where I is the (J + N)-dimensional identity matrix. We can see, however, that what we want to work with is the representation, say $\hat{R}$, of R over the basic vectors R_B,

$$R = R_B \hat{R} \,; \tag{3-5}$$

for,

$$\hat{R} = \left[I \,, \quad R_B^{-1} R_N \,, \quad R_B^{-1} r \right] , \tag{3-6}$$

which yields us the values of the basic variables in the last column.

Given any semibasis for the programming problem, we shall call the representation of R over the basic vectors the basic representation, and it will be denoted by $\hat{R} \equiv (\hat{r}_{ij})$. Note that $\hat{R}$ is the basic representation relative to whatever semibasis is currently at hand in the course of the algorithm.

3.3. The Search Procedure

Let us first recall the role of the critical values. As we begin the search for the basic variable to be made nonbasic, we will have on hand a negative basic dual variable which we are driving to zero. This basic dual variable we will call the floating pivot variable, the reason being that possibly this variable will be with us for several iterations as we try repeatedly to drive it to zero, and then when finally it can feasibly be driven to zero it will be the variable made nonbasic when we execute the pivoting operation. We will also have found a nonbasic variable, called the parametric variable, which is to be increased as long as it is beneficial and feasible. The parametric variable will be a primal variable when a proper basis obtains, and otherwise it will be

a dual variable; hence, rather than use the (y, w) notation of Section 2, we will here represent the parametric variable as z_p, where p is the index of its column in $\hat{R}$. The two critical values of the parametric variable are $\bar{z}_p$, at which no further increase is beneficial; and $\bar{\bar{z}}_p$, at which no further increase is feasible, as determined by the fact that some basic primal variable becomes negative.

In order to determine the critical values, let $\hat{z}$ and $\check{z}$ be the vectors of values of basic and nonbasic variables, respectively, and observe from (3-1) that

$$\hat{z} = R_B^{-1} r - R_B^{-1} R_N \check{z} . \tag{3-7}$$

Now as $\check{z}_p$ is increased,

$$\hat{z} = \hat{R}_L - \hat{R}_p \check{z}_p , \tag{3-8}$$

where from (3-6), $\hat{R}_L$ and $\hat{R}_p$ are the appropriate columns of $\hat{R}$, and we let the index $L = 2(J + N) + 1$ signify the last column. The formulas for the critical values now follow immediately from (3-8).

1. Let f be the index of the row in $\hat{R}$ of the floating pivot variable. Then $\bar{z}_p$, the value of z_p at which the floating pivot variable becomes zero, is

$$\bar{z}_p = \hat{r}_{fL} / \hat{r}_{fp} \tag{3-9}$$

provided $\hat{r}_{fp} < 0$, and it is unbounded otherwise ($\hat{r}_{fL}$, the current value of the floating pivot variable is negative by construction).

2. The first value of z_p at which a further increase is infeasible is $\bar{\bar{z}}_p$, determined as the minimum nonnegative ratio $\hat{r}_{iL} / \hat{r}_{ip}$, where i ranges over the rows of those basic primal variables for which $\hat{r}_{ip} > 0$.

These easy formulas for the critical values direct one to go down column p of $\hat{R}$, and for each row i for which either 1) $i = f$ and $\hat{r}_{ip} < 0$, or 2) i indexes the row of a basic primal variable and $\hat{r}_{ip} > 0$, calculate the ratio $\hat{r}_{iL}/\hat{r}_{ip}$. The minimum nonnegative ratio then identifies the basic variable to be made nonbasic. If no such ratio exists, then the solution is unbounded with the parametric variable increased to infinity. If the minimum is zero, then the problem is degenerate and it is well to perturbate that $\hat{r}_{iL} = 0$ to a small positive value $\hat{r}_{iL} = \epsilon$ before continuing.

3.4. The Pivoting Operation

What we need for the pivoting operation is a formula that will take the row index, say it is b, of the basic variable to be made nonbasic, and the column index p of the parametric variable to be made basic, as input data and then transform the current basic representation into the new basic representation. This operation might be depicted schematically as follows:

$$\hat{R} \leftarrow \underline{P(b, p)}\, \hat{R}\,, \tag{3-10}$$

where P denotes the pivoting operation. In theoretical terms the job of the pivoting operation is to replace a basic vector by a nonbasic vector in the basis over which $\hat{R}$ is represented. The arithmetic details of such a formula are available to us from linear algebra (cf. Orden), and they have been applied previously to linear programming by Dantzig, and by Dantzig, Orden and Wolfe.

Pivoting Operation: For a pivot column p defined by a nonbasic vector, and a pivot row b defined by a basic vector, transform the matrix $\hat{R} \equiv (\hat{r}_{ij})$ as follows, in the indicated order:

$$
\begin{aligned}
&\text{(i)} && \hat{r}_{ip} \longleftarrow -\hat{r}_{ip}/\hat{r}_{bp}, && i \neq b; \\
&\text{(ii)} && \hat{r}_{ij} \longleftarrow \hat{r}_{ij} + \hat{r}_{ip}\,\hat{r}_{bj}, && i \neq b,\ j \neq p; \\
&\text{(iii)} && \hat{r}_{bj} \longleftarrow \hat{r}_{bj}/\hat{r}_{bp}, && j \neq p; \\
&\text{(iv)} && \hat{r}_{bp} \longleftarrow 1/\hat{r}_{bp}. &&
\end{aligned}
$$

The operation is, of course, not defined when the pivot element r_{bp} is zero.

It might be noted here that the formulas for the critical values determined earlier are a reflection of Step (iii) in the pivoting operation, since the critical values are actually just the values the parametric variable would assume for various possible basis changes.

3.5. The Initial Basic Representation

The reader is now in a position to understand all of the flow chart in Figure 3-1 except Step 1, which will be explained below.

The first step of the algorithm is to establish the basic representation for the initial basis. This can be done using the pivoting operation by obtaining a sequence of relations equivalent to the transition from (3-4) to (3-5). Using the pivoting operation, one replaces the unit vectors in the basis of R one after another by the basic vectors in R_B to obtain the sequence.

$$
\begin{aligned}
R &= IR \equiv B^{0} R^{0} \\
R &= B^{1} R^{1} \\
&\vdots \\
R &= B^{J+N} R^{J+N} \equiv R_B \hat{R},
\end{aligned}
\qquad (3\text{-}11)
$$

where

(a) B^{p} is the matrix of vectors in the basis after p unit vectors have been replaced, with $B^{0} \equiv I$ and $B^{J+N} \equiv R_B$;

and

(b) R^p is the representation of R over the basis defined by B^p, with $R^0 \equiv R$ and $R^{J+N} \equiv \hat{R}$.

The computational scheme yielding this sequence is defined by the

Initialization Procedure: Perform the pivoting operation on R^{p-1} to obtain R^p for $p = 1, 2, \ldots, (J + N)$, where for each p the pivot row b can be any row not previously chosen for which the pivot element is nonzero.

A sequence of nonzero pivot elements must exist if R_B is nonsingular (Orden), which we impose as a condition on the choice of the initial basis. If $b \neq p$ then, as shown by Orden, the rows are permuted so that row b is relabeled as row p (this is done often in practice, since choosing the pivot element which is largest in absolute value minimizes round-off errors).

Figure 3-1

Flow Chart of the Simplicial Algorithm for Quadratic Programming

0. Obtain an initial feasible basis.

1. To obtain $\hat{R}$, establish the initial basis via the sequence (3-7) by executing the pivoting operation* on R^{p-1} for $p = 1,\ldots,(J + N)$, for each p letting b be that row among those not previously used for which r_{bp}^{p-1} is largest in absolute value; if $b \neq p$, then relabel row b as row p.

2. Identify the most negative† value (of dual variables) in column $L = 2(J + N) + 1$ of $\hat{R}$, designating that variable as the floating pivot variable, and its complementary nonbasic variable as the parametric variable; if no negative values are found, terminate the algorithm.

3. Let p be the column of the parametric variable, and find the minimum nonnegative ratio $\hat{r}_{iL}/\hat{r}_{ip}$ for i ranging over the rows of the basic primal variables and the row of the floating pivot variable (if this minimum is zero for some i perturbate $\hat{r}_{iL}$ by a small $\epsilon > 0$ and repeat; if no nonnegative ratios exist terminate with an unbounded solution); then designate the variable for which the minimum occurs as the basic pivot variable.

* Because the columns of the basic vectors are not needed, the pivoting operation for each p can be skipped for those $j < p$.

† We select the most negative value so that the objective function will increase fastest, at least initially, as the variable is driven to zero.

4. Execute the pivoting operation* on $\hat{R}$ for p the column of the parametric variable, and b the row of the basic pivot variable; then relabel row b as the row of the variable which is the parametric variable, and relabel column p as the column of the variable which is the basic pivot variable.
5. If the floating pivot variable and the basic pivot variable are the same, then return to Step 2; otherwise, identify the variable complementary to the basic pivot variable as the new parametric variable and return to Step 3.

---

*The calculations can be skipped for those $j \leq J + N$ and also, because nothing would be changed thereby, for those $\bar{j}$ for which $\hat{r}_{bj} = 0$ and those i for which $\hat{r}_{ip} = 0$.

3.6. The Computer Routine for Quadratic Programming

I have prepared a computer routine for quadratic programming which implements the Simplicial Algorithm as described here. Appendix C describes the technical details of this routine, shows how to use it, and presents the machine instructions in the FORTRAN language.

The main features of the routine are as follows. Unless modified appropriately, it requires that $J + N \leq 103$. Input data can be prepared in nearly any form suitable for the user, since it will accept variable format indications. Output is in a fixed format in which the optimal values of only the basic variables are given, and these are segregated into program variables and slack variables, and indexed in order. Also, a summary of the computations is given, indicating for example the number of iterations required. Any number of problems can be solved in sequence. As presently constituted, the routine requires an initial feasible basis identification as input, and no provision is made for checking that the A matrix is negative semidefinite.

The routine has been tested on approximately 120 test problems, of which

(1) 3 for which $J + N = 4$ were checked by hand,

(2) 56 for which $J = 12$, $N = 1$, were portfolio analysis problems prepared from actual data, and

(3) about 60 for which $J + N \geq 50$ were prepared using a random number generator assuming A to be a diagonal matrix.

For the most part, the routine is self-checking, since if the arithmetic is correct then finding all nonnegative values for the basic variables assures a correct solution.

Only Wolfe's and Beale's algorithms have been implemented in computer routines, and the computer routine for Wolfe's algorithm is the only one available for general use (on the IBM 704 and 709 series of machines).*

The Simplicial Algorithm for quadratic programming proposed here is very similar to Beale's algorithm; indeed, it can be shown that starting from the same initial basis they will proceed through the same sequence of bases to the solution. Nevertheless, Beale's theoretical justification differs substantially, and his definition of a basis is restrictive. Also, he is forced to execute pivoting operations in two parallel tableaux.† In perspective, the Simplicial Algorithm for quadratic programming can be construed as an extension of Beale's algorithm in which the use of the floating pivot variable and the concept of a semibasis make possible simpler and easier pivoting operations on a more compact tableau.

Wolfe's algorithm consists essentially of exploding the quadratic programming problem into a larger linear programming problem. This produces a tableau only slightly larger than the one here, and the computations required in each iteration are about the same. However, Wolfe says his algorithm requires an average of $4(J + N)$ interations, whereas experience indicates that $(J + N)$ is a conservative estimate for the number of iterations required by the Simplicial Algorithm. This is to be expected, since Wolfe loses considerable information about the structure of the problem when he converts it into a linear programming problem. However, a direct

† In remarks before the University of Chicago Symposium of Mathematical Programming, June 20, 1962, Dr. Wolfe said that Beale's algorithm appears to the most efficient in terms of computer time among those algorithms mentioned above. However, Dorn asserts that Beale's algorithm (as well as Lemke's and Theil and Van de Panne's) will not work for objective functions that are not negative definite.

* Markowitz's algorithm, which requires that the objective be negative definite, is specially designed for portfolio analysis, and has been used in routines for that purpose.

comparison between the two computer routines on identical problems has not been made as yet.

4. Concave Programming

Algebraic techniques are my concern in this section also. Here I extend the usefulness of the techniques developed for quadratic programming in Section 3 to more general concave programming by employing the technique of successive approximations.

In the special case of quadratic programming, linearity of the equations (2-6) in the Major Theorem permits refinement of the algebraic technique to a sequence of simple pivoting operations. In the general case, the difficulty of solving simultaneous nonlinear equations drastically alters the economics of computational efficiency. Nevertheless, by judiciously combining familiar approximation methods with the simplicial procedure, a relatively easy algorithm is still possible which differs in only a few respects from the computational scheme for quadratic programming.

4.1. Introduction

Quadratic programming is easy because linear equations obtain in the conditions for a solution. In fact, linear algebraic operations are virtually the only kind that are even feasible in large problems: when nonlinear equations occur the only recourse is to solve them by successive linear approximations.

In our case, quadratic functions in the programming problem yield linear equations in the conditions for a solution. Our job, therefore, is to construct a method for solving general problems by solving a succession of approximating quadratic problems.

A systematic procedure for this purpose, however, must be developed with computational efficiency a primary consideration, and a fact of the

situation is that complete reapproximations are burdensome and not very rewarding after the neighborhood of the solution has been reached. Consequently, we shall consider a modified approach in which only the linear fit of the approximation is updated after the first few times, but not the quadratic fit.

In the following discussions, the technique of quadratic approximations is reviewed and some notation is established, and then the computational procedure is described. Remarks on the conditions under which the approximations will converge are deferred until last. Sections 5 and 6 describe applications of the method in two special types of problems.

4.2. The Quadratic Approximation

According to the Equivalence Theorem of Kuhn-Tucker presented in Section 2, the concave programming problem is equivalent to a saddle problem with objective function

$$G(x; u) \equiv g(x) + u^t f(x) . \qquad (4\text{-}1)$$

Here the objective and constraint functions are not disjoint, and it is meaningful to approximate the problem by approximating $G(x; u)$ by another function. When we refer to an approximating problem, therefore, we shall mean that the objective function of the saddle problem has been approximated. In the following discussions it will always be assumed that $g(x)$, and each $f_n(x)$, are twice differentiable.

Recall from the introductory remarks above that we must consider successive quadratic approximations which are updated only in the linear fit after an initial (or perhaps several) quadratic fit. The situation will be as follows. Initially we will obtain a complete quadratic approximation

to $G(x; u)$ at a point $(x^*; u^*)$; thereafter, we will update the linear fit of the approximation at a new point $(x^k; u^k)$ for iteration $k = 2, 3, \ldots$ (by convention $(x^*; u^*) \equiv (x^1; u^1)$).

The quadratic approximation to $G(x; u)$ for a first-order fit at $(x^k; u^k)$, and a second-order fit at $(x^*; u^*)$, is a function ${}^*Q^k(x; u)$ of the same form as G would take for a quadratic programming problem, viz.,

$$ {}^*Q^k(x; u) \equiv p^t x + x^t Ax + u^t[c - Dx] + \text{Constant} , \qquad (4\text{-}2) $$

such that

(a) the first partial derivatives of G and ${}^*Q^k$ are the same at $(x^k; u^k)$, and

(b) the second partial derivatives are the same at $(x^*; u^*)$.

Differentiating in (4-1) and (4-2), we see that this definition requires the following equalities:†

$$ \begin{aligned} &\text{(a) First-Order Conditions, } \nabla G(x^k; u^k) = \nabla {}^*Q^k(x^k; u^k); \text{ viz.,} \\ &\quad \text{(i)} \quad p + (A + A^t)x^k - D^t u^k = \nabla g(x^k) + \nabla f(x^k)u^k ; \\ &\quad \text{(ii)} \quad c - Dx^k = f(x^k) ; \\ &\text{(b) Second-Order Conditions, } \nabla^2 G(x^*; u^*) = \nabla^2 {}^*Q^k(x^*; u^*); \text{ viz.,} \\ &\quad \text{(i)} \quad (A + A^t) = \nabla^2 g(x^*) + \Sigma_n u_n^* \nabla^2 f_n(x^*) ; \\ &\quad \text{(ii)} \quad -D^t = \nabla f(x^*) . \end{aligned} \qquad (4\text{-}3) $$

The second-order conditions determine the matrices A and D, and then with these in hand one can determine the vectors p and c from the first-order conditions.

---

† The iterated gradient operator ∇^2 is defined to yield the symmetric matrix of second partial derivatives of a scalar valued function; e.g., $\nabla^2 g(x) \equiv (\partial^2 g(x)/\partial x_i \partial x_j)$.

We shall define ${}^*R^k$ here just as we defined R in Section 3, except that we will append a $(J + N)$-dimensional identity matrix on the right for use in the approximating procedure:

$$ {}^*R^k \equiv \left[\begin{array}{cccc|c|cc} A + A^t & 0 & I_J & -D^t & -p & I_J & 0 \\ D & I_N & 0 & 0 & c & 0 & I_N \end{array}\right], \tag{4-4} $$

where A, D, p, and c are all determined by (4-3). Segregating the basic and nonbasic columns, this can be formulated as

$$ {}^*R^k \equiv \left[R_B^*, \quad R_N^*, \quad {}^*r^k, \quad I \right] \tag{4-5} $$

in obvious notation. The basic representation is then

$$ {}^*\hat{R}^k = \left[I, \quad R_B^{*-1} R_N^*, \quad R_B^{*-1} \cdot {}^*r^k, \quad R_B^{*-1} \right], \tag{4-6} $$

4.3. The Approximating Procedure

Depending upon whether or not a new second-order fit is obtained, we shall say that a major or minor reapproximation is made. In general, our methodology will be that for the first few times (until we reach the vicinity of the solution) we will make major reapproximations, and thereafter, only minor reapproximations.

The complete procedure for a major reapproximation is to take the existing values of the program variables (viz., an initial guess $(x^*; u^*)$, or whatever improved values may have been obtained in the course of the algorithm) and obtain a new representation ${}^*R^1$ by calculating A, D, p, and c via (4-3).

The procedure for a minor reapproximation is much easier. On hand in the course of the algorithm will be the current basic representation ${}^*\hat{R}^{k-1}$ and we must calculate ${}^*\hat{R}^k$. Only column $L = 2(J + N) + 1$ changes in a minor reapproximation, however, and from (4-6) the formula for the new values in this column is

$$ {}^{*}\hat{R}_{L}^{k} = R_{B}^{*-1} \cdot {}^{*}r^{k} , \qquad (4\text{-}7) $$

where L denotes the column index. Hence, we obtain the new values by calculating ${}^{*}r^{k}$ and multiplying it times the matrix existing in the last (J + N) columns of ${}^{*}\hat{R}^{k-1}$. Note that

$$ {}^{*}r^{k} = \begin{bmatrix} -p \\ c \end{bmatrix} \qquad (4\text{-}8) $$

is obtained from (4-3) using as $(x^{k}; u^{k})$ the values of the primal and dual program variables determined currently in the course of the algorithm, and using $(A + A^{t})$ and D as determined earlier in the algorithm at the last $(x^{*}; u^{*})$ at which a major reapproximation was made.

4.4. The Extended Simplicial Algorithm

In its purest form the algorithm consists merely of alternating between the approximating procedure and the Simplicial Algorithm for quadratic programming, the substance of the procedure being in each cycle to solve the quadratic problem obtained by a major reapproximation.

This form of the algorithm is inefficient because in each cycle it throws away a major investment consisting of an (approximately) optimal basis, and for that basis, a basic representation already calculated. It is to preserve this investment that we use minor reapproximations: even though convergence must then be slower in terms of the number of successive approximations, each reapproximation is much easier. It is also to preserve this investment that after a minor reapproximation we will attempt to use the existing basis and basic representation as the starting point in the Simplicial Algorithm.

In order to distinguish this procedure from the Simplicial Algorithm described in Section 2, we shall call it the Extended Simplicial Algorithm:

1. Select an initial basis and an initial guess $(x^*; u^*)$.

2. Major reapproximation procedure: calculate ${}^*R^1$ from (4-3).

3. Obtain ${}^*\hat{R}^1$ as in Step 1 of Figure (3-1).

4. Given ${}^*\hat{R}^k$, use the Simplicial Algorithm for quadratic programming to obtain the solution to the approximating problem, and let $(x^{k+1}; u^{k+1})$ represent the solution values of the primal and dual program variables.

5. If the desired number of major reapproximations has not been made, return to Step 2 with $(x^*; u^*) = (x^{k+1}; u^{k+1})$.

6. Minor reapproximation procedure: obtain ${}^*\hat{R}_L^{k+1}$ using (4-7).

7. Stopping procedure: if $\left|{}^*\hat{R}_L^k - {}^*\hat{R}_L^{k+1}\right|$ is sufficiently small, so that a predetermined accuracy requirement has been met, then terminate the algorithm.

8. If ${}^*\hat{R}_L^{k+1} \geq 0$ then return to Step 6; otherwise, return to Step 4 with ${}^*\hat{R}^{k+1}$.

Except for Step 8, all of the steps in this algorithm are straightforward or familiar.

In Step 8 we encounter a new situation: we might return to Step 4 with a basic primal variable that has become negative in the course of the approximation procedure in Step 6, whereas the Simplicial Alogirthm was not constructed to handle primal infeasibility. Remedies for this situation must, therefore, be our next topic.

4.5. Primal Infeasibility

Two recourses are open to us should we ever lose primal feasibility during the approximation procedure. The one which certainly will work is to return to Step 2 and start over either from the initial basis or from a redetermined initial basis which is feasible for the current approximation. This way, however, is laborious, so we will resort to it only if the easier method fails.

The other method, the easier one, is to modify the Simplicial Algorithm so that if <u>either</u> a primal or dual basic variable is negative, then it will be driven to zero using the simplicial procedures. One change in the simplicial procedures, however, is necessary in the case of primal infeasibility.† The second critical value ($\bar{\bar{z}}_p$) of the parametric variable, that value at which it is no longer "feasible" to increase the parametric variable further, is determined as the first value at which <u>any</u> variable which has been positive becomes negative. Although a proof that this method will always work has not been obtained,†† in the general case of primal or dual infeasibility it has the rationale that by systematically driving negative variables to zero, without letting any other variable become negative, we will eventually eliminate all negative variables.

In any case, when primal infeasibility occurs this extension of the simplicial procedures is mandatory, because then it is possible that the values of the basic primal variables are determined by the relation $-f(x) + s = 0$ in (2-6) independently of the values of the dual variables.

---

† I conjecture that this change is desirable also for dual infeasibility.

†† Over a hundred test problems of large size, $J + N \geq 50$, have been solved without ever failing to maintain primal feasibility by this method. However, my computer routines have provisions in them to start over from the initial basis if this method does not regain primal feasibility in $J + N$ iterations of the simplicial procedures.

Consequently, increasing the nonbasic dual variable which is complementary to the negative primal variable will not suffice to drive it to zero unless we allow semibases to occur for which a nonbasic primal variable is the parametric variable. This is done by allowing the nonbasic dual variable to replace a basic dual variable which becomes negative.

4.5. Efficiency and Convergence

A quadratic approximation has at least one thing to recommend it: using the approximating quadratic problem one can search easily for an approximately optimal basis, giving a significant headstart in the Simplicial Algorithm discussed in Section 2. Whether or not it is worthwhile to continue with successive quadratic approximations might seem to be in doubt were it not that computational experience to be reported in Sections 5 and 6 indicates that, besides being the easiest thing to do, it is also efficient. In addition, maintaining primal feasibility by the modification of the Simplicial Algorithm discussed above has been successful.

Convergence of the approximating procedure has not been found to be a serious problem in practice from a computational point of view: first, because in practice it is usually easy to make an initial guess $(x^*; u^*)$ which is at least within an order of magnitude of the optimal values; and second, experience and theory both indicate that, at least in the types of (nonpathological) problems of primary interest, the neighborhood of the optimal values in which the procedure will converge is relatively large. The first point is idealized in the problem of linear programming under uncertainty to be discussed in Section 5, for which the

structure of the problem permits one to make a very good guess from a simple rule of thumb.

The theoretical evidence that the neighborhood of convergence is usually large is based upon the presumption that $\nabla^2 g$, $\nabla^2 f_n$, and ∇f do not change radically in the neighborhood of an approximation. The argument can be presented briefly as follows. A quadratic approximation is equivalent to approximating the functions in the equation (2-6) linearly. If we represent these equations abstractly as $h(x^0) = 0$, then in the k-th iteration the approximation procedure amounts to approximating $h(x)$ by

$$*h^k(x) \equiv [h(x^k) - \nabla h(x^*)^t x^k] + \nabla h(x^*)^t x . \qquad (4\text{-}9)$$

Hochstrasser shows that successive approximations of this type, without allowing for the simplicial procedures, will converge if x^* and all x^k are in a neighborhood of x^0 for which all of the eigenvalues of

$$I - [\nabla h(x^*)^t]^{-1} \nabla h(x) \qquad (4\text{-}10)$$

are less than unity in absolute value for every x in the neighborhood. As an approximate condition we have that all of the eigenvalues of

$$I - [\nabla h(x^*)^t]^{-1} \nabla h(x^0) \qquad (4\text{-}11)$$

must be less than unity in absolute value. Certainly convergence is assured if we choose $x^* = x^0$, because (4-11) will then be of the form I-I to yield the zero matrix, and there will be a neighborhood of x^0 in which convergence is assured. If ∇h is not greatly variable, then this neighborhood will be large; the ultimate occurs when h is linear (quadratic programming), for then ∇h is constant and (4-11) reduces to the zero matrix. Equivalently, the neighborhood is large if $\nabla^2 g$, $\nabla^2 f_n$, and ∇f_n, all of which enter into the composition of ∇h, are not greatly variable.

It should be noted that limitations of the algorithm as regards convergence are inherent in dealing with nonlinear equations. The limitations of the theory and methods of solving nonlinear equations are an impasse to which one working in the theory and methods of programming must be reconciled. Although the computer routines described in Sections 5 and 6 have never failed to converge for test problems, it must be admitted that should they ever fail, the only recourse is to start over with another (better) initial guess.

4.6. Other Methods

Recent surveys by Wolfe (1962) and Dorn (1963) describe the other methods that have been proposed for general concave programming. Of these, the most prominent are the gradient-projection method of Rosen, the cutting-plane method of Kelley, and the method of feasible directions of Zoutendijk. Only the first of these has been implemented in a computer routine, and then only for linear constraints: the routine is not available for general use. Comparisons with the method proposed here are not possible at present, at least in terms of computational efficiency. It should be mentioned here, however, that Kelley's method will not work for nonlinear constraints, mainly because in effect he omits the terms $\Sigma_n u_n^* \nabla^2 f_n(x^*)$ in (4-3).

In industrial practice, concave programming problems have usually been solved by the laborious process of approximating nonlinear functions by piecewise linear functions (cf. Miller, and also the comments of Dorn).

5. A Linear Programming Problem Under Uncertainty

A special case of concave programming which is of significant interest is a problem of linear programming under uncertainty. The problem is formulated in this section, and cast into the computational format of the Extended Simplicial Algorithm. It serves also as a vehicle to demonstrate precisely how the Extended Simplicial Algorithm is implemented in a computer routine.

5.1. Formulation of the Problem†

The problem that we shall consider is a generalized inventory stocking problem, of the type commonly known as the continuous "newsboy problem". In the simplest case there is only one product and a decision must be made as to how much inventory, say y, of this product is to be stocked. The demand for this product is not known with certainty at the time of the stocking decision, but we consider its probability distribution to be known: using a tilde to denote a random variable, $\tilde{d}$ will be the random variable of demand.††

The economic structure of the problem in this simplest case is given by a penalty function, $P(\Delta)$, which specifies the loss incurred when a discrepancy $\Delta = d - y$ obtains between actual demand and available inventory. For example, when demand exceeds supply the opportunity loss consists of lost sales, and goodwill; also, when the inventory stock exceeds demand there will be an inventory carrying charge on the excess. In this form,

---

† Several formulations of problems of linear programming under uncertainty are extant, among which are the ones of Dantzig (1955), and Charnes and Cooper. The formulation here differs substantially from both of these, however.

†† We will assume throughout that y is a real-valued variable, and that $\tilde{d}$ has a continuous probability density function.

the problem is to minimize $EP(\tilde{d} - y)$ for $y \geq 0$, where E is the expectation operator over the distribution of $\tilde{d}$. When P represents a V-shaped proportional loss structure,

$$P(d - y) = \alpha \, \text{Max}(0, d - y) + \beta \, \text{Max}(0, y - d) ,$$

where α is the unit underage cost and β is the unit overage cost, a well-known result (cf. Schlaifer, Chapter 4) is that the optimal choice y° is $y^{\circ} = \text{Max}(0, \hat{y}^{\circ})$ for $\hat{y}^{\circ}$ the solution to

$$\text{Prob}(\tilde{d} < \hat{y}^{\circ}) = \alpha/[\alpha + \beta] .$$

Here we shall generalize the economic structure of this problem to include multiple products, and we shall include an analysis of the programming problem of finding the best way in which to produce the vector of amounts y° required for inventory. Our interest, however, will be confined to the computational problems of solution, and will not enter into the statistical problems. In the more general inventory problem, we will assume the production and economic structures to be linear, as in linear programming, and confine ourselves to a one-stage problem.

The production structure consists of a set of activities indexed by j and we will let x denote the J-dimensional vector of activity levels to be chosen. Then besides $x \geq 0$, the choice of x is restricted by N linear constraints $Dx \leq c$. The output from the production activities is given by a linear production function Ax, where A is an I by J matrix of production coefficients. Of this amount, an I-dimensional vector y will be selected for inventory and the remainder thrown away, so that $0 \leq y \leq Ax$.

The economic structure consists of a J-dimensional vector q of costs per unit level of the activities, and an I-dimensional vector p of

prices for the products. Thus when a program x of activity levels is chosen and a vector y of outputs is put into inventory, the net revenue is

$$\Sigma_i p_i \text{ Min}(d_i, y_i) - q^t x = p^t y - q^t x - \Sigma_i p_i \text{ Max}(0, y_i - d_i) .$$

The term $\Sigma_i p_i \text{ Max}(0, y_i - d_i)$ represents an adjustment for discrepancies between demands and stocking levels. Additional adjustments might be necessary to reflect lost goodwill, inventory charges, etc.; we summarize all the adjustments in a penalty function $P(\Delta)$ for $\Delta \equiv d - y$. Letting E be the expectation operator over the distribution of $\tilde{d}$, the expected penalty is given by $EP(\tilde{d} - y)$.

Hence, we have a decision problem in which the decision maker has to make two sets of choices: (1) the production levels x, (2) the inventory levels y. This defines

A Linear Programming Problem Under Uncertainty: find vectors x and y yielding the maximal value of

$$g(x, y) \equiv p^t y - q^t x - EP(\tilde{d} - y)$$

subject to $x \geq 0$, $y \geq 0$,

$$y \leq Ax ,$$

and

$$Dx \leq c ,$$

where the penalty function P is convex.

We assume that the (conditional) penalty function is convex so that the objective will be concave, making this problem a special case of the concave programming problem. In addition, in order to satisfy the differentiability conditions assumed in Section 4, we assume that the random

vector $\tilde{d}$ of demands has a continuous density function. We shall refer to this problem as LPUU in the discussions below.

After transforming the data of the LPUU problem into the format of the Extended Simplicial Algorithm, we will consider in detail the special case in which P represents proportional loss structures for the various products. Assuming these structures to be independent makes it possible to use only the marginal probability distributions of the unknown demands.

5.2. The Extended Simplicial Algorithm for Linear Programming Under Uncertainty

In order to transform the data of the LPUU problem into the format of the Extended Simplicial Algorithm, we need to specify the matrix ${}^{*}R^{k}$ which is the principal datum in the algorithm. Care must be exercised, however, because the notation that we have adopted for the LPUU is not consistent with the notation of Section 4. Also, there are two primal program vectors (x and y) to be determined and our notation must keep them separated.

Consider first the constraints. They are linear, and in the canonical form of Section 4 they become,

$$\begin{bmatrix} -A & I \\ D & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \leq \begin{bmatrix} 0 \\ c \end{bmatrix}. \qquad (5\text{-}1)$$

Because the second-order partial derivatives of the constraint function are zero, a quadratic approximation to only the objective function suffices in the algorithm, and the data of (5-1) enter ${}^{*}R^{k}$ just as they would in a quadratic programming problem.

For the quadratic approximation to the objective function we calculate the following first and second order partial derivatives:

$$\nabla_x g(x; y) = -q \; ; \qquad \nabla_y g(x; y) = p + \nabla EP(\tilde{d} - y) \; ;$$

$$\nabla^2_{xx} g(x; y) = 0 \; ; \qquad \nabla^2_{yy} g(x; y) = -\nabla^2 EP(\tilde{d} - y) \; ; \tag{5-2}$$

$$\nabla^2_{xy} g(x; y) = 0 \; .$$

Hence, in the notation of (4-5)

$$*r^k = \begin{bmatrix} q \\ *w^k \\ 0 \\ c \end{bmatrix} , \tag{5-3}$$

where the I-dimensional vector $*w^k$ is defined by interpreting in (4-3) and (4-4):

$$*w^k = -p - \nabla EP(\tilde{d} - y^k) - \nabla^2 EP(\tilde{d} - y^*)\, y^k \; . \tag{5-4}$$

In terms of the discussion in Section 4, (5-4) reflects a quadratic approximation to the nonlinear term $EP(\tilde{d} - y)$ in the objective function, for which a full second-order fit was last obtained at y^* and which has been updated since then to a first-order fit at y^k.

The full tableau (4-4), with the columns appropriately labeled, is as follows, using (5-1) and (5-2) and letting (5-4) define $*w^k$:

$$*R^k = \left[\begin{array}{cccccccc|c|cccc} x & y & s^h & s^c & v^x & v^y & u^h & u^c & & & & & \\ 0 & 0 & 0 & 0 & I_J & 0 & A^t & -D^t & q & I_J & 0 & 0 & 0 \\ 0 & -\nabla^2 EP(d,y^*) & 0 & 0 & 0 & I_I & -I_I & 0 & *w^k & 0 & I_I & 0 & 0 \\ -A & I_I & I_I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I_I & 0 \\ D & 0 & 0 & I_M & 0 & 0 & 0 & 0 & c & 0 & 0 & 0 & I_M \end{array}\right] \tag{5-5}$$

With $*R^k$ in hand there is little more to be said beyond the instructions for the algorithm given in Section 4: (1) a major reapproximation is accomplished by recalculating a new tableau (5-5); (2) the Simplicial

Algorithm, performed on the basic representation $*\hat{R}^k$ obtained from (5-5), is as described in Section 3; (3) a minor reapproximation amounts to using the current solution for the optimal value of y as y^k in (5-4) to determine $*w^k$, and then multiplying

$$\begin{bmatrix} q \\ *w^k \\ 0 \\ c \end{bmatrix} \tag{5-6}$$

times the inverse of the basic matrix (stored in the last J + N columns of $*R^k$) to get the new value of column L = 2(J + N) + 1 in $*R^k$.

5.3. Proportional Losses

In a special case of practial interest the penalty function P reflects a proportional, or V-shaped, loss structure. For example, if demand exceeds production, then the penalty, representing lost sales and goodwill, might be proportional to the discrepancy by a factor α. Similarly, when production exceeds demand the cost of disposal or the inventory carrying charge might be proportional to the overage by a factor β.

In the case of several products indexed by i = 1, 2,...,I, we consider a penalty function reflecting independent proportional loss structures for the various products:

$$P(\tilde{d} - y) \equiv \Sigma_i \{\alpha_i \text{Max}(0, \tilde{d}_i - y_i) + \beta_i \text{Max}(0, y_i - \tilde{d}_i)\} . \tag{5-7}$$

For this function,

$$\partial\, EP(\tilde{d} - y)/\partial y_i = -\alpha_i \text{ Prob}(\tilde{d}_i > y_i) + \beta_i \text{ Prob}(\tilde{d}_i \leq y_i) , \tag{5-8}$$

and,

$$\partial^2 EP(\tilde{d} - y)/\partial y_i\, \partial y_k = \begin{cases} 0 & \text{if } i \neq k \\ [\alpha_i + \beta_i] \text{ Prob}(\tilde{d}_i = y_i) \end{cases} . \tag{5-9}$$

5.4. The Computer Program for LPUU with Proportional Losses

A computer routine has been prepared which implements the Extended Simplicial Algorithm for LPUU with proportional losses. Currently the routine is prepared to handle only Normal probability distributions; however, slight changes will enable it to handle any other tabulated distribution. Appendix C describes the technical features of this routine, shows how to use it, and presents the machine instructions in the FORTRAN language.

The main features of the routine are as follows. Without modification it will accept problems for which $J \leq 50$, $I \leq 30$, $N \leq 30$, provided $J + 2I + N \leq 80$. Variable format provisions allow the user to prepare input data in nearly any form that is convenient. Output is in a fixed format in which the optimal values of the basic variables are given in order, but segregated by program and slack variables. A summary of the computations follows which gives the number of successive approximations required to obtain a degree of accuracy specified by the user, and also the number of pivoting operations required. Any number of problems can be solved in sequence. As presently written, the routine requires an initial feasible basis as an input datum. A rather complete set of error indications is built into the routine, including checks that the objective is concave ($\alpha_i + \beta_i \geq 0$ for each i). Also, a maximum number of allowable approximations can be specified by the user; e.g., to halt diverging approximations. Provision is also made for the user to specify the number of major re-approximations (complete quadratic fits) to be made.

It is not necessary to make an initial guess y^* for this routine: as presently written the routine automatically takes y_1^* as the sum of

the mean and standard deviation of $\tilde{d}_i$. The reason for this is that among the conditions (cf. (5-5)) for a solution we find that

$$u_i^h = \alpha_i - [\alpha_i + \beta_i] \text{ Prob}(\tilde{d}_i < y_i) \tag{5-10}$$

when y_i is not constrained to zero. Now assuming α_i to be larger than β_i and u_i^h to be an order of magnitude smaller than α_i, (5-10) in the form

$$\text{Prob}(\tilde{d}_i < y_i) = \frac{\alpha_i - u_i^h}{\alpha_i + \beta_i} \tag{5-11}$$

is approximately satisfied by choosing y_i^* as described above.

The computer routine has been tested on approximately 70 test problems, of which

1) 1 for which $J = 3$, $I = 1$, $N = 2$,
   2 for which $J = 3$, $I = 2$, $N = 1$, and
   3 for which $J = 10$, $I = 5$, $N = 5$,
   were checked by hand, and
2) over 60 for which $J \geq 10$, $I \geq 5$, $N \geq 5$, among which were problems reaching the capacity of the routine, were prepared using a random number generator.

The routine is largely self-checking, since finding all nonnegative values for the basic variables assures a correct solution provided the arithmetic has been correct. However, no measures have been taken to guard against deceptive appearances of convergence in the approximating procedure: the analysis of convergence has not been carried far enough to ascertain what pathological possibilities exist in this regard.

None of the test problems have required as many as $(J + 2I + N)$ pivoting operations. The total number of approximations required, however, has varied widely depending upon the number of major reapproximations specified. An example will suffice to illustrate the situation: for a problem with $J = 10$, $I = 5$, $N = 5$, and the convergence criterion that the maximum difference between the values of the basic variables in successive approximations be less than 10^{-5}, the following results were obtained in successive tests:

Number of Major Reapproximations	Number of Minor Reapproximations	Total Number of Approximations
1	27	28
2	7	9
3	2	5
5	0	5

(The number of major reapproximations includes the initial approximation.) The indications are, therefore, that selection of the number of major reapproximations plays an important role in determining computational efficiency. An "optimal" tradeoff between major and minor reapproximations has not been investigated, although 2 for small problems and 3 for large problems seems a good rule of thumb.

6. Variable-Factor Programming

Another special case of concave programming to which the Extended Simplicial Algorithm can be applied is variable-factor programming, which is of significant interest in economic planning and in industrial scheduling. In this section I formulate the problem of variable-factor programming, and show how the Extended Simplicial Algorithm is implemented to solve the problem in a computer routine.

6.1. Formulation

In variable-factor programming we consider situations in which both variable and fixed factors of production play a role, and in which not only a program vector of activity levels, but also a program of variable-factor inputs must be decided upon. Agricultural planning will be the context of the formulation here, but another important context is scheduling of oil refinery operations (cf. Manne).

Let $x \equiv (x_j)$ be the J-dimensional program vector of activity levels, specifying the acreages to be planted in various crops indexed by j. Besides the natural constraint $x \geq 0$, limitations of land available and of other fixed factors of production enforce N fixed-factor constraints $Dx \leq c$.

Now the net yield (in monetary units) from an acre of crop j depends upon the M-dimensional vector $z^j \equiv (z^j_m)$ of variable-factor amounts allocated to it, where m indexes the variable factors of production such as man-hours of cultivation, fertilizer, and irrigation water. Letting this dependency be expressed by a yield function $y_j(z^j)$, the total yield from the agricultural program is given by $\Sigma_1 \; x_j \; y_j(z^j)$.

Finally, letting b be the M-dimensional vector of variable-factor supplies, the problem can be defined as follows,

Variable-Factor Programming: find vectors x and z^j, $j = 1,\ldots,J$, yielding the maximal value of

$$g(x, Z) = \Sigma_j x_j y_j(z^j)$$

subject to $x \geq 0$, $z^j \geq 0$,

$$Dx \leq c ,$$

and

$$\Sigma_j x_j z^j \leq b .$$

This problem will be referred to as the VFP problem in the discussions below.

We will assume that each of the functions $y_j(z^j)$ is concave; nevertheless, the variable-factor constraints are not concave, so this problem is not a special case of the concave programming problem. Even so, we know from the theorems of Kuhn and Tucker that any solution to the saddle problem is a solution to the programming problem even without concavity,† and in the following discussions we shall find that we can obtain a solution to the saddle problem by finding a solution to a certain problem (called the dual problem) which is a concave programming problem.

---

† As mentioned in Section 2, p. 11, the concavity assumption was not needed by Kuhn and Tucker in order to show that any solution $(x^*; u^*)$ to the saddle problem with

$$G(x; u) = g(x) + u^t f(x)$$

yields x^* as a solution to the programming problem.

6.2. Reduction of the Problem Via the Saddle Problem

A characteristic feature of the VFP problem is the large number of variables involved; consequently, it is desirable to reduce the problem by a preliminary transformation derived from the saddle problem.

For variable-factor programming the objective function in the saddle problem (cf. Section 2) takes the form,

$$G(x, z; u, v) \equiv \Sigma_j x_j[y_j(z^j) - v^t z^j] + u^t[c - Dx] + v^t b, \qquad (6\text{-}1)$$

and we see that given a value of v we can maximize for the z^j independently of the other variables. Hence, defining new functions

$$\bar{y}_j(v) \equiv \max_{z^j \geq 0} [y_j(z^j) - v^t z^j] , \qquad (6\text{-}2)$$

we obtain a simpler saddle problem with objective function,

$$\bar{G}(x; u, v) \equiv x^t[\bar{y}(v) - D^t u] + u^t c + v^t b , \qquad (6\text{-}3)$$

where $\bar{y}$ is the vector of functions (6-2). But this is just the saddle problem of another programming problem in which the role of the primal and dual variables are reversed; viz., the

Dual VFP Problem: find vectors u and v yielding the minimal value of

$$u^t c + v^t b$$

subject to $u \geq 0$, $v \geq 0$, and

$$\bar{y}(v) \leq D^t u .$$

By eliminating the variable-factor programs z^j from independent consideration, the dual VFP problem reduces the computational burden and the data storage requirements substantially.

In order to solve the dual VFP problem using the Extended Simplicial Algorithm, it is necessary that the nonlinear functions $\bar{y}(v) \equiv (\bar{y}_j(v))$ be convex: it can be verified from (6-2) that $\bar{y}_j(v)$ will be convex if $y_j(z^j)$ is concave.

Retracing our steps from the dual VFP problem to the primal VFP problem, we can establish that a solution to the dual problem yields a solution to the primal problem as follows. Given a solution (u^o, v^o) to the dual problem, the Equivalence Theorem shows that there must exist a vector (of dual variables) x^o such that $(x^o; u^o, v^o)$ solves the saddle problem with objective function $\bar{G}$ in (6-3), and in the course of the computations we can actually obtain x^o. Now since $\bar{G}$ was obtained by maximizing G in (6-1) with respect to z^j, there will exist a matrix $z^o \equiv (z^{jo})$ of vector values for which (6-2) is an attained maximum for $v = v^o$, and $(x^o, Z^o; u^o, v^o)$ must solve the saddle problem with objective function G in (6-1). Finally, we know (Section 2, p. 11, first line) that this solution must yield (x^o, Z^o) as a solution to the primal VFP problem.

6.3. The Extended Simplicial Algorithm for Variable-Factor Programming

As in Section 5 for linear programming under uncertainty, our job here is to transform the data of the dual VFP problem into the format of the Extended Simplicial Algorithm. For this it suffices to find the form of the tableau representation ${}^{w}R^k$ defined in (4-4) and (4-3). Some translation of notation is necessary also.

Referring to (6-3), we can calculate the first and second order partial derivatives needed for the approximation schema. Those that are not identically zero are as follows:

$$\nabla_x \bar{G} = \bar{y}(v) - D^t u \; ; \qquad \nabla_u \bar{G} = c - Dx \; ; \qquad \nabla_v \bar{G} = b + \nabla \bar{y}(v)x \; ;$$

$$\nabla^2_{xu} \bar{G} = -D \; ; \qquad \nabla^2_{xv} \bar{G} = \nabla \bar{y}(v)^t \; ; \qquad \nabla^2_{vv} \bar{G} = \Sigma_j x_j \nabla^2 \bar{y}_j(v) \; . \tag{6-4}$$

Turning to the formation of the tableau, we insert the information from (6-4) into place according to (4-4) and (4-3) as follows, where the column headings are self-explanatory except that w denotes the vector complementary to x:

$$ {}^*R^k = \left[\begin{array}{ccccccc|c|ccc} u & v & w & s^u & s^v & x & & & & & \\ 0 & 0 & 0 & I_N & 0 & D & ; & c & I_N & 0 & 0 \\ 0 & -\Sigma_j x_j^* \nabla^2 \bar{y}_j(v^*) & 0 & 0 & I_M & -\nabla \bar{y}(v^*) & ; & {}^*b^k & 0 & I_M & 0 \\ -D^t & \nabla \bar{y}(v^*)^t & 0 & 0 & 0 & 0 & ; & {}^*a^k & 0 & 0 & I_T \end{array}\right] , \tag{6-5}$$

for,

$$ {}^*b^k = b + \Sigma_j x_j^* \nabla^2 \bar{y}_j(v^*)v^k + [\nabla \bar{y}(v^k) - \nabla \bar{y}(v^*)]x^k \; , \tag{6-6}$$

and

$$ {}^*a^k = \bar{y}(v^k) - \nabla \bar{y}(v^*)^t v^k \; . \tag{6-7}$$

This completes the formulation of the problem in terms of the Extended Simplicial Algorithm provided we know how to compute $\nabla \bar{y}$ and $\nabla^2 \bar{y}$.

6.4. Computing $\nabla \bar{y}_j$ and $\nabla^2 \bar{y}_j$

In order to find formulas for computing $\nabla \bar{y}_j$ and $\nabla^2 \bar{y}_j$, we delve further into the relationship between the functions y_j and $\bar{y}_j$ defined by (6-2).

Let $z^j(v)$ be the value of z^j at which the maximum occurs in (6-2) given v; then, differentiating formally in (6-2) yields

$$\nabla \bar{y}_j(v) = \nabla z^j(v) \nabla y_j(z^j(v)) - z^j(v) - \nabla z^j(v)v \; . \tag{6-8}$$

However, a necessary condition for the maximum in (6-2) is that

$$\nabla y_j(z^j(v)) \leq v \ , \tag{6-9}$$

with strict inequality occurring for the k-th component only if z_k^j is constrained to zero, in which case $\nabla z_k^j(v) = 0$. Hence, (6-8) reduces to

$$\nabla \bar{y}_j(v) = -z^j(v) \ , \tag{6-10}$$

and to calculate $\nabla \bar{y}_j(v)$ we must solve (6-2) for the optimal $z^j(v)$ given v.

An easy special case occurs when y_j is separable into

$$y_j(z^j) = \Sigma_m \, y_{jm}(z_m^j) \ . \tag{6-11}$$

For, in (6-2) an attained maximum is then defined by the univariate conditions that, unless constrained to zero, $z_m^j(v)$ is determined by

$$\partial y_{jm}(z_m^j)/\partial z_m^j = v_m \ .$$

Hence, letting $h_{jm}(\cdot)$ be the inverse function $(\partial y_{jm}/\partial z_m^j)^{-1}$,

$$z_m^j(v) = \text{Max}(0, \, h_{jm}(v_m)) \ , \tag{6-12}$$

which yields (6-10) in an appropriate form for calculation.

Considering only the separable case (6-11), we can calculate $\nabla^2 \bar{y}_j$ from (6-10) and (6-12) as

$$\partial^2 \bar{y}_j(v)/\partial v_k \, \partial v_m = \begin{cases} 0 \text{ if } k \neq m \\ 0 \text{ if } k = m \text{ but } z_m^j(v_m) \text{ is constrained to zero} \\ \nabla h_{jm}(v_m) \text{ otherwise} \end{cases} \ . \tag{6-13}$$

From (6-12) and (6-13) it is clear that a necessary analytical step before the problem is ready for calculations is inversion of the functions ∇y_{jm}, and also finding the first derivative of the inverse function.

6.5. The Computer Routine for Variable-Factor Programming

I have prepared a computer routine which implements the Extended Simplicial Algorithm for variable-factor programming. Appendix C describes the technical details of this routine, shows how to use it, and presents the machine instructions in the FORTRAN language.

This routine has been tested sufficiently for only one type of function $y_j(z^j)$; namely, a quadratic function

$$y_j(z^j) = p_j\left[1 - q_j\left(1 - \frac{\Sigma_k\, a_{mj}\, z^j_m}{s_j}\right)^2\right] - c_j\,, \qquad (6\text{-}14)$$

for J = 11, N = 15, K = 2, occurring in an actual agricultural planning problem.† For this problem it was possible to check the results against answers obtained by other means.

I have, in addition, made provisions in the routine for it to handle separable functions $y_j = \Sigma_k\, y_{jm}$ for which each y_{jm} is of one of the following types, where a, b, c, and d are parameters and a > 0 :

$$\begin{aligned} &(az + b)^c + d\,, && 0 < c < 1\,; \\ &-(az + b)^c + d\,, && c < 0\,; \\ &a \log (z + b) + d\,; \\ &e^{az+b} + d\,. \end{aligned} \qquad (6\text{-}15)$$

Although the routine appeared to solve some ten test problems prepared using a random number generator, I have not checked these results by hand calculations.

† Professor Robert Dorfman introduced me to this problem, and provided the data for the analysis.

Unless modified, the routine requires that $J \leq 50$, $N \leq 40$, $M \leq 20$, and $J + N + M \leq 80$. Otherwise, its main features are similar to the other two routines.

APPENDIX A

Solution of a Numerical Example by the Simplicial Algorithm

In this appendix a numerical example is solved using the Simplicial Algorithm described in Section 2. Although a computer routine to solve this example would apply Newton's method, as described in Section 4, the example is not solved that way here, in order to illustrate the algebraic manipulations required by the general algorithm. I am indebted to Professor John Bishop for providing this example.

Let $J = 2$ and $N = 2$, and define

$$\begin{aligned} g(x_1, x_2) &\equiv x_2 - (x_1 + x_2 - 1)^4 \\ f_1(x_1, x_2) &\equiv 4x_1 - 3x_2^2 + 4 , \\ f_2(x_1, x_2) &\equiv -2x_1^2 + x_2 + 1 . \end{aligned} \tag{A-1}$$

The simultaneous equations (2-6) take the following form:

$$\begin{aligned} -4(x_1 + x_2 - 1)^3 \qquad\qquad + v_1 \quad + 4u_1 - 4u_2x_1 &= 0 \\ -4(x_1 + x_2 - 1)^3 \qquad\qquad + v_2 - 6u_1x_2 + u_2 &= -1 \\ -4x_1 + 3x_2^2 \qquad + s_1 \qquad\qquad\qquad\qquad &= 4 \\ 2x_1^2 - x_2 \qquad\qquad + s_2 \qquad\qquad\qquad &= 1 \end{aligned} \tag{A-2}$$

Since the origin is feasible,

$$\begin{aligned} f_1(0, 0) &= 4 > 0 , \\ f_2(0, 0) &= 1 > 0 , \end{aligned} \tag{A-3}$$

it suffices to take an initial basis for which the slack variables are all basic; viz.,

$$B_0 = (3, 4, 5, 6) . \tag{A-4}$$

Examining (A-2) with $x_1 = x_2 = u_1 = u_2 = 0$, the following values of the basic variables are immediate:

$$\hat{s}_1 = 4, \qquad \hat{s}_2 = 1, \qquad \hat{v}_1 = -4, \qquad \hat{v}_2 = -5. \tag{A-5}$$

Choosing to drive the basic dual variable v_2 to zero, we calculate the critical values of its complementary nonbasic variable x_2 by letting x_2 be a parameter in (A-2). The second equation in (A-2) indicates that

$$-4(\bar{x}_2 - 1)^3 = -1 ,$$

or, (A-6)

$$\bar{x}_2 = 1 + \sqrt{1/4} .$$

The third and fourth equations offer the alternatives,

$$\bar{\bar{x}}_2 = \sqrt{4/3} , \qquad \bar{\bar{x}}_2 = -1 , \tag{A-7}$$

the second of which (being negative) is extraneous. Since $\bar{\bar{x}}_2 < \bar{x}_2$ we make s_1 nonbasic and x_2 basic. This change of basis identifies u_1 as the new parameter to be varied in (A-2). We now have

$$\hat{v}_2 = -1 + 4(\sqrt{4/3} - 1)^3 , \tag{A-8}$$

which is still negative, and we continue to drive $\hat{v}_2$ to zero by increasing u_1. From the second equation in (A-2) we see that this is accomplished for

$$\bar{u}_1 = [-4(\sqrt{4/3} - 1)^3 + 1]/6\sqrt{4/3} . \tag{A-9}$$

Since we are at a vertex of the feasible subspace, identified by the fact that the basic primal variables (x_2 and s_2) are completely determined by the last two equations of (A-2), we recognize that $\bar{\bar{u}}_1$ is unbounded. Hence we make v_2 nonbasic and u_1 basic, and then repeat the procedure.

The basis for the next iteration is

$$B_1 = (2, 4, 5, 7) , \tag{A-10}$$

and from the previous analysis we have that $\hat{u}_1 = \bar{u}_1$ in (A-9). The other basic dual variable is v_1, for which the first equation of (A-2) yields

$$\hat{v}_1 = 4[(\sqrt{4/3} - 1)^3 - \hat{u}_1] < 0 . \tag{A-11}$$

Observing that this is negative, we shall drive $\hat{v}_2$ to zero. The critical values are obtained from (A-2) with $s_1 = v_2 = u_2 = 0$ and x_1 as a parameter; viz., from the first three equations

$$\begin{aligned} -4(\bar{x}_1 + \hat{x}_2 - 1)^3 + 4\hat{u}_1 &= 0 , \\ -4(\bar{x}_1 + \hat{x}_2 - 1)^3 - 6\hat{u}_1 \hat{x}_2 &= -1 , \\ -4\bar{x}_1 + 3\hat{x}_2^2 &= 4 , \end{aligned} \tag{A-12}$$

the first two of which reduce immediately to

$$-4(\bar{x}_1 + \hat{x}_2 - 1)^3 + 4/[4 + 6\hat{x}_2] = 0 . \tag{A-13}$$

It can be verified from these equations that $\bar{x}_1 < 1/3$. On the other hand, the last equation in (A-2) yields $s_2 > 0$ when $x_1 = 1/3$, $x_2 = 4/3$, satisfying the third equation. Hence, $\bar{x}_1 < \bar{\bar{x}}_1$ and we make v_1 nonbasic to be replaced by x_1 as basic.

The basis now contains only u_1 as a basic dual variable. But from (A-12) we see that

$$4\hat{u}_1 = 1 - 6\hat{u}_1 \hat{x}_2 , \tag{A-14}$$

or,

$$\hat{u}_1 = 1/[4 + 6\hat{x}_2] ,$$

which is strictly positive. Hence, we have found the optimal basis. The optimal values of the basic variables are given by (A-2); viz., using (A-13) and (A-14),

$$u_1^\circ = 1/[4 + 6x_2^\circ] ,$$

and

$$s_1^\circ = 1 - 2x_1^{\circ 2} + x_2^\circ ,$$

for x_1° and x_2° determined by the simultaneous equations,

$$-(x_1^\circ + x_2^\circ - 1)^3 + 1/[4 + 6x_2^\circ] = 0 ,$$

$$-4x_1^\circ + 3x_2^{\circ 2} - 4 = 0 .$$

An approximate solution is given by $x_1^\circ \doteq 1/3$, $x_2^\circ \doteq 4/3$, both values being somewhat too large.

APPENDIX B

A Numerical Example of Quadratic Programming Solved by the Simplicial Algorithm

In this appendix I solve a small quadratic programming problem using the Simplicial Algorithm as described in Section 3.

Let J=2, N=2, and in the notation of (3-2),

$$p = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad A = \begin{bmatrix} -1 & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} \end{bmatrix}, \quad c = \begin{bmatrix} 2 \\ 3 \end{bmatrix}, \quad D = \begin{bmatrix} 1 & 2 \\ 3 & 2 \end{bmatrix}.$$

This problem is depicted in Figure B-1. The absolute maximum of the objective function occurs at $Q=(2,3)$, and the constrained maximum occurs at $P_3=(1/2, 3/4)$, at which both constraints are binding. The path to the solution is indicated by the segmented line $P_0P_1P_2P_3$.

An initial feasible basis is provided by taking all the slack variables to be basic. The basic matrix is then the identity matrix, $R_B=I$, and we can skip the first step of the algorithm in which the initial basis is established, since $\hat{R}=R$.

Deleting the basic columns from $\hat{R}$ yields the tableau shown in Figure B-2, which also shows the tableaux for the successive iterations of the algorithm. The basic variables identify the rows, and the nonbasic variables the columns, according to the labeling shown. The notes on the right indicate for each tableau the information obtained from that tableau in order to identify the pivot row and column. After executing the pivoting operation four times, one obtains the solution in the last column of the fifth tableau.

The procedures all conform to the flow chart of the algorithm given in Figure 3-1. Here, we will go through these procedures for the iteration from the first to the second tableau. Looking in the last column of the first tableau for the values of the basic variables, we find that both of the dual variables are negative. Since they are both of the same magnitude we arbitrarily choose v_1 to be driven to zero and designate it as the floating pivot variable. Its complementary

variable is x_1, so we designate x_1 as the parametric variable. Hence, in the notation of the text, p=1 and we find the minimum positive ratio $\hat{r}_{ip}/\hat{r}_{iL}$ where L indexes the last column and i runs over the rows of v_1 and the primal variables. These ratios are (i) for v_1, (-1)/(-2); (ii) for s_1, 2/1; (iii) for s_2, 3/3. They are all positive and the smallest one is the one for v_1. Hence, we make v_1 nonbasic to be replaced by the parametric variable x_1. Executing the pivoting operation on the tableau accomplishes this basis change, and interchanging the labels on the pivot row and column prepares us for the next iteration. Because the floating pivot variable was made nonbasic in the first iteration, the second iteration starts the procedure over again. If the floating pivot variable had not been made nonbasic, then the complement of the primal variable that was made nonbasic would become the parametric variable for the second iteration. This is illustrated in the next iteration, where u_2 becomes the new parametric variable.

Figure B-1

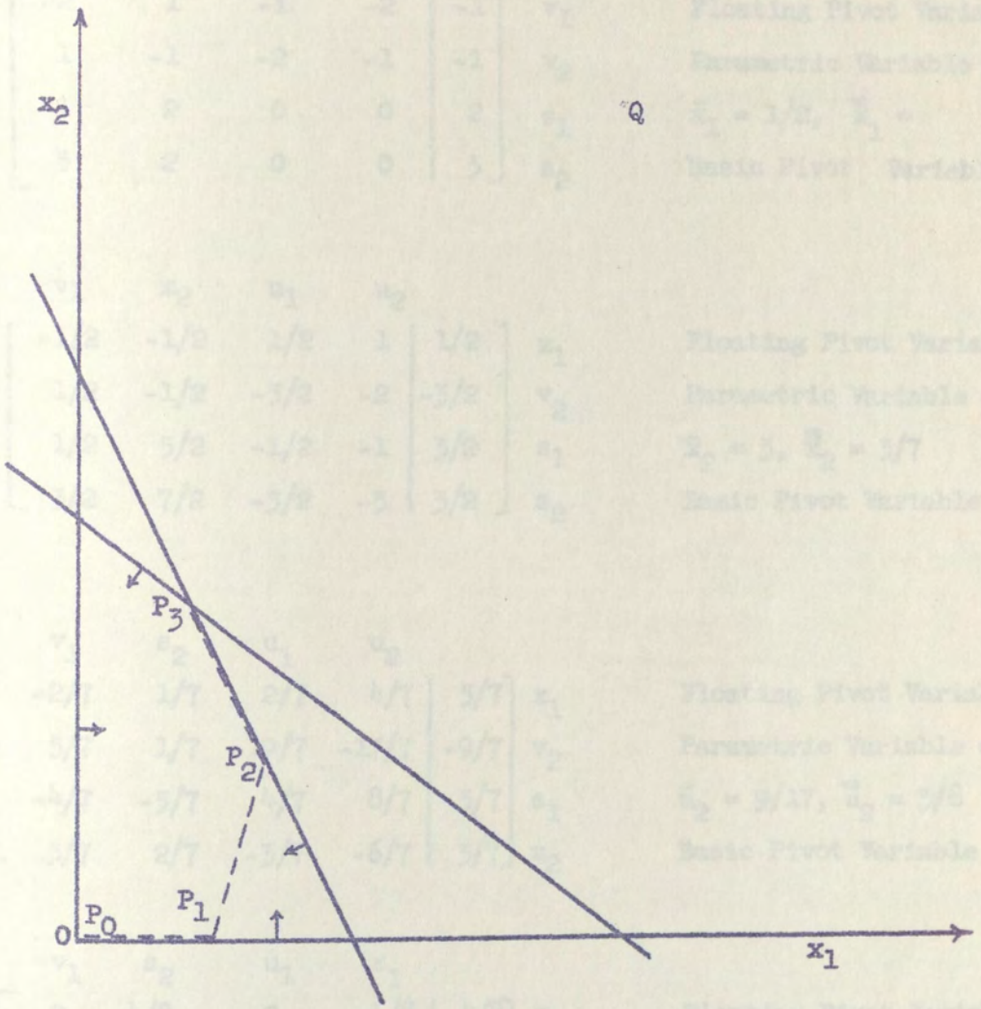


Figure B-2

The Sequence of Tableaux

Iteration Number	Tableau						Notes
1	x_1	x_2	u_1	u_2			
	-2	1	-1	-2	-1	v_1	Floating Pivot Variable = v_1
	1	-1	-2	-1	-1	v_2	Parametric Variable = x_1
	1	2	0	0	2	s_1	$\bar{x}_1 = 1/2$, $\bar{\bar{x}}_1 =$
	3	2	0	0	3	s_2	Basic Pivot Variable
2	v_1	x_2	u_1	u_2			
	-1/2	-1/2	1/2	1	1/2	x_1	Floating Pivot Variable = v_2
	1/2	-1/2	-5/2	-2	-3/2	v_2	Parametric Variable = x_2
	1/2	5/2	-1/2	-1	3/2	s_1	$\bar{x}_2 = 3$, $\bar{\bar{x}}_2 = 3/7$
	3/2	7/2	-3/2	-3	3/2	s_2	Basic Pivot Variable = s_2
3	v_1	s_2	u_1	u_2			
	-2/7	1/7	2/7	4/7	5/7	x_1	Floating Pivot Variable = v_2
	5/7	1/7	-19/7	-17/7	-9/7	v_2	Parametric Variable = u_2
	-4/7	-5/7	4/7	8/7	3/7	s_1	$\bar{u}_2 = 9/17$, $\bar{\bar{u}}_2 = 3/8$
	3/7	2/7	-3/7	-6/7	3/7	x_2	Basic Pivot Variable = s_1
4	v_1	s_2	u_1	s_1			
	0	4/8	0	-4/8	4/8	x_1	Floating Pivot Variable = v_2
	-4/8	-77/8	-12/8	17/8	-3/8	u_1	Parametric Variable = u_1
	-4/8	-5/8	4/8	7/8	3/8	u_2	$\bar{u}_1 = 3/12$, $\bar{\bar{u}}_1 =$ infinity
	0	-2/8	0	6/8	6/8	x_2	Basic Pivot Variable = v_2

Figure B-2 continued

5

v_1	s_2	v_2	s_1		
	0		6/12	x_1	
	-8/12		3/12	u_1	
	4/12		3/12	u_2	Final Tableau
	0		9/12	x_2	

Appendix C

The Computer Routines

In this appendix I describe the computer routines that implement the Simplicial Algorithm for

1. Quadratic Programming,

and the Extended Simplicial Algorithm for

2. Linear Programming Under Uncertainty, and
3. Variable-Factor Programming.

The routines are listed separately at the end of this appendix along with sample outputs, while all of the associated material (including the source and object program card decks) that can not be included here is deposited with Professor John Bishop. The routines are available for general use, and instructions for using them are included here.

1. Preparing Input Data

The input data for each problem to be solved is prepared in a data package consisting of an identification card followed by the data on punched cards. In order to use a routine, one takes the object-program binary-card deck and appends the data packages in sequence behind it. In addition, the FORTRAN Monitor System used on most machines requires that two other cards be added: (1) a card with an * (asterisk) in column 1 and XEQ in columns 7-9, inserted before the object-program deck, and (2) a card with an * in column 1 and DATA in columns 7-10, inserted between the

object-program deck and the first data package. The entire card deck is then put on tape and mounted on logical tape unit 5 (station A2) to be read by the routine. Output is written on logical tape unit 6 (station A3) for later printing.

The identification card for each data package must be prepared in a fixed format, which differs for each routine. Because the routines allow for variable format statements, however, the other input data can be prepared in nearly any form convenient to the user, provided it is ordered properly. The user must acquaint himself with format statements.*

Exhibits C-1, 2, and 3 describe the composition of a data package for each of the three routines.

2. Organization of the Routines

The computer routines are written in the FORTRAN language.* To the person versed in this language, the routines are almost self-explanatory given an understanding of the algorithm, and of the notation being used. Consequently, the minute details of the routines are not discussed here. Exhibits C-4, 6, and 8 depict the organization of the three routines schematically in terms of relations among subroutines. Exhibits C-5, 7, and 9 then describe the functions of the various subroutines in terms of the flow charts of the algorithms; in addition, the notation in each routine

* Cf. Reference Manual, 709/7090 FORTRAN Programming System, Revised Edition (1961), IBM Corporation, Poughkeepsie, New York.

is defined in relation to the notation in the text.

3. The FORTRAN Statements and Sample Output

Exhibits C-10, 12, and 14 are the FORTRAN statements for the respective routines, each organized by subroutines in the order used in Exhibits C-5, 7, and 9. I do not expect these routines to be comprehensible to every reader, but they are presented here for the person knowing the FORTRAN language, and particularly for the specialist. Also, they serve to document the work reported in the text.

Exhibits C-11, 13, and 15 are sample output sheets from the respective routines for various test problems (mostly generated from random numbers). They serve to indicate the output format, which is largely self-explanatory.

Exhibit C-1

Quadratic Programming

Data Package Composition

1. ID Card

Columns	Data
1-5	J
6-10	N
15-19	EPSILN; format E4.0: the initial basic matrix is considered singular if its determinant is less than this quantity in absolute value; e.g., 1-05 means that the minimum allowable absolute value of the determinant is 10^{-5}.
(20	JEKNSD: not used, leave blank)
(25	KRANDM: indicator for random data generation, leave blank)

2. Vector $p \equiv (p_j)$: format card, followed by data,

   $(p_j;\ j = 1, \ldots, J)$.

3. Matrix $A \equiv (a_{ij})$: format card, followed by data,

   $((a_{ij};\ i = 1, \ldots, I);\ j = 1, \ldots, J)$.

4. Constraint Set, $Dx \leq c$: format card, followed by data,

   $(c_n, (d_{nj};\ j = 1, \ldots, J);\ n = 1, \ldots, N)$.

5. Initial basis: column $j \neq 0$ only if x_j is to be a basic variable, and column $(n + J) \neq 0$ only if u_n is to be basic; e.g., all zeroes implies all slack variables are basic. (After 72 columns to a card, continue on next card.)

Exhibit C-2

Linear Programming Under Uncertainty

Data Package Composition

0. Before the first data package, there must be the following:

   1. Format card for the Normal tables:

   2. Parameter card with
      NTABLE in columns 1-5,
      DELTA in columns 6-8 (format F3.3);

   3. Normal distribution table (upper half of the distribution);

   4. Normal density table (either half).

   NTABLE is the number of entries per table, and DELTA is the tabulation increment.

1. Identification Card

Columns	Data
1-3	J
4-6	I
7-9	N
10-21	EP; cf. EPSILN, Exhibit C-1 (format E12.4; e.g., 25000-07 means that EP is 2.5×10^{-7}).
22-24	MAXIT; maximum number of approximations allowed.
25-27	ITMIN; number of major reapproximations required.
28-39	CV; accuracy desired; viz., the maximum allowable sum of squared differences between values of the basic variables in successive minor reapproximations (format E12.4; e.g., 10000-0n means that CV is 10^{-n} for $n \leq 9$).

2. The format cards, in order for data subgroups 3-8 below:

3. Vector of product prices,
   $(p_i; i = 1, \ldots, I)$.

Exhibit C-2 - page 2

4. Vector of negatives of unit activity costs,
$(-q_j; j = 1, \ldots, J)$.

5. Initial inventories, say
$(h_i; i = 1, \ldots, I)$.

6. Matrix of production coefficients,
$((a_{ij}; j = 1, \ldots, J); i = 1, \ldots, I)$.

7. Vector of constraint capacities,
$(c_n; n = 1, \ldots, N)$.

8. Matrix of the constraint set,
$((d_{nj}; j = 1, \ldots, J); n = 1, \ldots, N)$.

9. Parameters of the penalty function,
$((\alpha_i, \beta_i, \bar{d}_i, s_i); i = 1, \ldots, I)$
where $\bar{d}_i$ and s_i are the mean and standard deviation, respectively, of $\tilde{d}_i$.

10. Initial basis description: nonzero columns signify basic program variables as follows,

Column	Program Variables
j	x_j is basic
$i + J$	y_i is basic
$i + I + J$	u_i^h is basic
$n + 2I + J$	u_n^c is basic

in the notation of Section 5.

Exhibit C-3

Variable-Factor Programming

Data Package Composition

1. Identification Card

Columns	Data
1-5	KIND = 19 for the agricultural planning problem =1 for the functions described in Section 6.
6-10	J; number of activities.
11-15	N; number of fixed-factor constraints.
16-20	M; number of variable-factor constraints.
21-25	MAXIT; maximum number of approximations allowable.
26-30	MINIT; number of major reapproximations desired.
40-43	CV; accuracy requirement; viz., maximum allowable absolute difference between values of basic variables in successive minor reapproximations (format E4.0).
50-53	EP; cf. EPSILN, Exhibit C-1 (format E4.0).
(60	KRANDM: nonzero only if random input data desired).

2. Format cards in order for the data subgroups 3-5 below.

3. Fixed-factor constraints,
$(c_n, (d_{nj}; j = 1, \ldots, J); n = 1, \ldots, N)$.

4. Variable-factor constraints,
$(b_m; m = 1, \ldots, M)$.

5. Initial guesses,
$(v_m^*; m = 1, \ldots, M)$, $(x_j^*; j = 1, \ldots, J)$.

6. Initial basis: a nonzero column indicates a basic variable as follows:

Column	Variable
n	u_n
m + N	v_m
j + M + N	x_j

Exhibit C-3 continued

7. Parameters of the yield functions $y_i(z^j)$: each user must write a FORTRAN subroutine to calculate values (cf. Exhibit C-9) of the yield functions he is using, and this subroutine can, if necessary, read in parameter values.

Exhibit C-4

Quadratic Programming

Organization Chart of the Subroutines

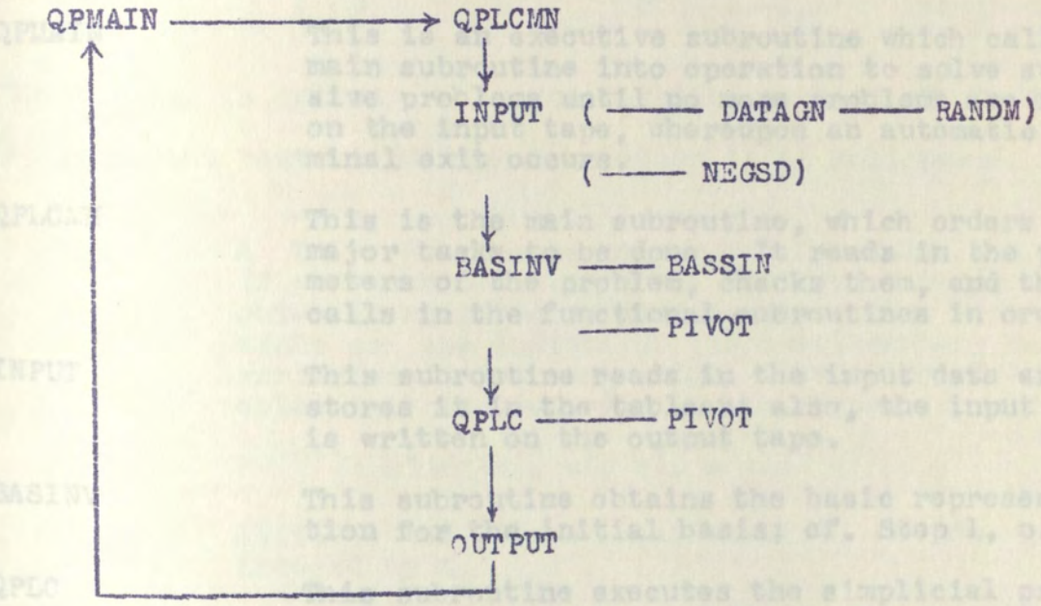


Exhibit C-5

Quadratic Programming

Description of the Subroutines

Name of the Subroutine	Description
QPMAIN	This is an executive subroutine which calls the main subroutine into operation to solve successive problems until no more problems are found on the input tape, whereupon an automatic terminal exit occurs.
QPLCMN	This is the main subroutine, which orders the major tasks to be done. It reads in the parameters of the problem, checks them, and then calls in the functional subroutines in order.
INPUT	This subroutine reads in the input data and stores it in the tableau; also, the input data is written on the output tape.
BASINV	This subroutine obtains the basic representation for the initial basis; cf. Step 1, p. 33.
QPLC	This subroutine executes the simplicial procedures on the tableau of the basic representation until the optimum is found; cf. Steps 2-5, p. 33.
PIVOT	This subroutine executes the pivoting operation on the tableau whenever called for by BASINV or QPLC; cf. p. 30.
OUTPUT	This subroutine writes the output data (solution values of the basic variables and the number of iterations required to find the optimum, on the output tape.
BASSIN	If the initial basis yields a singular basic matrix, then this subroutine terminates the computations.
NEGSD	Although not completed to date, this subroutine provides an option to check that the objective is concave.
RANDM DATAGN	These are subroutines to generate random numbers as input data, if called for by an indicator among the input parameters of the problem.

Exhibit C-5 continued

Format of the Tableau

$$W \equiv \begin{array}{c} \begin{matrix} x & u & v & s & \end{matrix} \\ \begin{bmatrix} A + A^t & -D^t & I_J & 0 & -p \\ D & 0 & 0 & I_N & c \end{bmatrix} \end{array}$$

The tableau is called W in the routine, instead of R and $\hat{R}$ as in the text. Additional notation is as follows:

IB	A $2(J + N)$-dimensional vector for which the first $(J + N)$ components are the indices of basic variables in some order, and the next $(J + N)$ components are the indices of the complementary nonbasic variables in the same order. (Indexing of the variables follows the order in the vector $z^t \equiv (x^t, u^t, v^t, s^t)$.)
JV	JV(K) is the column location in W of the variable indexed by K.
IR	IR(K) is the row in W labeled by the basic variable indexed by IB(K), $K \leq J + N$.
B	This is the matrix D of the constraint set as used in the text.

Exhibit C-6

Linear Programming Under Uncertainty

Organization Chart of the Subroutines

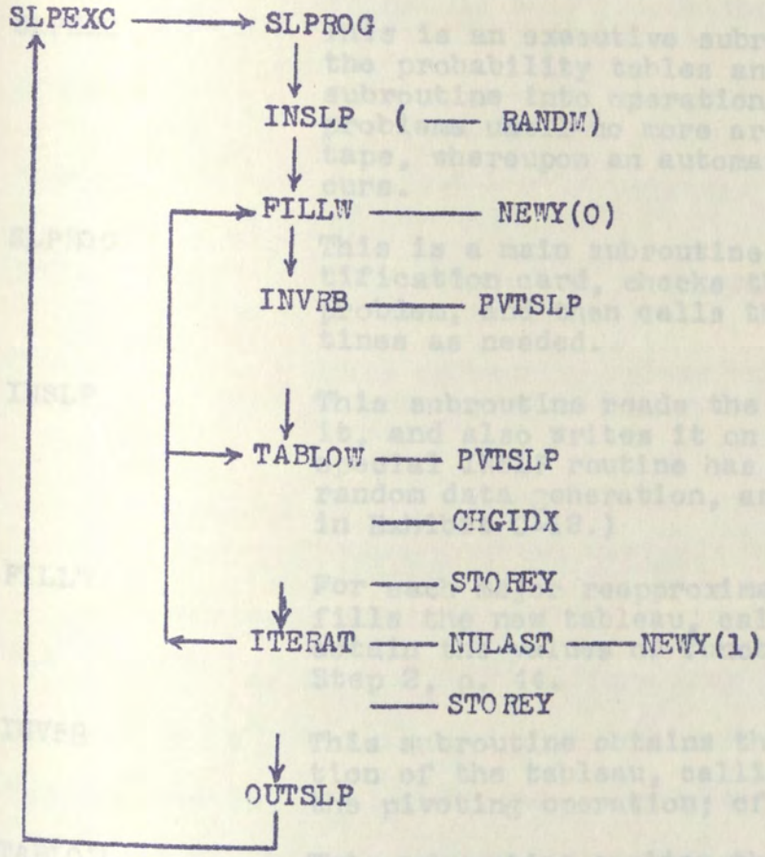


Exhibit C-7

Linear Programming Under Uncertainty

Description of the Subroutines

Name of the Subroutine	Description
SLPEXC	This is an executive subroutine which reads in the probability tables and then calls the main subroutine into operation to solve successive problems until no more are found on the input tape, whereupon an automatic terminal exit occurs.
SLPROG	This is a main subroutine which reads the identification card, checks the parameters of the problem, and then calls the functional subroutines as needed.
INSLP	This subroutine reads the input data, stores it, and also writes it on the output tape. (A special INSLP routine has been written for random data generation, and it is given last in Exhibit C-12.)
FILLW	For each major reapproximation, this subroutine fills the new tableau, calling upon NEWY to obtain the values of functions required; cf. Step 2, p. 44.
INVRB	This subroutine obtains the basic representation of the tableau, calling upon PVTSLP for the pivoting operation; cf. Step 3, p. 44.
TABLOW	This subroutine applies the Simplicial Algorithm on the basic representation of the tableau in order to iterate to an optimum for the approximating quadratic problem. It calls upon PVTSLP for the pivoting operation and CHGIDX for relabeling of the rows and columns of the tableau. At the end it calls STOREY to store the optimal y-values for use in the approximating procedure; cf. Step 4, p. 44.
ITERAT	This subroutine performs minor reapproximations by calling on NULAST and STOREY until an exit to OUTSLP, or back to TABLOW, is necessary; cf. Steps 6-8, p. 44.
NULAST	This subroutine calculates new values of the

Exhibit C-7 continued

	column of basic variables for each minor reapproximation, obtaining the values of the functions involved by calling NEWY.
NEWY	This subroutine calculates function values as required for a major (KEY= 0) or a minor (KEY= 1) reapproximation. Currently it is written to handle only proportional losses and Normal probability distributions.
STOREY	This subroutine stores the solution values of the y-variables.
CHGIDX	This subroutine relabels pivot row and column of the tableau.
PVTSLP	This subroutine performs the pivoting operation on the tableau.
OUTSLP	This subroutine writes the solution values and a summary of the computations on the output tape.

Format of the Tableau

The tableau, which is called W in the routine instead of ${}^{*}R^{k}$ and ${}^{*}\hat{R}^{k}$ as in the text, differs from (5-5) in that the columns of s^{h} and u^{h}, and also s^{c} and u^{c}, are interchanged. The vectors IB, JV, and IR and the matrix B are defined in Exhibit C-5. In this routine, the vector p of the text is called PI, and the vector -q of the text is called P; also, GAMMA(I) is the sum of α_i and β_i.

Exhibit C-8

Variable-Factor Programming

Organization Chart of the Subroutines

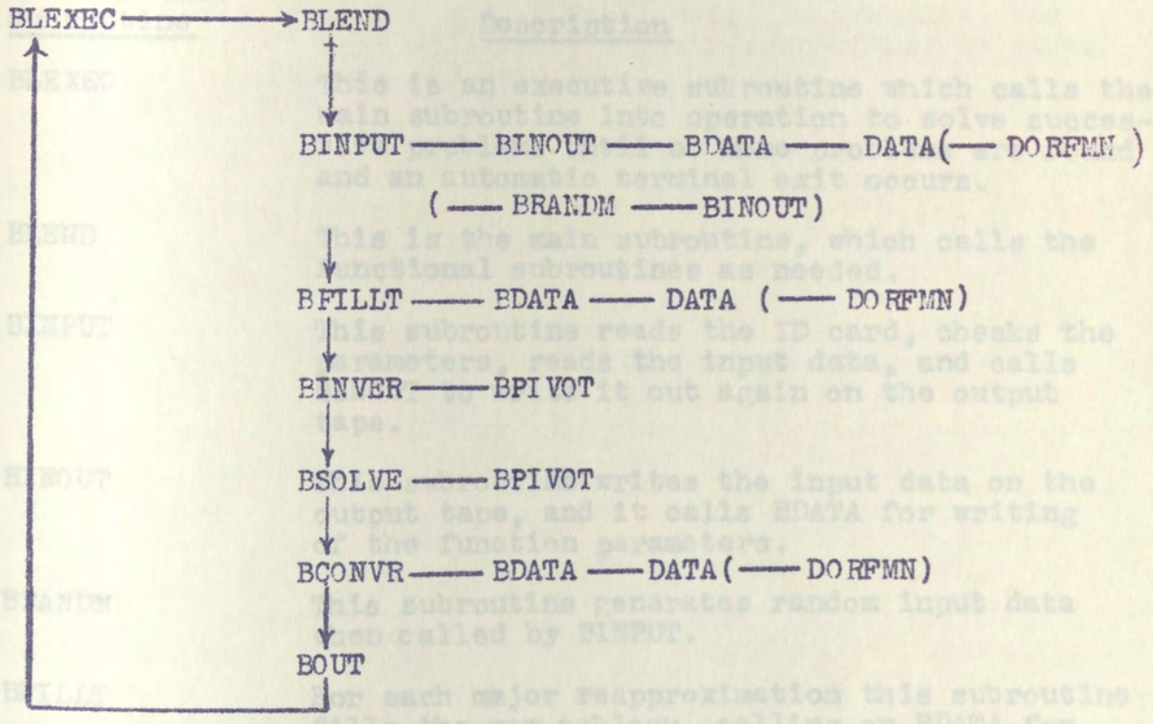


Exhibit C-9

Variable-Factor Programming

Description of the Subroutines

Name of the Subroutine	Description
BLEXEC	This is an executive subroutine which calls the main subroutine into operation to solve successive problems until no more problems are found and an automatic terminal exit occurs.
BLEND	This is the main subroutine, which calls the functional subroutines as needed.
BINPUT	This subroutine reads the ID card, checks the parameters, reads the input data, and calls BINOUT to write it out again on the output tape.
BINOUT	This subroutine writes the input data on the output tape, and it calls BDATA for writing of the function parameters.
BRANDM	This subroutine generates random input data when called by BINPUT.
BFILLT	For each major reapproximation this subroutine fills the new tableau, calling on BDATA for function values; cf., Step 2, p. 44.
BSOLVE	This subroutine performs the Simplicial Algorithm on the basic representation of the tableau to find an optimum for the approximating quadratic problem; cf., Step 4, p. 44.
BPIVOT	This subroutine performs the pivoting operation, and it also interchanges the labeling of the pivot row and column when called by BSOLVE; cf., pp. 30-31.
BCONVR	This subroutine performs minor reapproximations until either convergence is obtained, or a basic variable becomes negative; cf., Steps 6-8, p. 44.
BOUT	This subroutine writes out the solution values and a summary of the computations.
BDATA DATA	These subroutines read in the yield-function parameters, and calculate values needed by the

Exhibit C-9 continued

approximation procedure. They call on DORFMN to obtain values for the agricultural planning problem mentioned in Section 6, and SHLAFR to obtain values for the other functions listed at the end of Section 6. In practice, the user will write similar subroutines to calculate $\nabla\bar{y}$ and $\nabla^2\bar{y}$.

Exhibit C-10

Quadratic Programming

FORTRAN Statements

```
*     LIST8
*     LIBE
*     LABEL
CQPMAIN
C                    QUADRATIC PROGRAMMING WITH LINEAR CONSTRAINTS
C                    EXECUTIVE ROUTINE
      NAME=0
    1 NAME=NAME+1
      CALL QPLCMN(NAME)
      GO TO 1
      END
```

```
*    LIST8
*    LIBE
*    LABEL
CINPUT
     SUBROUTINE INPUT (NUMBER,KRANDM)
     COMMON I,J,K,L,M,N,IP,JP,KP,LP,MP,NP,MM,LAST,DUMA,
    1ITERNO,ITERNN,MAXIT,IREPET,JEKNSD,IFBFEZ,DUMB,
    2ALPHA,BETA,GAMMA,DELTA,EPSILN,CONVER,DUMC,
    3IPRGRM,DUMD,FMTA,FMTB,FMTC,FMTP,FMTU,
    4IB,JV,IR,W,IBETA,
    5DUMMY
C
     DIMENSION DUMA(3), DUMB(2), DUMC(18), DUMD(3),
    1FMTA(12), FMTB(12), FMTC(12), FMTP(12), FMTU(12),
    2IB(208), JV(208), IR(104), W(104,208), IBETA(104),
    3DUMMY(673)

     WRITE OUTPUT TAPE 6,1,NUMBER
   1 FORMAT (1H1/////42X33HQUADRATIC PROGRAMMING PROBLEM NO. I3///
    1   55X10HINPUT DATA//)
     IF(KRANDM)20,20,21
  20 READ INPUT TAPE 5,FMTP,(W(MP,LAST),MP=1,J)
     WRITE OUTPUT TAPE 6,2,J,(W(MP,LAST),MP=1,J)
   2 FORMAT (//42X33HACTIVITY PROFIT RATES, P(J), J=1, I3//(6X5F20.6))
     DO 3 MP=1,J
   3 W(MP,LAST)=-W(MP,LAST)
     READ INPUT TAPE 5, FMTA, ((W(MP,NP), MP=1,J), NP=1,J)
     WRITE OUTPUT TAPE 6,10,J
  10 FORMAT (///37X42HMATRIX OF THE QUADRATIC FORM, A(I,J), J=1,I3/
    1   /2X1HJ/)
     DO 11 JP=1,J
  11 WRITE OUTPUT TAPE 6,12,JP, (W(K,JP),K=1,J)
  12 FORMAT (/1XI2,3X5F20.6/(6X5F20.6))
     DO 121   NP=2,J
     MPP=NP-1
     DO 122 MP=1,MPP
     W(MP,NP)=W(MP,NP)+W(NP,MP)
     W(NP,MP) = W(MP,NP)
 122 CONTINUE
 121 CONTINUE
     IF (JEKNSD-1) 124,125,124
 125 CALL NEGSD
 124 JPP=J+1
     READ INPUT TAPE 5,FMTC,(W(MP,LAST),(W(NP,MP),NP=1,J),MP=JPP,M)
     DO 115 MP=1,J
     DO 114 NP=JPP,M
     W(NP,MP)=W(MP,NP)
 114 W(MP,NP)=-W(MP,NP)
 115 CONTINUE
  55 FORMAT (72I1)
     READ INPUT TAPE 5,55,(IB(K),K=1,M)
     IB(LAST) = LAST
     DO 50 K=1,M
     KM=K+M
     IF(IB(K))51,52,51
  51 IB(K)=K
     IB(KM)=KM
```

```
      GO TO 50
   52 IB(K)=KM
      IB(KM)=K
   50 CONTINUE
      WRITE OUTPUT TAPE 6,13,J,N
   13 FORMAT (///36X34HCONSTRAINT SET, C(N), B(N,J), J=1, I3,8H,    N=1,
     1 I3/2X1HN,54X6HB(N,J), 51X4HC(N))
      DO 14 NP=1,N
      KP=J+NP
   14 WRITE OUTPUT TAPE 6,17,NP,W(KP,LAST),(W(KP,L),L=1,J)
   17 FORMAT (/1XI2,3X94XF20.6/(6X5F20.6))
      WRITE OUTPUT TAPE 6,15,M,(IB(K),K=1,M)
   15 FORMAT (///49X19HINITIAL BASIS, K=1, I3//(35X10I5))
      RETURN
   21 CALL DATAGN
      RETURN
      END
```

```
*    LIST8
*    LIBE
*    LABEL
CBASINV
C                BASIS INVERSION SUBROUTINE
     SUBROUTINE BASINV
C                          COMMON AND DIMENSION STATEMENTS
     COMMON I,J,K,L,M,N,IP,JP,KP,LP,MP,NP,MM,LAST,DUMA,
    1ITERNO,ITERNN,MAXIT,IREPET,JEKNSD,IFBFEZ,DUMB,
    2ALPHA,BETA,GAMMA,DELTA,EPSILN,CONVER,DUMC,
    3IPRGRM,DUMD,FMTA,FMTB,FMTC,FMTP,FMTU,
    4IB,JV,IR,W,IBETA,
    5DUMMY
C
     DIMENSION DUMA(3), DUMB(2), DUMC(18), DUMD(3),
    1FMTA(12), FMTB(12), FMTC(12), FMTP(12), FMTU(12),
    2IB(208), JV(208), IR(104), W(104,208), IBETA(104),
    3DUMMY(673)
   1 DO 2 K = 1, M
   2 IBETA (K) = 0
C
     DO 8  MP = 1, M
     LP = IB(MP)
     L = JV(LP)
     ALPHA = 0.0
C
     DO 3  K = 1, M
     IF (IBETA(K)) 4, 4, 3
   4 DELTA = ABSF(W(K,L))
     IF (DELTA-ALPHA) 3,3,5
   5 ALPHA = DELTA
     IP = K
   3 CONTINUE
C
     IF (ALPHA-EPSILN) 6,6,7
   6 CALL BASSIN
     RETURN
   7 IR(MP) = IP
     IBETA (IP) = 1
     CALL PIVOT (IP, MP, MP, LAST)
   8 CONTINUE
     DO 9 K=1,M
     I=IR(K)
     IF(W(I,LAST))10,9,9
  10 I=IB(K)
     IF(I-J)11,11,12
  12 IF(I-M-J)9,9,11
   9 CONTINUE
     RETURN
  11 WRITE OUTPUT TAPE 6,13
  13 FORMAT (//////45X29HINITIAL BASIS IS NOT FEASIBLE)
     CALL EXIT
     END
```

```
18 FORMAT (///39X34HSOLUTION UNBOUNDED, DUAL VARIABLE I3,8H IS NEG.)
   RETURN

17 IF(DA)7,19,20
19 W(K,LAST)=0.0000002
   GO TO 21

20 IF(PA-DA)24,25,25
24 IF(PA)25,25,23

23 CALL PIVOT(KK,IPC,M+1,LAST)
   I=IB(IPR)
   JJ=IB(IPC)
   K=JV(I)
   L=JV(JJ)
   IB(IPR)=JJ
   IB(IPC)=I
   JV(I)=L
   JV(JJ)=K
   GO TO 21

25 CALL PIVOT(K,IPC,M+1,LAST)
   I=IB(KPR)
   JJ=IB(IPC)
   K=JV(I)
   L=JV(JJ)
   IB(KPR)=JJ
   IB(IPC)=I
   JV(I)=L
   JV(JJ)=K
   IPC=KPR+M
   GO TO 4

 7 WRITE OUTPUT TAPE 6,9
 9 FORMAT (////43X33HPRIMAL INFEASIBILITY, NO SOLUTION)
   RETURN
   END
```

```
*   LIST8
*   LIBE
*   LABEL
COUTPUT
      SUBROUTINE OUTPUT
C                          COMMON AND DIMENSION STATEMENTS
      COMMON I,J,K,L,M,N,IP,JP,KP,LP,MP,NP,MM,LAST,DUMA,
     1ITERNO,ITERNN,MAXIT,IREPET,JEKNSD,IFBFEZ,DUMB,
     2ALPHA,BETA,GAMMA,DELTA,EPSILN,CONVER,DUMC,
     3IPRGRM,DUMD,FMTA,FMTB,FMTC,FMTP,FMTU,
     4IB,JV,IR,W,IBETA,
     5DUMMY
C
      DIMENSION DUMA(3), DUMB(2), DUMC(18), DUMD(3),
     1FMTA(12), FMTB(12), FMTC(12), FMTP(12), FMTU(12),
     2IB(208), JV(208), IR(104), W(104,208), IBETA(104),
     3DUMMY(673)
C
      WRITE OUTPUT TAPE 6,1,ITERNO
    1 FORMAT (1H1/////57X6HOUTPUT//53XI3,11H ITERATIONS////
     1  43X34HSOLUTION VALUES OF BASIC VARIABLES //
     1  51X18HECONOMIC VARIABLES//)
      DO 2 MP=1,MM
      IF(MP-M-1)16,17,16
   17 WRITE OUTPUT TAPE 6,18
   18 FORMAT (///52X15HSLACK VARIABLES//)
   16 DO 3 K=1,M
      JP=IB(K)
      IF(JP-MP)3,4,3
    3 CONTINUE
      GO TO 2
    4 I=IR(K)
      IF(JP-J)5,5,7
    5 WRITE OUTPUT TAPE 6,6,JP,W(I,LAST)
    6 FORMAT (39X17HACTIVITY LEVEL X(I2,2H)=F20.6)
      GO TO 2
    7 IF(JP-M)8,8,10
    8 JP=JP-J
      WRITE OUTPUT TAPE 6,9,JP,W(I,LAST)
    9 FORMAT (26X30HMARGINAL FACILITY VALUATION U(I2,2H)=F20.6)
      GO TO 2
   10 IF(JP-M-J)11,11,13
   11 JP=JP-M
      WRITE OUTPUT TAPE 6,12,JP,W(I,LAST)
   12 FORMAT (43X13HDUAL SLACK V(I2,2H)=F20.6)
      GO TO 2
   13 JP=JP-M-J
      WRITE OUTPUT TAPE 6,14,JP,W(I,LAST)
   14 FORMAT (41X15HPRIMAL SLACK S(I2,2H)=F20.6)
    2 CONTINUE
      WRITE OUTPUT TAPE 6,15
   15 FORMAT (1H1/////)
      RETURN
      END
```

```
   LIST8
   LIBE
   LABEL
BASSIN
   SUBROUTINE BASSIN
   COMMON I,J,K,L,M,N,IP,JP,KP,LP,MP,NP,MM,LAST,DUMA,
  1ITERNO,ITERNN,MAXIT,IREPET,JEKNSD,IFBFEZ,DUMB,
  2ALPHA,BETA,GAMMA,DELTA,EPSILN,CONVER,DUMC,
  3IPRGRM,DUMD,FMTA,FMTB,FMTC,FMTP,FMTU,
  4IB,JV,IR,W,IBETA,
  5DUMMY

   DIMENSION DUMA(3), DUMB(2), DUMC(18), DUMD(3),
  1FMTA(12), FMTB(12), FMTC(12), FMTP(12), FMTU(12),
  2IB(208), JV(208), IR(104), W(104,208), IBETA(104),
  3DUMMY(673)
   WRITE OUTPUT TAPE 6,1,ALPHA,EPSILN,MP
 1 FORMAT (/////13X47HINITIAL BASIS IS SINGULAR***NO SOLUTION, PIVOT=E12.
  1E12.8,9H,    MIN.=E12.8,11H,    RANK+1=I3)
   RETURN
   END
```

```
*     LIBE
*     LABEL
CNEGSD
      SUBROUTINE NEGSD
      RETURN
      END
```

```
*    RANDOM
*     FAP
*    RANDOM
      ENTRY   RANDM
RANDM CLA     U
      ALS     18
      ADD     U
      ADD     U
      ADD     U
      STO     U
      ARS 8
      ORA     N
      FAD     N
      STO*    1,4
      TRA     2,4
U     OCT     212345675230
N     OCT     200000000000
      END
*    LIST8
*    LIBE
*    LABEL
CDATAGN
      SUBROUTINE DATAGN
      COMMON I,J,K,L,M,N,IP,JP,KP,LP,MP,NP,MM,LAST,DUMA,
     1ITERNO,ITERNN,MAXIT,IREPET,JEKNSD,IFBFEZ,DUMB,
     2ALPHA,BETA,GAMMA,DELTA,EPSILN,CONVER,DUMC,
     3IPRGRM,DUMD,FMTA,FMTB,FMTC,FMTP,FMTU,
     4IB,JV,IR,W,IBETA,
     5DUMMY
C
      DIMENSION DUMA(3), DUMB(2), DUMC(18), DUMD(3),
     1FMTA(12), FMTB(12), FMTC(12), FMTP(12), FMTU(12),
     2IB(208), JV(208), IR(104), W(104,208), IBETA(104),
     3DUMMY(673)

      DO 25 JP=1,J
      CALL RANDM(XX)
   25 W(JP,JP)=-ABSF(XX)
      DO 30 JP=1,J,5
   30 W(JP,JP)=0.0
      DO 26 JP=1,J
      CALL RANDM(XX)
   26 W(JP,LAST)=ABSF(XX)
      KP=J+1
      DO 27 NP=KP,M
      DO 28 JP=1,J
      CALL RANDM(XX)
   28 W(JP,NP)=XX
      CALL RANDM(XX)
   27 W(NP,LAST)=ABSF(XX)
      DO 80 MP=1,M
   80 IB(MP)=0
      WRITE OUTPUT TAPE 6,2,J,(W(MP,LAST),MP=1,J)
    2 FORMAT (//42X33HACTIVITY PROFIT RATES, P(J), J=1, I3//(6X5F20.6))
      DO 3 MP=1,J
    3 W(MP,LAST)=-W(MP,LAST)
```

```
      WRITE OUTPUT TAPE 6,10,J
   10 FORMAT (///37X42HMATRIX OF THE QUADRATIC FORM, A(I,J), J=1,I3/
     1   /2X1HJ/)
      DO 11 JP=1,J
   11 WRITE OUTPUT TAPE 6,12,JP, (W(K,JP),K=1,J)
   12 FORMAT (/1XI2,3X5F20.6/(6X5F20.6))
      DO 121   NP=2,J
      MPP=NP-1
      DO 122 MP=1,MPP
      W(MP,NP)=W(MP,NP)+W(NP,MP)
      W(NP,MP) = W(MP,NP)
  122 CONTINUE
  121 CONTINUE
      IF (JEKNSD-1) 124,125,124
  125 CALL NEGSD
  124 JPP=J+1
      DO 115 MP=1,J
      DO 114 NP=JPP,M
      W(NP,MP)=W(MP,NP)
  114 W(MP,NP)=-W(MP,NP)
  115 CONTINUE
      IB(LAST) = LAST
      DO 50 K=1,M
      KM=K+M
      IF(IB(K))51,52,51
   51 IB(K)=K
      IB(KM)=KM
      GO TO 50
   52 IB(K)=KM
      IB(KM)=K
   50 CONTINUE
      WRITE OUTPUT TAPE 6,13,J,N
   13 FORMAT (///36X34HCONSTRAINT SET, C(N), B(N,J), J=1, I3,8H,   N=1,
     1   I3/2X1HN,54X6HB(N,J), 51X4HC(N))
      DO 14 NP=1,N
      KP=J+NP
   14 WRITE OUTPUT TAPE 6,17,NP,W(KP,LAST),(W(KP,L),L=1,J)
   17 FORMAT (/1XI2,3X94XF20.6/(6X5F20.6))
      WRITE OUTPUT TAPE 6,15,M,(IB(K),K=1,M)
   15 FORMAT (///49X19HINITIAL BASIS, K=1, I3//(35X10I5))
      RETURN
      END
```

Exhibit C-11

Quadratic Programming

Sample Output

QUADRATIC PROGRAMMING PROBLEM NO. 2

INPUT DATA

ACTIVITY PROFIT RATES, P(J), J=1, 5

1.541143	1.062127	0.718809	1.066560	1.460803

MATRIX OF THE QUADRATIC FORM, A(I,J), J=1, 5

J					
1	0.	0.	0.	0.	0.
2	0.	-0.850432	0.	0.	0.
3	0.	0.	-1.038530	0.	0.
4	0.	0.	0.	-1.807811	0.
5	0.	0.	0.	0.	-0.086739

CONSTRAINT SET, C(N), B(N,J), J=1, 5, N=1, 3

N	B(N,J)				C(N)
1	0.582890	3.694908	0.296351	3.716483	0.961645	4.321519
2	0.318578	0.188048	0.130545	1.874105	0.894820	2.007926
3	0.479253	2.466393	3.090534	6.148127	0.134245	7.110610

INITIAL BASIS, K=1, 8

9 10 11 12 13 14 15 16

OUTPUT

4 ITERATIONS

SOLUTION VALUES OF BASIC VARIABLES

ECONOMIC VARIABLES

ACTIVITY LEVEL X(1)=	6.162542
ACTIVITY LEVEL X(2)=	0.179238
ACTIVITY LEVEL X(3)=	0.084050
MARGINAL FACILITY VALUATION U(2)=	4.837575

SLACK VARIABLES

DUAL SLACK V(4)=	7.999561
DUAL SLACK V(5)=	2.867958
PRIMAL SLACK S(1)=	0.042256
PRIMAL SLACK S(3)=	3.455365

QUADRATIC PROGRAMMING PROBLEM NO. 7

INPUT DATA

ACTIVITY PROFIT RATES, P(J), J=1, 50

3.935859	5.605611	3.105469	1.407563	0.992316
0.577219	0.346961	3.547173	0.792429	0.415007
0.462057	3.619241	0.540693	1.915146	0.439602
0.328533	0.014782	1.055153	5.266595	1.525798
5.219790	0.048753	0.420253	0.082743	2.856711
4.323065	6.517596	0.197990	6.529578	5.395559
5.607148	0.270715	4.032815	0.806396	1.205836
2.204757	0.594005	0.603327	0.273916	0.854220
3.264351	3.898126	2.009594	0.974429	0.940058
0.895766	1.814198	0.329700	1.874457	5.248551

MATRIX OF THE QUADRATIC FORM, A(I,J), J=1, 50

J					
1	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
2	0.	-3.832345	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
3	0.	0.	-3.524405	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.

	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
4	0.	0.	0.	-0.041055	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
5	0.	0.	0.	0.	-0.316416
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
6	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
7	0.	0.	0.	0.	0.
	0.	-0.163145	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
8	0.	0.	0.	0.	0.
	0.	0.	-0.098346	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
9	0.	0.	0.	0.	0.

	0.	0.	0.	-0.243534	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
10	0.	0.	0.	0.	0.
	0.	0.	0.	0.	-0.845487
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
11	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
12	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	-2.021941	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
13	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	-5.786099	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
14	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	-0.129781	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.		

	0.	0.	0.	0.
	0.	0.	0.	0.
	0.	0.	0.	0.
	0.	0.	0.	0.
5	0.	0.	0.	0.
	0.	0.	0.	0.
	0.	0.	0.	-3.519929
	0.	0.	0.	0.
	0.	0.	0.	0.
	0.	0.	0.	0.
	0.	0.	0.	0.
	0.	0.	0.	0.
	0.	0.	0.	0.
	0.	0.	0.	0.
6	0.	0.	0.	0.
	0.	0.	0.	0.
	0.	0.	0.	0.
	0.	0.	0.	0.
	0.	0.	0.	0.
	0.	0.	0.	0.
	0.	0.	0.	0.
	0.	0.	0.	0.
	0.	0.	0.	0.
7	0.	0.	0.	0.
	0.	0.	0.	0.
	0.	0.	0.	0.
	0.	-1.021706	0.	0.
	0.	0.	0.	0.
	0.	0.	0.	0.
	0.	0.	0.	0.
	0.	0.	0.	0.
	0.	0.	0.	0.
8	0.	0.	0.	0.
	0.	0.	0.	0.
	0.	0.	0.	0.
	0.	0.	-2.157273	0.
	0.	0.	0.	0.
	0.	0.	0.	0.
	0.	0.	0.	0.
	0.	0.	0.	0.
	0.	0.	0.	0.
9	0.	0.	0.	0.
	0.	0.	0.	0.
	0.	0.	0.	0.
	0.	0.	0.	-1.276467
	0.	0.	0.	0.
	0.	0.	0.	0.
	0.	0.	0.	0.
	0.	0.	0.	0.
	0.	0.	0.	0.

	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
5	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	-3.519929
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
6	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
7	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	-1.021706	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
8	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	-2.157273	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
9	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	-1.276467	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.

	0.	0.	0.	0. 0.
	0.	0.	0.	0. 0.
	0.	0.	0.	0. 0.
	0.	0.	0.	0. -3.902148
	0.	0.	0.	0. 0.
	0.	0.	0.	0. 0.
	0.	0.	0.	0. 0.
	0.	0.	0.	0. 0.
	0.	0.	0.	0. 0.
	0.	0.	0.	0. 0.
21	0.	0.	0.	0. 0.
	0.	0.	0.	0. 0.
	0.	0.	0.	0. 0.
	0.	0.	0.	0. 0.
	0.	0.	0.	0. 0.
	0.	0.	0.	0. 0.
	0.	0.	0.	0. 0.
	0.	0.	0.	0. 0.
	0.	0.	0.	0. 0.
	0.	0.	0.	0. 0.
22	0.	0.	0.	0. 0.
	0.	0.	0.	0. 0.
	0.	0.	0.	0. 0.
	0.	0.	0.	0. 0.
	0.	-3.749791	0.	0. 0.
	0.	0.	0.	0. 0.
	0.	0.	0.	0. 0.
	0.	0.	0.	0. 0.
	0.	0.	0.	0. 0.
	0.	0.	0.	0. 0.
23	0.	0.	0.	0. 0.
	0.	0.	0.	0. 0.
	0.	0.	0.	0. 0.
	0.	0.	0.	0. 0.
	0.	0.	-0.267280	0. 0.
	0.	0.	0.	0. 0.
	0.	0.	0.	0. 0.
	0.	0.	0.	0. 0.
	0.	0.	0.	0. 0.
	0.	0.	0.	0. 0.
24	0.	0.	0.	0. 0.
	0.	0.	0.	0. 0.
	0.	0.	0.	0. 0.
	0.	0.	0.	0. 0.
	0.	0.	0.	-0.729621 0.
	0.	0.	0.	0. 0.
	0.	0.	0.	0. 0.
	0.	0.	0.	0. 0.
	0.	0.	0.	0. 0.
	0.	0.	0.	0. 0.
25	0.	0.	0.	0. 0.
	0.	0.	0.	0. 0.
	0.	0.	0.	0. 0.
	0.	0.	0.	0. 0.
	0.	0.	0.	0. -1.972204

0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	-1.849961	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	-0.850037	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	-3.101603	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	-0.004135
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.

31	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
32	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	-0.184182	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
33	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	-3.203625	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
34	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	-1.182380	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
35	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	-0.678630
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
36	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.

0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	-0.341936	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	-0.754418	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	-0.185834	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	-5.760971
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.

0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	-1.633047	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	-0.615540	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	-0.075539	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	-2.293786
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.

	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	-3.706236	0.	0.	0.
48	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	-2.745845	0.	0.
49	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	-0.889868	0.
50	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	-0.250133

CONSTRAINT SET, C(N), B(N,J), J=1, 50, N=1, 20

N	B(N,J)				C(N)
1					0.702592
	2.005426	0.207037	0.217807	0.721753	0.700772
	0.606539	1.943054	0.911453	2.899902	0.146774
	0.711729	0.407111	3.365095	0.767293	5.843315
	7.437354	0.017145	5.269560	3.654370	4.426405
	0.389882	4.840940	1.506941	0.074766	5.338877
	1.325058	7.850798	1.813438	1.216329	0.488518
	3.661010	0.698196	3.903805	2.855306	3.995180
	4.575568	1.874196	0.950149	3.332519	0.447433
	2.372860	1.091685	3.744475	0.704131	1.599434
	3.689001	2.277184	2.125204	3.761902	3.444582
2					0.017113
	1.860978	0.468137	0.310809	1.516483	3.008658

1.377627	1.453596	7.848592	1.231705	6.923586
2.600070	0.111071	0.816267	1.795932	0.331147
1.333347	5.019756	0.529603	0.941583	0.532572
0.493866	1.376891	0.954139	0.626828	0.347442
0.400870	3.366927	0.885122	0.079782	1.982556
1.023198	0.148094	1.136681	1.488717	3.351087
2.814591	1.141995	0.519139	0.836883	0.349043
1.124623	0.116241	0.167044	1.912188	0.466347
1.588389	5.332820	6.814928	0.447093	0.007690
3				1.249836
0.272279	0.254403	3.655158	1.544902	7.476564
3.477634	0.805315	0.057732	6.363688	5.662535
0.087753	0.156165	0.588852	3.822360	0.541852
0.950958	3.316314	0.415847	0.066752	0.915268
3.563361	1.107625	0.314094	3.601011	2.298670
2.765834	7.218951	0.605299	2.778311	0.887238
0.821031	3.764182	0.513979	0.536264	0.061132
1.080826	5.538337	1.160135	4.076597	3.577141
0.193369	1.111647	3.718243	0.786271	2.138405
0.131171	1.951223	1.346270	0.258304	0.983216
4				0.528022
0.325033	1.303742	0.242558	0.248499	1.744916
0.288792	1.880627	0.684636	0.795543	0.465657
1.896656	1.308149	6.627885	0.055158	1.418204
0.012807	2.626006	7.051017	1.519000	1.384424
2.635542	0.838358	0.200361	0.223439	3.468256
0.196910	1.377885	0.247561	0.569760	0.481235
1.293955	1.219634	1.495008	7.973366	1.009945
0.544881	3.987642	0.155062	0.492992	0.260171
0.992829	0.806161	1.206196	0.726285	1.501945
0.618776	1.333121	0.430696	1.169303	0.631648
5				5.953381
3.866446	0.295266	1.072092	3.550321	0.501066
0.036351	0.599455	0.866309	0.815318	0.649171
0.226093	3.028034	4.197075	2.677835	0.286666
0.707434	5.316853	0.620735	5.943593	1.742164
4.319604	1.599859	1.080467	0.641720	0.988217
0.615289	0.028979	0.789475	1.904149	6.007598
1.770902	1.139257	1.702029	0.852771	5.829997
4.280214	5.211308	0.372962	2.786884	2.729302
3.105946	1.176909	1.084696	3.832001	0.866867
2.316802	5.387201	1.655193	7.239815	0.981494
6				0.090857
2.526355	0.183957	6.940575	3.021006	0.915214
0.347011	7.708825	1.317034	0.278675	1.490789
3.857181	0.617754	3.138636	0.012981	0.127653
3.915631	1.459674	0.568852	3.378317	7.582466
0.171270	0.983536	1.044023	3.724756	0.412609
0.094950	3.424892	0.282788	7.925764	1.596928
0.957816	0.280359	2.975879	2.834698	0.904184
0.094063	0.289070	0.887856	1.451005	0.724629
1.644362	0.302671	0.416399	1.548703	1.797033
3.375494	0.889942	6.979175	0.229640	4.210160
7				1.051425
1.617421	0.069106	1.857845	1.050238	0.430109

7.418330	2.513024	3.391315	7.113460	0.519865
0.233098	0.041016	0.074104	1.040224	0.893396
0.039742	0.157326	0.586282	0.203514	1.334004
0.544104	2.092862	5.381673	0.681786	0.036331
0.081916	0.658077	0.249868	0.074124	1.724758
0.029972	0.283549	5.070872	0.251209	1.605050
4.434126	0.822947	2.515273	3.388380	3.385640
1.830749	2.733609	6.896345	0.086553	0.371441
0.033900	3.441746	0.857517	3.209365	1.378750

8

1.402096

1.899804	1.936000	1.035523	0.841286	0.717791
2.940670	1.901773	0.088813	1.949792	0.200247
4.210453	1.414569	0.506947	2.704490	3.953686
7.041296	6.664599	0.076991	3.857089	0.185429
0.877837	1.730368	2.780063	0.276766	0.810911
0.941838	0.001928	1.070058	0.192817	1.366566
0.314498	0.812216	0.042812	3.787696	0.592479
1.020479	3.458256	3.565230	1.133535	0.378838
1.376956	0.623178	0.189991	1.062690	1.912608
0.347228	0.217475	2.094354	0.368506	0.786442

9

0.798945

1.334600	2.777467	0.642002	2.854808	3.350831
2.411717	0.078204	0.085727	0.553342	0.934280
2.502403	2.760693	3.520911	2.139612	1.987150
5.409875	0.821863	0.845067	5.389081	3.489693
4.436424	1.605655	0.835012	7.138486	0.596313
1.094156	3.662625	0.570234	0.360996	0.950879
0.323176	7.414827	6.671788	5.297286	3.737623
6.750163	6.862365	0.552841	1.193770	4.845930
0.524966	0.246452	3.016083	0.153027	4.200537
0.773161	1.826727	4.173839	1.820214	1.530145

10

0.033345

0.511184	0.943709	1.843733	0.136110	0.742458
0.459526	5.571625	0.221703	3.176245	0.384520
0.733844	3.769535	0.397664	3.267181	0.453400
0.738480	1.078738	0.485894	5.614280	3.350650
2.839639	0.720496	7.470316	0.473083	0.305519
3.074700	1.724767	2.049805	0.775902	0.349381
0.226335	2.138315	0.755856	0.322576	0.939114
2.021964	0.919938	3.880838	1.083633	4.152090
0.475441	1.510446	3.134808	2.216402	6.383557
3.203052	1.246152	1.063179	2.327407	2.827223

11

7.846649

0.663757	1.890059	0.696364	1.345831	3.080875
2.260296	0.458474	5.321438	3.459238	1.617914
0.430151	3.043006	0.693150	2.497122	2.537950
0.997663	1.102354	0.674566	4.536823	0.331520
3.540766	1.309891	1.996224	0.041417	0.530986
0.880833	3.727246	4.871962	1.070668	0.125046
0.341270	3.688838	1.847307	1.884297	2.680020
6.242899	1.304260	1.779036	0.467939	0.801972
2.401529	0.384543	3.615284	1.924070	2.551289
1.348931	1.085194	0.110244	0.219779	1.305893

12

2.163731

1.893456	1.705773	1.193536	0.904630	0.227478

	0.199547	2.741825	1.267254	5.854198 0.539326
	2.599931	0.367715	0.851882	1.135730 0.018329
	0.832277	3.314804	0.963432	0.431988 0.256481
	5.207966	0.597648	5.815424	1.965465 0.354043
	2.237325	0.983215	0.865310	1.371692 0.134876
	0.532201	0.765440	0.901418	7.712213 0.671399
	0.704315	0.140702	1.010744	0.882959 0.047158
	0.389689	1.913712	0.987535	2.508028 7.945653
	0.550834	7.099134	0.679787	4.186517 0.750255
13				0.676397
	6.955589	2.786384	0.529500	3.630554 2.721312
	0.956611	0.678190	0.919283	1.308280 0.788065
	0.841131	7.633615	0.310065	0.684758 1.317962
	1.744947	0.304013	7.774511	0.379082 6.578388
	0.646846	0.084448	1.557973	0.913886 0.472440
	0.609663	0.812048	1.898347	0.326617 2.809597
	6.775611	1.520231	1.876259	6.301908 2.266136
	1.219912	0.441328	1.379105	0.330736 1.393118
	0.807274	0.574617	0.728933	7.374760 3.563879
	0.098428	0.553111	0.865634	3.237800 3.636100
14				7.745829
	0.833370	0.956657	0.369640	7.303052 0.301065
	1.374525	6.150242	0.854640	0.644898 0.094029
	1.347852	1.620425	3.314441	1.359494 1.241980
	2.432865	0.241557	3.553560	0.786836 2.902025
	0.543024	0.099517	0.153494	0.101237 3.081636
	0.394670	0.868673	0.053994	1.659741 0.014653
	0.287563	5.539261	0.566381	1.332959 3.609149
	7.313160	0.364284	7.667193	0.943678 1.643535
	7.888797	0.540965	2.123306	2.305495 2.723219
	3.589852	2.060293	0.468052	1.961933 2.921719
15				0.851011
	0.942016	0.698967	0.074264	0.154884 1.043692
	0.171586	0.681208	0.542974	1.015823 3.500389
	0.107782	3.083272	5.238953	7.934817 0.458320
	1.668285	1.973634	1.338093	0.977718 0.911737
	0.267673	1.109255	1.837423	1.041247 6.842678
	3.571187	1.960754	3.458706	0.364667 0.405914
	0.306960	1.070609	0.224595	0.938701 1.221698
	0.216786	1.606151	6.939037	0.976597 0.053167
	1.059251	5.594004	1.715500	0.279995 6.560759
	0.801230	0.022837	2.125937	3.933505 0.233799
16				3.699193
	0.215889	0.664732	1.005284	0.133050 5.406394
	3.021731	1.801614	7.228200	0.367519 0.073391
	1.061363	1.084052	2.476023	1.977904 0.791609
	3.698169	1.970029	2.356825	2.680431 2.871157
	1.103063	1.555942	1.370127	2.879575 3.976436
	2.900531	3.230520	3.586778	1.491662 0.809475
	0.715944	2.612114	0.949345	1.941556 2.210463
	2.314770	1.994450	1.566886	0.426289 1.823052
	1.795949	0.642990	1.634107	0.115414 2.712005
	7.962248	1.841360	1.133106	0.113194 1.160383
17				0.484973
	0.552671	0.308872	0.952574	1.302069 0.759882

0.349989	0.680469	3.731639	3.785889	0.886460
3.202033	0.824910	6.979550	0.522567	0.566812
1.994670	0.433358	4.993072	3.378310	3.602074
0.401432	0.549815	0.395669	3.405417	1.986085
1.017286	3.302327	6.472351	0.424026	0.525528
3.041381	0.788784	1.360279	0.265655	0.533299
3.235621	0.053737	0.338215	3.080108	0.239669
0.507772	3.558443	0.767717	2.399214	0.689372
0.737999	0.223648	0.699898	0.186555	0.820247

18 1.007401

0.581575	0.507607	1.737101	7.416564	0.745468
0.516688	0.263291	1.859104	6.415384	1.440094
0.411893	0.549675	7.182021	1.197979	6.549686
2.258153	0.018833	0.064308	0.046850	0.851167
3.584702	1.433098	0.467432	7.626824	2.466694
0.958917	0.676505	7.487802	0.749721	1.108111
3.950588	0.679257	0.373439	0.007005	0.361553
0.106274	3.069311	0.345518	0.620135	2.444584
0.342632	0.109078	1.243543	1.485202	2.630543
0.524810	0.655707	4.580774	0.534220	0.051946

19 2.441658

4.437504	1.589648	0.776753	4.056809	0.414630
2.988247	0.015912	2.975341	0.176589	0.730038
0.300405	0.639370	1.021657	0.093903	0.264686
2.971983	0.303185	1.535634	0.924736	0.638060
1.011481	1.167607	0.399498	4.571012	0.630519
0.640725	0.169681	0.503115	0.982218	1.258591
7.486455	3.804725	0.569651	0.593290	1.306025
0.624134	3.224988	1.762219	0.130884	3.281239
0.871952	4.322533	1.577336	1.006309	0.469921
0.291133	4.140074	3.878900	0.503184	0.583163

20 0.838989

1.401482	0.608493	1.990479	1.018570	4.617070
1.758472	0.162425	1.148300	5.711913	0.466338
1.094418	0.592366	0.734068	0.877012	0.489691
3.859474	0.763988	2.432593	0.210942	1.584637
6.843458	0.003458	0.622970	3.413387	1.266865
0.880704	1.882438	0.842073	3.267805	0.646108
4.686092	4.856676	0.741307	1.520324	0.900358
2.036308	0.057316	0.590254	0.567204	0.778653
0.783542	1.578508	0.264005	0.016193	0.909136
2.472648	0.672261	2.014700	1.842699	0.497510

INITIAL BASIS, K=1, 70

71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130
131	132	133	134	135	136	137	138	139	140

OUTPUT

8 ITERATIONS

SOLUTION VALUES OF BASIC VARIABLES

ECONOMIC VARIABLES

ACTIVITY LEVEL X(2)=	0.021112
ACTIVITY LEVEL X(25)=	0.018510
ACTIVITY LEVEL X(29)=	0.010010
MARGINAL FACILITY VALUATION U(2)=	4.495007
MARGINAL FACILITY VALUATION U(6)=	0.428026
MARGINAL FACILITY VALUATION U(10)=	3.541015

SLACK VARIABLES

DUAL SLACK V(1)=	7.320710
DUAL SLACK V(3)=	7.791055
DUAL SLACK V(4)=	7.184075
DUAL SLACK V(5)=	15.552417
DUAL SLACK V(6)=	7.390942
DUAL SLACK V(7)=	29.215752
DUAL SLACK V(8)=	33.081086
DUAL SLACK V(9)=	16.110505
DUAL SLACK V(10)=	32.706248
DUAL SLACK V(11)=	15.474800
DUAL SLACK V(12)=	10.492417
DUAL SLACK V(13)=	5.879986
DUAL SLACK V(14)=	17.732275
DUAL SLACK V(15)=	2.709042
DUAL SLACK V(16)=	9.955833
DUAL SLACK V(17)=	26.993665
DUAL SLACK V(18)=	3.289459
DUAL SLACK V(19)=	20.292089
DUAL SLACK V(20)=	15.978314
DUAL SLACK V(21)=	7.128653
DUAL SLACK V(22)=	9.112649
DUAL SLACK V(23)=	30.767980
DUAL SLACK V(24)=	6.004341
DUAL SLACK V(26)=	8.407050
DUAL SLACK V(27)=	16.190134
DUAL SLACK V(28)=	11.160072
DUAL SLACK V(30)=	5.436733
DUAL SLACK V(31)=	0.203561
DUAL SLACK V(32)=	8.086773
DUAL SLACK V(33)=	5.026823
DUAL SLACK V(34)=	8.240968
DUAL SLACK V(35)=	17.569757
DUAL SLACK V(36)=	17.646915
DUAL SLACK V(37)=	7.920514

DUAL SLACK V(38)=	15.852342
DUAL SLACK V(39)=	7.946107
DUAL SLACK V(40)=	15.727504
DUAL SLACK V(41)=	4.178210
DUAL SLACK V(42)=	2.102439
DUAL SLACK V(43)=	10.019899
DUAL SLACK V(44)=	16.132071
DUAL SLACK V(45)=	24.529628
DUAL SLACK V(46)=	19.030908
DUAL SLACK V(47)=	26.950430
DUAL SLACK V(48)=	37.055451
DUAL SLACK V(49)=	8.474905
DUAL SLACK V(50)=	6.599315
PRIMAL SLACK S(1)=	0.587226
PRIMAL SLACK S(3)=	1.174107
PRIMAL SLACK S(4)=	0.430599
PRIMAL SLACK S(5)=	5.909796
PRIMAL SLACK S(7)=	1.048552
PRIMAL SLACK S(8)=	1.344284
PRIMAL SLACK S(9)=	0.725657
PRIMAL SLACK S(11)=	7.786201
PRIMAL SLACK S(12)=	2.107435
PRIMAL SLACK S(13)=	0.605557
PRIMAL SLACK S(14)=	7.651979
PRIMAL SLACK S(15)=	0.705949
PRIMAL SLACK S(16)=	3.596626
PRIMAL SLACK S(17)=	0.437446
PRIMAL SLACK S(18)=	0.943522
PRIMAL SLACK S(19)=	2.386595
PRIMAL SLACK S(20)=	0.769984

Exhibit C-12

Linear Programming Under Uncertainty

FORTRAN Statements

```
*    LIST8
*    LIBE
*    LABEL
C    SLPEXC
CSLPEXC
     COMMON I,J,K,L,M,N,IP,JP,KP,LP,MP,NP,MM,NN,LAST,IPR,IPC,MAXIT,ITMIN,
    1N,ITERNO,ITERNN,IREPET,AA,BA,GA,DA,CV,EP,FMTA,FMTB,FMTC,FMTP,FMTINV,
    2V,FMTY,FMTTAB,IBETA,IB,JV,IR,A,B,C,P,ALPHA,GAMMA,DMEAN,DSIGMA,INV,
    3DISTAB,DENTAB,DELTA,NTABLE,Y,W,DUMMY,FMTI
     DIMENSION IBETA(80),IB(161),JV(161),IR(80),A(30,50),B(30,50),
    1C(30),P(50),ALPHA(30),GAMMA(30),DMEAN(30),DSIGMA(30),INV(30),
    2DISTAB(500),DENTAB(500),FMTA(12),FMTB(12),FMTC(12),FMTP(12),
    3FMTINV(12),FMTI(12),FMTY(12),FMTTAB(12),W(80,241),Y(30),DUMMY(1000)
    4)
     READ INPUT TAPE 5,1,FMTTAB,NTABLE,DELTA
   1 FORMAT (12A6/(I5,F3.3))
     READ INPUT TAPE 5, FMTTAB, (DISTAB(K),K=1,NTABLE),(DENTAB(K),K=1,
    1NTABLE)
     WRITE OUTPUT TAPE 6,491
 491 FORMAT (1H1/////)
     WRITE OUTPUT TAPE 6,2
   2 FORMAT (//////////20X79HSTOCHASTIC LINEAR PROGRAMMING WITH PROPORTIONAL
    1IONAL LOSSES AND NORMAL DISTRIBUTIONS/////)
     NUMBER=0
   3 NUMBER=NUMBER+1
     CALL SLPROG(NUMBER)
     GO TO 3
     END
```

```
*    LIST8
*    LIBE
*    LABEL
CSLPROG
C                  STOCHASTIC LINEAR PROGRAMMING
C              WITH PROPORTIONAL LOSSES AND NORMAL DISTRIBUTIONS
C
C                        MAIN ROUTINE
     SUBROUTINE SLPROG(NUMBER)

     COMMON I,J,K,L,M,N,IP,JP,KP,LP,MP,NP,MM,NN,LAST,IPR,IPC,MAXIT,ITMIN,
    1N,ITERNO,ITERNN,IREPET,AA,BA,GA,DA,CV,EP,FMTA,FMTB,FMTC,FMTP,FMTINV,
    2V,FMTY,FMTTAB,IBETA,IB,JV,IR,A,B,C,P,ALPHA,GAMMA,DMEAN,DSIGMA,INV,
    3DISTAB,DENTAB,DELTA,NTABLE,Y,W,DUMMY,FMTI
     DIMENSION IBETA(80),IB(161),JV(161),IR(80),A(30,50),B(30,50),
    1C(30),P(50),ALPHA(30),GAMMA(30),DMEAN(30),DSIGMA(30),INV(30),
    2DISTAB(500),DENTAB(500),FMTA(12),FMTB(12),FMTC(12),FMTP(12),
    3FMTINV(12),FMTI(12),FMTY(12),FMTTAB(12),W(80,241),Y(30),DUMMY(1000)
    4)

     READ INPUT TAPE 5,1,JP,IP,NP,EP,MAXIT,ITMIN,CV
   1 FORMAT (3I3,E12.4,2I3,E12.4)

     ITERNO=0
     ITERNN=0
     MP=JP+IP+IP+NP
     MM=MP+MP
     LAST=MM+1
     NN=LAST+MP

     IF(MP) 2,2,3
   2 WRITE OUTPUT TAPE 6,30
  30 FORMAT (///46X27HPROBLEM DIMENSION TOO LARGE)
     CALL EXIT
   3 IF(MP-80) 4,4,2
   4 IF (MAXIT) 5,5,6
   5 MAXIT=30
   6 IF(JP) 2,2,7
   7 IF(JP-50) 8,8,2
   8 IF(IP) 2,2,9
   9 IF (IP-30) 10,10,2
  10 IF (NP) 2,2,11
  11 IF(NP-30) 14,14,2

  14 CALL INSLP(NUMBER)
  15 CALL FILLW
  16 CALL INVRB
  17 CALL TABLOW

     ITERNN=ITERNN+1
     IF(IREPET-5)40,15,40
  40 CONTINUE
     IF(ITERNN-ITMIN) 15,18,18
  18 IF (ITERNN-MAXIT) 19,19,22
  19 CONTINUE
  21 CALL ITERAT
     IF(IREPET-1)22,24,17
```

```
   24 ITMIN=ITMIN+1
      GO TO 15
   22 CONTINUE
   23 CALL OUTSLP
      RETURN
      END
```

```
*    LIST8
*    LIBE
*    LABEL
CINSLP
      SUBROUTINE INSLP(NUMBER)

      COMMON I,J,K,L,M,N,IP,JP,KP,LP,MP,NP,MM,NN,LAST,IPR,IPC,MAXIT,ITMIN,
     1N,ITERNO,ITERNN,IREPET,AA,BA,GA,DA,CV,EP,FMTA,FMTB,FMTC,FMTP,FMTINV,
     2V,FMTY,FMTTAB,IBETA,IB,JV,IR,A,B,C,P,ALPHA,GAMMA,DMEAN,DSIGMA,HNV,
     3DISTAB,DENTAB,DELTA,NTABLE,Y,W,DUMMY,FMTI
      COMMON PI
      DIMENSION IBETA(80),IB(161),JV(161),IR(80),A(30,50),B(30,50),
     1C(30),P(50),ALPHA(30),GAMMA(30),DMEAN(30),DSIGMA(30),HNV(30),
     2DISTAB(500),DENTAB(500),FMTA(12),FMTB(12),FMTC(12),FMTP(12),
     3FMTINV(12),FMTI(12),FMTY(12),FMTTAB(12),W(80,241),Y(30),DUMMY(1000)
     4)
      DIMENSION PI(50)
      DIMENSION FMTPI(12)

      WRITE OUTPUT TAPE 6,1,NUMBER
    1 FORMAT (////53X11HPROBLEM NO.I3)
      READ INPUT TAPE 5,2,FMTPI,FMTP,FMTINV,FMTA,FMTC,FMTB,FMTI
    2 FORMAT (12A6)
      READ INPUT TAPE 5,FMTPI,(PI(K),K=1,IP)
      READ INPUT TAPE 5,FMTP, (P(K),K=1,JP)
      READ INPUT TAPE 5,FMTINV, (HNV(K),K=1,IP)
      READ INPUT TAPE 5,FMTA,((A(K,L),L=1,JP),K=1,IP)
      READ INPUT TAPE 5,FMTC, (C(K),K=1,NP)
      READ INPUT TAPE 5,FMTB, ((B(K,L),L=1,JP),K=1,NP)

      DO 22 K=1,IP
      READ INPUT TAPE 5,FMTI, ALPHA(K),GAMMA(K),DMEAN(K),DSIGMA(K)
      GAMMA(K)=GAMMA(K)+ALPHA(K)
      IF (GAMMA(K))9,8,8
    9 WRITE OUTPUT TAPE 6,3,K
    3 FORMAT (30HOLOSS STRUCTURE NOT CONVEX, I=I4)
      CALL EXIT
    8 CONTINUE
      IF (DSIGMA(K))9,9,22
   22 CONTINUE
      DO 50 I=1,IP
   50 Y(I)=DMEAN(I)+DSIGMA(I)

      READ INPUT TAPE 5,20, (IB(K),K=1,MP)
   20 FORMAT(72I1)
      IB(LAST)=LAST
      DO 12 K=1,MP
      KP=K+MP
      IF (IB(K)) 13,13,14
   13 IB(K)=KP
      IB(KP)=K
      GO TO 12
   14 IB(K)=K
      IB(KP)=KP
   12 CONTINUE
      WRITE OUTPUT TAPE 6,30
```

```
30 FORMAT (////55X10HDATA INPUT)
   WRITE OUTPUT TAPE 6,31,IP,(PI(I),I=1,IP)
31 FORMAT (///44X30HPRICES OF OUTPUTS, PI(I), I=1,I2/(20X5F20.6))
   WRITE OUTPUT TAPE 6,32,JP,(P(J),J=1,JP)
32 FORMAT (///37X41HDIRECT COSTS OF ACTIVITIES, GAMMA(J),J=1,I2/(20X5F20.6))
  1F20.6))
   WRITE OUTPUT TAPE 6,33,IP,(HNV(I),I=1,IP)
33 FORMAT (///43X31HINITIAL INVENTORIES, H(I), I=1,I2/(20X5F20.6))
   WRITE OUTPUT TAPE 6,34,JP,IP
34 FORMAT (////31X47HMATRIX OF PRODUCTION COEFFICIENTS, A(I,J), J=1,I2
  12,6H, I=1,I2)
   DO 35 I=1,IP
35 WRITE OUTPUT TAPE 6,36,I,JP,(A(I,J),J=1,JP)
36 FORMAT (//1X2HI=I2,6H, J=1,I2,7X5F20.6/(20X5F20.6))
   WRITE OUTPUT TAPE 6,37,JP,NP
37 FORMAT (////27X54HCAPACITY CONSTRAINTS ON FACILITIES, C(N), B(N,J)
  1, J=1,I2,6H, N=1,I2)
   WRITE OUTPUT TAPE 6,38

38 FORMAT (//1X2H N,10X4HC(N),41X6HB(N,J))
   DO 39 N=1,NP
39 WRITE OUTPUT TAPE 6,40,N,C(N),(B(N,J),J=1,JP)
40 FORMAT (//1XI2,F15.6,2X5F20.6/(20X5F20.6))
   WRITE OUTPUT TAPE 6,41,IP
41 FORMAT (////48X20HLOSS STRUCTURE, I=1,I2//2X1HI29X8HALPHA(I),
  1  12X8HGAMMA(I),12X7HMEAN(I),13X8HSIGMA(I))
   WRITE OUTPUT TAPE 6,42,(I,ALPHA(I),GAMMA(I),DMEAN(I),DSIGMA(I),
  1  I=1,IP)

42 FORMAT (1XI2,17X4F20.6)
   WRITE OUTPUT TAPE 6,43,(IB(K),K=1,MP)
43 FORMAT (////53X13HINITIAL BASIS/(20X20I5))
   WRITE OUTPUT TAPE 6,44
44 FORMAT (////////// )
   RETURN
   END
```

```
      W(L,KP)=-B(K,L)
   10 W(KP,L)=B(K,L)
    9 CONTINUE
      CALL NEWY(0)
      DO 14 K=1,IP
      KP=K+JP
      W(KP,KP)=OLDDEL(K)
   14 W(KP,LAST)=V(K)
      DO 11 K=1,JP
   11 W(K,LAST)=-P(K)

      DO 12 K=1,NP
      KP=K+IP+IP+JP
   12 W(KP,LAST)=C(K)

      DO 13 K=1,IP
      KP=K+IP+JP
   13 W(KP,LAST)=HNV(K)

      DO 50 K=1,LAST
   50 JV(K)=K

      RETURN
      END
```

```
*    LIST8
*    LIBE
*    LABEL
CINVRB
     SUBROUTINE INVRB

     COMMON I,J,K,L,M,N,IP,JP,KP,LP,MP,NP,MM,NN,LAST,IPR,IPC,MAXIT,ITMIN,
    1N,ITERNO,ITERNN,IREPET,AA,BA,GA,DA,CV,EP,FMTA,FMTB,FMTC,FMTP,FMTINV,
    2V,FMTY,FMTTAB,IBETA,IB,JV,IR,A,B,C,P,ALPHA,GAMMA,DMEAN,DSIGMA,INV,
    3DISTAB,DENTAB,DELTA,NTABLE,Y,W,DUMMY,FMTI
     DIMENSION IBETA(80),IB(161),JV(161),IR(80),A(30,50),B(30,50),
    1C(30),P(50),ALPHA(30),GAMMA(30),DMEAN(30),DSIGMA(30),INV(30),
    2DISTAB(500),DENTAB(500),FMTA(12),FMTB(12),FMTC(12),FMTP(12),
    3FMTINV(12),FMTI(12),FMTY(12),FMTTAB(12),W(80,241),Y(30),DUMMY(1000)
    4)

     DO 1 K=1,MP
   1 IBETA(K)=0

     DO 8 K=1,MP
     KP=IB(K)
     J=JV(KP)
     AA=0.0

     DO 3 I=1,MP
     IF (IBETA(I)) 4,4,3
   4 DA=ABSF(W(I,J))
     IF (DA-AA) 3,3,5
   5 AA=DA
     IPR=I
   3 CONTINUE

     IF (AA-EP) 6,6,7
   6 WRITE OUTPUT TAPE 6,9,ITERNN,AA,EP
   9 FORMAT (///23X29HBASIS SINGULAR, ITERATION NO. I3, 9H,  PIVOT= E12.4
    1.4,8H,  MIN.=E12.4)
     CALL EXIT
   7 IR(K)=IPR
     IBETA(IPR)=1
     CALL PVTSLP (IPR,K,K,NN)
   8 CONTINUE

     RETURN
     END
```

```
*    LIST8
*    LABEL
CITERAT


*    LIST8
*    LIBE
*    LABEL
CTABLOW
     SUBROUTINE TABLOW
     COMMON I,J,K,L,M,N,IP,JP,KP,LP,MP,NP,MM,NN,LAST,IPR,IPC,MAXIT,ITMIN,
    1N,ITERNO,ITERNN,IREPET,AA,BA,GA,DA,CV,EP,FMTA,FMTB,FMTC,FMTP,FMTINV,
    2V,FMTY,FMTTAB,IBETA,IB,JV,IR,A,B,C,P,ALPHA,GAMMA,DMEAN,DSIGMA,INV,
    3DISTAB,DENTAB,DELTA,NTABLE,Y,W,DUMMY,FMTI
     DIMENSION IBETA(80),IB(161),JV(161),IR(80),A(30,50),B(30,50),
    1C(30),P(50),ALPHA(30),GAMMA(30),DMEAN(30),DSIGMA(30),INV(30),
    2DISTAB(500),DENTAB(500),FMTA(12),FMTB(12),FMTC(12),FMTP(12),
    3FMTINV(12),FMTI(12),FMTY(12),FMTTAB(12),W(80,241),Y(30),DUMMY(1000)
     4)

     IPP=JP+IP
     IPPP=MP+IPP
     KEY=0
     IF(IREPET-5)102,103,102
 103 KEY=1
 102 NUMB=0
     IREPET=0
  20 AA=0.0
     DO 4 K=1,MP
   2 KP=IR(K)
     DA=W(KP,LAST)
     IF(DA)21,4,4
  21 KP=IB(K)
     IF(KP-IPP)22,22,23
  23 IF(KP-IPPP)24,24,22
  22 AA=0.0
     DO 110 I=K,MP
     KP=IB(I)
     IF(KP-IPP)111,111,112
 112 IF(KP-IPPP)110,110,111
 111 KP=IR(I)
     DA=W(KP,LAST)
     IF(DA-AA)113,110,110
 113 AA=DA
     IPR=I
 110 CONTINUE
     IF(KEY)6,6,114
 114 IF(NUMB-MP)6,6,115
 115 M=IR(IPR)
     LPP=MP+1
     DO 116 L=LPP,MM
     J=IB(L)
     IF(J-IPP)117,117,118
 118 IF(J-IPPP)116,116,117
 117 J=JV(J)
     IF(W(M,J))119,116,116
```

```
119 IPC=L-MP
    JPR=IPR+MP
    GO TO 33
116 CONTINUE
    GO TO 6
 24 IF(DA-AA)3,4,4
  3 AA=DA
    IPR=K
  4 CONTINUE
    IF (AA) 6,5,5
  5 CALL STOREY
    RETURN

  6 IREPET=1
    LPR=IB(IPR)
    KP=IR(IPR)
    JPR=IPR+MP
  7 LP=IB(JPR)
  9 L=JV(LP)
 10 AA=W(KP,LAST)/W(KP,L)
 12 ITERNO=ITERNO+1
    GA=0.0
    DO 13 K=1,MP
    J=IB(K)
    IF(J-IPP)15,15,14
 14 IF(J-IPPP)80,80,15
 80 IF(LPR-IPP)15,15,81
 81 IF(LPR-IPPP)13,13,15
 15 I=IR(K)
    IF (W(I,L))13,13,16
 16 DA=W(I,LAST)/W(I,L)
    IF(DA)94,90,60
 94 WRITE OUTPUT TAPE 6,95,ITERNO,ITERNN
 95 FORMAT (/////22X44HPRIMAL INFEASIBILITY OCCURRED, ITERATION NO.I4,
   123H,   GROSS ITERATION NO.I4)
    GO TO 13
 90 W(I,LAST)=.0000001
    GO TO 16
 60 IF (GA) 18,18,17
 17 IF (DA-GA) 18,13,13
 18 GA=DA
    IPC=K
 13 CONTINUE


    IF(AA)40,40,31
 40 IF(GA)91,91,33
 31 IF(AA-GA)32,32,41
 41 IF(GA)32,32,33

 32 CALL PVTSLP(KP,JPR,MP+1,NN)
    CALL CHGIDX(IPR,JPR)
    DO 70 K=1,MP
    J=IB(K)
    IF(J-IPP)71,71,72
 72 IF(J-IPPP)70,70,71
```

```
 71 I=IR(K)
    IF(W(I,LAST))73,70,70
 73 NUMB=NUMB+1
    IF(NUMB-NN)20,20,104
104 IF(KEY)105,105,106
106 WRITE OUTPUT TAPE 6,107,ITERNN
107 FORMAT (/////50X19HFEASIBLE BASIS LOST/50X19HGROSS ITERATION NO. I
   13)
    WRITE OUTPUT TAPE 6,108,(IB(K),K=1,MP)
108 FORMAT (//53X13HCURRENT BASIS //(10X20I5))
    CALL OUTSLP
    CALL EXIT
105 IREPET=5
    RETURN
 70 CONTINUE
    GO TO 20

 33 CALL PVTSLP(IR(IPC),JPR,MP+1,NN)
    CALL CHGIDX(IPC,JPR)
    JPR=IPC+MP
    GO TO 7

 91 WRITE OUTPUT TAPE 6,92
 92 FORMAT (18HOSOLUTION INFINITE)
    CALL OUTSLP
    CALL EXIT
    END
```

```
      LIST8
      LIBE
      LABEL
ITERAT
      SUBROUTINE ITERAT

      COMMON I,J,K,L,M,N,IP,JP,KP,LP,MP,NP,MM,NN,LAST,IPR,IPC,MAXIT,ITMIN,
     1N,ITERNO,ITERNN,IREPET,AA,BA,GA,DA,CV,EP,FMTA,FMTB,FMTC,FMTP,FMTINV,
     2V,FMTY,FMTTAB,IBETA,IB,JV,IR,A,B,C,P,ALPHA,GAMMA,DMEAN,DSIGMA,INV,
     3DISTAB,DENTAB,DELTA,NTABLE,Y,W,DUMMY,FMTI
      DIMENSION IBETA(80),IB(161),JV(161),IR(80),A(30,50),B(30,50),
     1C(30),P(50),ALPHA(30),GAMMA(30),DMEAN(30),DSIGMA(30),INV(30),
     2DISTAB(500),DENTAB(500),FMTA(12),FMTB(12),FMTC(12),FMTP(12),
     3FMTINV(12),FMTI(12),FMTY(12),FMTTAB(12),W(80,241),Y(30),DUMMY(1000)
     4)

 1    J=IB(1)
      J=JV(J)
      DO 2 K=1,MP
 2    W(K,J)=W(K,LAST)
      CALL NULAST
      ITERNN=ITERNN+1
      IF(ITERNN-MAXIT)3,3,7
 7    IREPET=0
      RETURN
 3    AA=0.0
      DO 4 K=1,MP
      IF (W(K,LAST)+EP) 8,4,4
 4    AA=AA+(W(K,J)-W(K,LAST))**2
      IF (AA-CV) 6,6,5
 5    CALL STOREY
      GO TO 1
 8    IREPET=2
      RETURN
 6    CALL STOREY
      RETURN
      END
```

```
   LIST8
   LABEL
NEWY
   SUBROUTINE NEWY(KEY)

   COMMON I,J,K,L,M,N,IP,JP,KP,LP,MP,NP,MM,NN,LAST,IPR,IPC,MAXIT,ITMIN,
  1N,ITERNO,ITERNN,IREPET,AA,BA,GA,DA,CV,EP,FMTA,FMTB,FMTC,FMTP,FMTINV,
  2V,FMTY,FMTTAB,IBETA,IB,JV,IR,A,B,C,P,ALPHA,GAMMA,DMEAN,DSIGMA,INV,
  3DISTAB,DENTAB,DELTA,NTABLE,Y,W,DUMMY,FMTI
   COMMON PI
   COMMON V
   DIMENSION IBETA(80),IB(161),JV(161),IR(80),A(30,50),B(30,50),
  1C(30),P(50),ALPHA(30),GAMMA(30),DMEAN(30),DSIGMA(30),INV(30),
  2DISTAB(500),DENTAB(500),FMTA(12),FMTB(12),FMTC(12),FMTP(12),
  3FMTINV(12),FMTI(12),FMTY(12),FMTTAB(12),W(80,241),Y(30),DUMMY(1000)
  4)
   DIMENSION PI(50)
   DIMENSION V(30)
   COMMON OLDDEL
   DIMENSION OLDDEL(30)

   DO 1 K=1,IP
   BA=(Y(K)-DMEAN(K))/DSIGMA(K)
   BA=ABSF(BA/DELTA)+1.0
   LP=BA
   IF(KEY)8,8,19
19 LP=LP+1
   AA=LP
   GO TO 11
 8 IF(LP-NTABLE)2,3,3
 3 OLDDEL(K)=-.0000001
 4 GA=.0000001
   GO TO 6
 2 GA=DENTAB(LP)
   LP=LP+1
   AA=LP
   DA=DENTAB(LP)
   GA=DA-(DA-GA)*(AA-BA)
   OLDDEL(K)=-GAMMA(K)*GA/DSIGMA(K)
11 IF(LP-NTABLE)15,15,4
15 CONTINUE
   GA=DISTAB(LP)
   LP=LP-1
   DA=DISTAB(LP)
   GA=GA-(GA-DA)*(AA-BA)
 6 IF (Y(K)-DMEAN(K)) 1,1,7
 7 GA=1.0-GA
 1 V(K)=-PI(K)-ALPHA(K)+GAMMA(K)*GA+OLDDEL(K)*Y(K)
   RETURN
   END
```

```
*    LIST8
*    LIBE
*    LABEL
CSTOREY
     SUBROUTINE STOREY

     COMMON I,J,K,L,M,N,IP,JP,KP,LP,MP,NP,MM,NN,LAST,IPR,IPC,MAXIT,ITMIN,
    1N,ITERNO,ITERNN,IREPET,AA,BA,GA,DA,CV,EP,FMTA,FMTB,FMTC,FMTP,FMTINV,
    2V,FMTY,FMTTAB,IBETA,IB,JV,IR,A,B,C,P,ALPHA,GAMMA,DMEAN,DSIGMA,INV,
    3DISTAB,DENTAB,DELTA,NTABLE,Y,W,DUMMY,FMTI
     DIMENSION IBETA(80),IB(161),JV(161),IR(80),A(30,50),B(30,50),
    1C(30),P(50),ALPHA(30),GAMMA(30),DMEAN(30),DSIGMA(30),INV(30),
    2DISTAB(500),DENTAB(500),FMTA(12),FMTB(12),FMTC(12),FMTP(12),
    3FMTINV(12),FMTI(12),FMTY(12),FMTTAB(12),W(80,241),Y(30),DUMMY(1000)
    4)

     DO 1 K=1,IP
   1 Y(K)=0.0
     DO 2 K=1,MP
     L=IB(K)
     IF (L-JP) 2,2,3
   3 IF (L-JP-IP) 4,4,2
   4 LP=L-JP
     L=IR(K)
     Y(LP)=W(L,LAST)
   2 CONTINUE
     RETURN
     END
```

```
*    LIST8
*    LIBE
*    LABEL
COUTSLP
     SUBROUTINE OUTSLP
     COMMON I,J,K,L,M,N,IP,JP,KP,LP,MP,NP,MM,NN,LAST,IPR,IPC,MAXIT,ITMIN,
    1N,ITERNO,ITERNN,IREPET,AA,BA,GA,DA,CV,EP,FMTA,FMTB,FMTC,FMTP,FMTINV,
    2V,FMTY,FMTTAB,IBETA,IB,JV,IR,A,B,C,P,ALPHA,GAMMA,DMEAN,DSIGMA,INV,
    3DISTAB,DENTAB,DELTA,NTABLE,Y,W,DUMMY,FMTI
     DIMENSION IBETA(80),IB(161),JV(161),IR(80),A(30,50),B(30,50),
    1C(30),P(50),ALPHA(30),GAMMA(30),DMEAN(30),DSIGMA(30),INV(30),
    2DISTAB(500),DENTAB(500),FMTA(12),FMTB(12),FMTC(12),FMTP(12),
    3FMTINV(12),FMTI(12),FMTY(12),FMTTAB(12),W(80,241),Y(30),DUMMY(1000)
    4)
     WRITE OUTPUT TAPE 6,493
493  FORMAT (1H1/////)
     WRITE OUTPUT TAPE 6,11
 11  FORMAT (//////57X6HOUTPUT///)
     WRITE OUTPUT TAPE 6,2
  2  FORMAT (///43X34HSOLUTION VALUES OF BASIC VARIABLES//)
     WRITE OUTPUT TAPE 6,499
499  FORMAT (/51X18HECONOMIC VARIABLES//)
     DO 3 LP=1,MM
     IF(LP-MP-1)490,497,490
497  WRITE OUTPUT TAPE 6,498
498  FORMAT (//52X15HSLACK VARIABLES//)
490  DO 495 K=1,MP
     J=IB(K)
     IF(J-LP)495,494,495
495  CONTINUE
     GO TO 3
494  KP=IR(K)
     LP=IB(K)
     IF (LP-JP) 20,20,22
 20  WRITE OUTPUT TAPE 6,21,LP,W(KP,LAST)
 21  FORMAT (39X17HACTIVITY LEVEL X(I2,2H)=F20.6)
     GO TO 3
 22  IF(LP-JP-IP)30,30,32
 30  LP=LP-JP
     WRITE OUTPUT TAPE 6,31,LP,W(KP,LAST)
 31  FORMAT (36X20HPRODUCTION OUTPUT Y(I2,2H)=F20.6)
     GO TO 3
 32  IF(LP-MP)117,117,25
117  IF(LP-JP-IP-IP)118,118,23
118  LP=LP-JP-IP
     WRITE OUTPUT TAPE 6,42,LP,W(KP,LAST)
 42  FORMAT (15X41HEXPECTED MARGINAL VALUATION OF DEMAND MU(I2,2H)=F20.6)
    16)
     GO TO 3
 23  LP=LP-JP-IP-IP
     WRITE OUTPUT TAPE 6,24,LP,W(KP,LAST)
 24  FORMAT (26X30HMARGINAL FACILITY VALUATION U(I2,2H)=F20.6)
     GO TO 3
```

```
 25 IF(LP-MP-JP-IP)26,26,28
 26 LP=LP-MP
    WRITE OUTPUT TAPE 6,27,LP,W(KP,LAST)
 27 FORMAT (43X13HDUAL SLACK V(I2,2H)=F20.6)
    GO TO 3
 28 LP=LP-MP-JP-IP
    WRITE OUTPUT TAPE 6,29,LP,W(KP,LAST)
 29 FORMAT (41X15HPRIMAL SLACK S(I2,2H)=F20.6)
  3 CONTINUE
    WRITE OUTPUT TAPE 6,496
496 FORMAT (///31X58HSOLUTION VALUES OF NONBASIC VARIABLES ARE IDENTICALLY
   1ALLY ZERO///)
    WRITE OUTPUT TAPE 6,1, ITERNO,ITERNN,MAXIT,AA,CV
  1 FORMAT (////////////54X12HCOMPUTATIONS//40X20HTABLEAU ITERATIONS =
   1I13/40X20HGROSS ITERATIONS   =I13/40X20HMAX. NO. ALLOWED    =I13/40X20H
   2X20HCONV. MEASURE      =E20.2/40X20HMAX. ALLOWED          =E20.2)
    IF(ITMIN)40,40,41
 40 ITMIN=1
 41 CONTINUE
    WRITE OUTPUT TAPE 6,10,ITMIN
 10 FORMAT (40X20HCOMPLETE INVERSIONS=I13)
    WRITE OUTPUT TAPE 6,492
492 FORMAT (1H1/////)
  6 RETURN
    END
```

```
*     LIST8
*     LIBE
*     LABEL
CPVTSLP
      SUBROUTINE PVTSLP (IPROW,JPCOL, MMIN, MMAX)

      COMMON I,J,K,L,M,N,IP,JP,KP,LP,MP,NP,MM,NN,LAST,IPR,IPC,MAXIT,ITMIN,
     1N,ITERNO,ITERNN,IREPET,AA,BA,GA,DA,CV,EP,FMTA,FMTB,FMTC,FMTP,FMTINV,
     2V,FMTY,FMTTAB,IBETA,IB,JV,IR,A,B,C,P,ALPHA,GAMMA,DMEAN,DSIGMA,INV,
     3DISTAB,DENTAB,DELTA,NTABLE,Y,W,DUMMY,FMTI
      DIMENSION IBETA(80),IB(161),JV(161),IR(80),A(30,50),B(30,50),
     1C(30),P(50),ALPHA(30),GAMMA(30),DMEAN(30),DSIGMA(30),INV(30),
     2DISTAB(500),DENTAB(500),FMTA(12),FMTB(12),FMTC(12),FMTP(12),
     3FMTINV(12),FMTI(12),FMTY(12),FMTTAB(12),W(80,241),Y(30),DUMMY(1000)
     4)

      LP=IB(JPCOL)
      J=JV(LP)
      AA=W(IPROW,J)

      IF (AA) 107,106,107
  106 STOP 00005
  107 CONTINUE

      DO 1 I=1,MP
    1 W(I,J)=-W(I,J)/AA
      W(IPROW,J)=1.0/AA

      DO 2  L=MMIN,MMAX
      IF (L-JPCOL) 3,2,3
    3 IF(L-LAST)5,4,4
    5 LP=IB(L)
      LP=JV(LP)
      GO TO 6
    4 LP=L
    6 DA=W(IPROW,LP)
      IF (DA) 7,2,7
    7 DO 8  K=1,MP
    8 W(K,LP)=W(K,LP)+DA*W(K,J)
      W(IPROW,LP)=DA/AA
    2 CONTINUE
      RETURN
      END
```

```
*    LIST8
*    LIBE
*    LABEL
CCHGIDX
     SUBROUTINE CHGIDX (MU,NU)

     COMMON I,J,K,L,M,N,IP,JP,KP,LP,MP,NP,MM,NN,LAST,IPR,IPC,MAXIT,ITMIN,
    1N,ITERNO,ITERNN,IREPET,AA,BA,GA,DA,CV,EP,FMTA,FMTB,FMTC,FMTP,FMTINV,
    2V,FMTY,FMTTAB,IBETA,IB,JV,IR,A,B,C,P,ALPHA,GAMMA,DMEAN,DSIGMA,INV,
    3DISTAB,DENTAB,DELTA,NTABLE,Y,W,DUMMY,FMTI
     DIMENSION IBETA(80),IB(161),JV(161),IR(80),A(30,50),B(30,50),
    1C(30),P(50),ALPHA(30),GAMMA(30),DMEAN(30),DSIGMA(30),INV(30),
    2DISTAB(500),DENTAB(500),FMTA(12),FMTB(12),FMTC(12),FMTP(12),
    3FMTINV(12),FMTI(12),FMTY(12),FMTTAB(12),W(80,241),Y(30),DUMMY(1000)
    4)

     K=IB(MU)
     L=JV(K)
     I=IB(NU)
     J=JV(I)
     IB(MU)=I
     JV(K)=J
     IB(NU)=K
     JV(I)=L
     RETURN
     END
```

```
*    LIST8
*    LIBE
*    LABEL
CSLPRDM
     SUBROUTINE INSLP(NUMBER)
C    INSLP FOR RANDOM GENERATION

     COMMON I,J,K,L,M,N,IP,JP,KP,LP,MP,NP,MM,NN,LAST,IPR,IPC,MAXIT,ITMIN,'
    1N,ITERNO,ITERNN,IREPET,AA,BA,GA,DA,CV,EP,FMTA,FMTB,FMTC,FMTP,FMTINV,
    2V,FMTY,FMTTAB,IBETA,IB,JV,IR,A,B,C,P,ALPHA,GAMMA,DMEAN,DSIGMA,HNV,
    3DISTAB,DENTAB,DELTA,NTABLE,Y,W,DUMMY,FMTI
     COMMON PI
     DIMENSION IBETA(80),IB(161),JV(161),IR(80),A(30,50),B(30,50),
    1C(30),P(50),ALPHA(30),GAMMA(30),DMEAN(30),DSIGMA(30),HNV(30),
    2DISTAB(500),DENTAB(500),FMTA(12),FMTB(12),FMTC(12),FMTP(12),
    3FMTINV(12),FMTI(12),FMTY(12),FMTTAB(12),W(80,241),Y(30),DUMMY(1000)
    4)
     DIMENSION PI(50)
     DIMENSION FMTPI(12)

     WRITE OUTPUT TAPE 6,1,NUMBER
   1 FORMAT (////53X11HPROBLEM NO. I3)
     DO 50 I=1,IP
     CALL RANDM(XX)
     PI(I)=ABSF(XX)
     CALL RANDM(XX)
     HNV(I)=ABSF(XX)/(ABSF(XX)+100.00)
     DO 51 J=1,JP
     CALL RANDM(XX)
  51 A(I,J)=ABSF(XX)
     CALL RANDM(XX)
     ALPHA(I)=PI(I)*(1.0+ABSF(XX)/(ABSF(XX)+100.00))
     CALL RANDM(XX)
     GAMMA(I)=ALPHA(I)*(1.0+ABSF(XX)/(ABSF(XX)+300.00))
     CALL RANDM(XX)
     DMEAN(I)=ABSF(XX)
     CALL RANDM(XX)
     DSIGMA(I)=DMEAN(I)*(1.0+ABSF(XX)/(ABSF(XX)+50.00))
  50 Y(I)=DMEAN(I)+DSIGMA(I)
     DO 52 J=1,JP
     CALL RANDM(XX)
     P(J)=-ABSF(XX)
     DO 53 N=1,NP
     CALL RANDM(XX)
     B(N,J)=ABSF(XX)
     IF(J-1)54,54,53
  54 CALL RANDM(XX)
     C(N)=ABSF(XX)
  53 CONTINUE
  52 CONTINUE
     DO 55 M=1,MP
     K=MP+M
     IB(M)=K
  55 IB(K)=M
     WRITE OUTPUT TAPE 6,30
  30 FORMAT (////55X10HDATA INPUT)
     WRITE OUTPUT TAPE 6,31,IP,(PI(I),I=1,IP)
```

Exhibit C-13

Linear Programming Under Uncertainty

Sample Output

STOCHASTIC LINEAR PROGRAMMING WITH PROPORTIONAL LOSSES AND NORMAL DISTRIBUTIONS

PROBLEM NO. 1

DATA INPUT

PRICES OF OUTPUTS, PI(I), I=1, 3

1.622071	0.886959	0.827645

DIRECT COSTS OF ACTIVITIES, GAMMA(J),J=1,15

-1.147306	-0.802272	-0.217228	-1.252260	-0.542117
-0.668750	-1.041234	-2.829689	-1.497317	-1.241434
-0.760716	-2.482318	-1.582731	-0.096137	-0.364484

INITIAL INVENTORIES, H(I), I=1, 3

0.017029	0.025754	0.004806

MATRIX OF PRODUCTION COEFFICIENTS, A(I,J), J=1,15, I=1, 3

I= 1, J=1,15	0.598640	0.897959	2.775512	0.326535	0.244901
	0.734703	0.816439	0.612329	1.673976	2.043854
	1.065781	1.197344	1.592032	1.552192	0.328288
I= 2, J=1,15	1.965269	3.791615	3.374844	2.124533	1.186799

	1.560397	1.362382	0.043574	0.065360	0.196081
	0.588244	1.529466	1.176797	0.882598	2.591174
I= 3, J=1,15	0.724401	0.692813	0.519610	1.117660	1.352980
	0.117882	0.088412	0.265235	0.795704	1.548448
	0.322672	0.484008	0.904045	0.712136	0.136409

CAPACITY CONSTRAINTS ON FACILITIES, C(N), B(N,J), J=1,15, N=1,10

N	C(N)	B(N,J)				
1	0.651510	1.441918	0.406817	0.162921	0.939195	1.252700
		0.006250	1.123701	0.489068	0.983903	1.724303
		0.570537	1.723477	1.496388	0.144205	0.546726
2	0.931795	0.488632	0.610225	0.488763	3.270343	1.758099
		0.009375	1.371103	0.366801	0.737927	2.345819
		1.423220	2.340861	0.244581	0.432616	1.280354
3	0.386152	0.795384	1.661349	0.932575	1.811030	2.548596
		0.028124	0.226616	0.200807	0.855126	1.518729
		0.539321	1.511291	0.366872	0.595696	1.841061
4	1.475372	0.579229	1.968092	0.797726	0.716545	1.822894
		0.084372	0.169962	0.301211	0.641345	1.112373
		0.404490	1.067748	0.201233	0.893545	3.046365
5	0.639172	0.852229	0.952138	0.393177	0.149635	2.937363
		0.253115	0.509887	0.903633	1.848068	0.834280
		0.426943	0.800811	0.301849	2.722536	1.139094
6	3.010192	1.835032	0.856414	0.589766	0.224453	0.812088
		0.759344	1.059320	2.843599	3.088407	2.011361
		0.640414	1.609734	0.905548	0.167609	0.854321
7	0.772931	1.030575	0.569242	1.538594	0.673359	0.609066
		1.112129	1.177961	0.530798	1.265221	1.017041
		1.842483	0.414602	2.866571	0.125707	2.251847
8	0.956383	1.275178	0.853863	1.231565	0.080314	1.654398
		0.834097	1.533883	0.398098	0.948915	1.051124
		3.054901	0.621902	0.599714	0.377120	1.377771
9	2.429798	3.476599	2.246359	0.923674	0.060235	1.926389
		2.009163	1.203299	0.388591	3.386985	1.153373
		1.164702	1.731415	0.449785	0.262720	0.266626

10	1.868181	1.644697	1.369538	3.084087	0.180706	0.889583
		1.013745	0.902474	0.582886	2.160956	1.460119
		0.873526	2.388487	0.698712	0.394081	0.199970

LOSS STRUCTURE, I=1, 3

I	ALPHA(I)	GAMMA(I)	MEAN(I)	SIGMA(I)
1	1.630019	1.635190	0.863769	0.873864
2	0.903384	0.913274	1.961699	1.997994
3	0.829335	0.831028	1.683034	1.750817

INITIAL BASIS

32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
52	53	54	55	56	57	58	59	60	61	62									

OUTPUT

SOLUTION VALUES OF BASIC VARIABLES

ECONOMIC VARIABLES

ACTIVITY LEVEL X(3)=	0.299381
ACTIVITY LEVEL X(6)=	0.261921
ACTIVITY LEVEL X(14)=	0.167185
PRODUCTION OUTPUT Y(1)=	1.299900
PRODUCTION OUTPUT Y(2)=	1.592375
PRODUCTION OUTPUT Y(3)=	0.310301
EXPECTED MARGINAL VALUATION OF DEMAND MU(1)=	2.121893
EXPECTED MARGINAL VALUATION OF DEMAND MU(2)=	1.400677
EXPECTED MARGINAL VALUATION OF DEMAND MU(3)=	1.476988
MARGINAL FACILITY VALUATION U(3)=	7.451588
MARGINAL FACILITY VALUATION U(5)=	0.260941
MARGINAL FACILITY VALUATION U(7)=	2.674437

SLACK VARIABLES

DUAL SLACK V(1)=	4.959882
DUAL SLACK V(2)=	6.713334
DUAL SLACK V(4)=	11.267787
DUAL SLACK V(5)=	17.748277
DUAL SLACK V(7)=	2.242080
DUAL SLACK V(8)=	4.229325
DUAL SLACK V(9)=	6.916561
DUAL SLACK V(10)=	8.597561
DUAL SLACK V(11)=	6.256525
DUAL SLACK V(12)=	9.663831
DUAL SLACK V(13)=	5.700041
DUAL SLACK V(15)=	15.875506
PRIMAL SLACK S(4)=	0.576989
PRIMAL SLACK S(5)=	0.710687
PRIMAL SLACK S(7)=	1.065062
PRIMAL SLACK S(9)=	2.606717
PRIMAL SLACK S(11)=	0.306160
PRIMAL SLACK S(12)=	1.583103
PRIMAL SLACK S(13)=	0.613461

SOLUTION VALUES OF NONBASIC VARIABLES ARE IDENTICALLY ZERO

COMPUTATIONS

TABLEAU ITERATIONS	=	12
GROSS ITERATIONS	=	6
MAX. NO. ALLOWED	=	50
CONV. MEASURE	=	0.33E-12
MAX. ALLOWED	=	1.00E-04
COMPLETE INVERSIONS	=	5

PROBLEM NO. 2

DATA INPUT

PRICES OF OUTPUTS, PI(I), I=1, 3

0.599909	0.431341	0.574155

DIRECT COSTS OF ACTIVITIES, GAMMA(J),J=1,15

-1.904591	-1.852040	-0.839413	-1.091780	-0.807521
-0.878139	-0.730708	-1.446439	-0.318663	-0.278887
-0.009157	-0.889112	-0.876698	-0.393373	-0.765451

INITIAL INVENTORIES, H(I), I=1, 3

0.015743	0.005846	0.014243

MATRIX OF PRODUCTION COEFFICIENTS, A(I,J), J=1,15, I=1, 3

I= 1, J=1,15	1.596713	0.395070	0.592605	1.555632	1.333791
	0.000686	0.001030	0.003089	0.009266	0.027798
	0.083393	0.250180	0.750540	1.006482	0.754861
I= 2, J=1,15	0.882066	2.584788	1.877182	3.263090	1.789270
	0.683904	0.051713	0.077569	0.232708	0.698125
	0.377498	0.283124	0.849371	2.192451	1.288676
I= 3, J=1,15	0.669571	0.502178	1.013068	1.039205	1.117614
	1.352843	0.117060	0.087795	0.263386	0.790157
	1.481878	0.222817	0.334226	0.005356	0.008034

CAPACITY CONSTRAINTS ON FACILITIES, C(N), B(N,J), J=1,15, N=1,10

N	C(N)	B(N,J)

1	2.282634	3.427545	0.778060	2.072959	1.275341	1.690249
		2.537671	0.768496	0.678634	0.477995	0.418331
		0.027472	0.666834	2.520374	0.360235	1.185416
2	0.543709	1.423951	0.334181	1.109439	1.826023	0.535373
		1.806506	0.576372	0.508975	0.867971	0.509984
		0.082417	0.002007	1.780561	0.540353	0.889062
3	0.446691	0.407782	0.501271	1.328318	2.956139	0.803060
		2.839037	1.458230	1.053851	0.603914	0.764976
		0.247252	0.001505	2.683367	1.242117	2.668747
4	0.040432	0.670036	1.007626	1.984953	0.868418	1.636723
		0.517111	0.749379	1.161554	0.905871	1.179715
		0.741757	0.004516	0.050102	1.726352	0.006241
5	0.090972	0.030324	1.022879	3.909717	0.651313	0.455084
		0.387834	0.562035	1.484662	2.870450	0.884786
		0.901082	0.013548	0.037577	2.358109	0.004681
6	0.818750	0.272917	1.068637	3.729150	1.907881	0.682626
		0.327002	1.372208	0.907973	0.611349	2.617435
		0.675812	0.040645	0.112730	1.537164	0.014043
7	0.737506	1.825004	1.205911	3.187451	3.447285	0.191513
		0.490503	0.233245	0.680979	0.458512	1.926152
		0.109742	0.121935	0.338190	1.222985	0.042129
8	0.318778	0.212519	1.617734	1.562354	2.341855	0.143635
		0.943015	0.174934	0.171753	0.751070	3.556912
		0.082307	0.365804	0.029139	0.917239	0.126386
9	3.476001	0.956333	1.706406	0.343531	1.512782	0.430904
		0.829046	0.524802	0.128815	0.253210	2.670737
		0.246920	0.194822	0.043708	3.006867	0.379159
10	1.642007	2.428004	0.559609	0.515297	1.076694	0.585426
		0.487139	1.148813	0.386444	0.379815	0.012210
		0.740759	0.292233	0.131124	1.020602	0.274955

LOSS STRUCTURE, I=1, 3

I	ALPHA(I)	GAMMA(I)	MEAN(I)	SIGMA(I)
1	0.606191	0.607791	1.525040	1.533761
2	0.439242	0.443872	1.588513	1.600581

0.574293 0.574431 0.216922 0.219709

INITIAL BASIS

32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51
52 53 54 55 56 57 58 59 60 61 62

. OUTPUT

SOLUTION VALUES OF BASIC VARIABLES

ECONOMIC VARIABLES

ACTIVITY LEVEL X(4)=	0.045719
ACTIVITY LEVEL X(15)=	0.116736
PRODUCTION OUTPUT Y(1)=	0.174984
PRODUCTION OUTPUT Y(2)=	0.305467
PRODUCTION OUTPUT Y(3)=	0.062693
EXPECTED MARGINAL VALUATION OF DEMAND MU(1)=	1.090937
EXPECTED MARGINAL VALUATION OF DEMAND MU(2)=	0.776729
EXPECTED MARGINAL VALUATION OF DEMAND MU(3)=	1.009784
MARGINAL FACILITY VALUATION U(3)=	0.391695
MARGINAL FACILITY VALUATION U(4)=	3.490624

SLACK VARIABLES

DUAL SLACK V(1)=	1.300000
DUAL SLACK V(2)=	2.619862
DUAL SLACK V(3)=	5.160896
DUAL SLACK V(5)=	2.861851
DUAL SLACK V(6)=	1.897181
DUAL SLACK V(7)=	3.758195
DUAL SLACK V(8)=	5.761502
DUAL SLACK V(9)=	3.260446
DUAL SLACK V(10)=	3.325999
DUAL SLACK V(11)=	0.814633
DUAL SLACK V(12)=	0.187627
DUAL SLACK V(13)=	0.286627
DUAL SLACK V(14)=	4.099591
PRIMAL SLACK S(4)=	2.085946
PRIMAL SLACK S(5)=	0.356439
PRIMAL SLACK S(8)=	0.060648
PRIMAL SLACK S(9)=	0.729884
PRIMAL SLACK S(10)=	0.574980
PRIMAL SLACK S(11)=	0.196956
PRIMAL SLACK S(12)=	3.362577
PRIMAL SLACK S(13)=	1.560684

SOLUTION VALUES OF NONBASIC VARIABLES ARE IDENTICALLY ZERO

COMPUTATIONS

TABLEAU ITERATIONS	=	14
GROSS ITERATIONS	=	6
MAX. NO. ALLOWED	=	50
CONV. MEASURE	=	0.40E-13
MAX. ALLOWED	=	1.00E-04
COMPLETE INVERSIONS	=	5

PROBLEM NO. 3

DATA INPUT

PRICES OF OUTPUTS, PI(I), I=1, 3

0.412433	0.884908	0.951195

DIRECT COSTS OF ACTIVITIES, GAMMA(J),J=1,15

-0.681388	-0.053141	-2.830486	-0.078262	-1.803926
-0.067422	-0.528216	-1.768245	-1.294780	-0.749343
-3.794129	-2.629890	-0.284156	-0.345058	-0.906513

INITIAL INVENTORIES, H(I), I=1, 3

0.004724	0.006505	0.033016

MATRIX OF PRODUCTION COEFFICIENTS, A(I,J), J=1,15, I=1, 3

I= 1, J=1,15	0.711899	0.542790	0.407092	0.442555	0.663832
	1.982993	3.897957	3.693870	3.081609	1.244827
	0.933620	3.203440	1.610320	0.415480	0.623220
I= 2, J=1,15	0.982085	3.785022	3.355067	2.065200	1.097800
	1.293400	1.880199	3.281192	1.843575	0.765363
	0.296088	0.444132	0.664793	0.997190	3.966275
I= 3, J=1,15	2.243005	1.364507	0.187043	0.140282	0.420847
	0.525085	0.787627	1.451523	0.177284	0.265926
	0.797779	1.573346	0.360020	0.540029	1.240176

CAPACITY CONSTRAINTS ON FACILITIES, C(N), B(N,J), J=1,15, N=1,10

N	C(N)	B(N,J)

1	0.132493	0.176657	0.159422	0.491459	0.117392	2.823558
		0.202265	1.169296	2.609470	1.884340	0.562007
		3.382388	1.944835	0.852467	0.070345	2.878160
2	0.384872	0.397479	0.478266	0.368594	0.352177	0.470674
		0.606794	1.507888	1.914205	3.306038	1.372042
		2.147164	3.669011	2.229608	0.105518	0.634479
3	1.463846	0.577308	0.869595	0.211565	0.113063	0.353005
		1.640765	1.047331	3.485229	1.918114	0.232250
		1.220746	3.007032	1.344412	0.316553	0.475859
4	0.587307	0.783076	0.608786	0.317348	0.169594	0.118031
		1.844587	0.785498	2.455688	0.877172	0.174188
		1.662238	1.021096	0.066475	0.949660	0.855156
5	1.143052	1.523842	0.913179	0.952043	0.508782	0.177047
		0.766881	1.425976	1.683532	0.631515	0.522563
		1.973429	0.765822	0.049856	3.395920	0.565467
6	2.287467	0.857289	2.958150	3.424512	1.052691	0.531142
		0.300644	0.138964	2.101192	0.947273	1.135379
		0.960144	1.189863	0.149568	2.187762	0.848201
7	0.587203	1.431200	0.874450	2.273537	1.158073	1.186850
		0.450966	0.208446	1.151787	3.367276	1.406137
		0.880431	0.892397	0.448704	1.281642	2.178414
8	0.642413	0.440402	0.655838	1.410306	1.474219	1.560550
		0.705794	0.625338	1.455362	2.101827	0.436820
		0.641294	2.708764	0.692226	1.844927	1.267621
9	3.563428	0.963619	1.935027	0.461836	0.845317	1.363298
		0.117381	1.752027	0.732173	1.152741	0.327615
		0.961941	0.126291	0.076679	3.069561	1.802864
10	0.070854	2.690285	3.610162	0.346377	0.633988	0.044948
		0.176072	2.512163	0.549130	1.458224	0.982844
		3.543297	0.094719	0.115019	1.208684	2.817183

LOSS STRUCTURE, I=1, 3

I	ALPHA(I)	GAMMA(I)	MEAN(I)	SIGMA(I)
1	0.419484	0.422863	1.653879	1.715139
2	0.918114	0.929289	3.089420	3.165845
3	0.967283	0.974716	1.484753	1.511250

INITIAL BASIS

32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
52	53	54	55	56	57	58	59	60	61	62									

OUTPUT

SOLUTION VALUES OF BASIC VARIABLES

ECONOMIC VARIABLES

ACTIVITY LEVEL X(4)=	0.009979
ACTIVITY LEVEL X(6)=	0.314616
ACTIVITY LEVEL X(13)=	0.079400
PRODUCTION OUTPUT Y(1)=	0.760880
PRODUCTION OUTPUT Y(2)=	0.486822
PRODUCTION OUTPUT Y(3)=	0.228201
EXPECTED MARGINAL VALUATION OF DEMAND MU(1)=	0.704486
EXPECTED MARGINAL VALUATION OF DEMAND MU(2)=	1.611935
EXPECTED MARGINAL VALUATION OF DEMAND MU(3)=	1.720646
MARGINAL FACILITY VALUATION U(1)=	2.158095
MARGINAL FACILITY VALUATION U(4)=	1.610787
MARGINAL FACILITY VALUATION U(10)=	5.169393

SLACK VARIABLES

DUAL SLACK V(1)=	10.287143
DUAL SLACK V(2)=	11.208732
DUAL SLACK V(3)=	0.176065
DUAL SLACK V(5)=	5.358537
DUAL SLACK V(7)=	10.171257
DUAL SLACK V(8)=	3.805086
DUAL SLACK V(9)=	8.864716
DUAL SLACK V(10)=	4.755251
DUAL SLACK V(11)=	29.580158
DUAL SLACK V(12)=	3.281572
DUAL SLACK V(14)=	5.445431
DUAL SLACK V(15)=	14.092129
PRIMAL SLACK S(5)=	0.013420
PRIMAL SLACK S(6)=	0.839761
PRIMAL SLACK S(8)=	0.892743
PRIMAL SLACK S(9)=	2.170499
PRIMAL SLACK S(10)=	0.398138
PRIMAL SLACK S(11)=	0.350685
PRIMAL SLACK S(12)=	3.511974

SOLUTION VALUES OF NONBASIC VARIABLES ARE IDENTICALLY ZERO

COMPUTATIONS

TABLEAU ITERATIONS	=	18
GROSS ITERATIONS	=	6
MAX. NO. ALLOWED	=	50
CONV. MEASURE	=	0.64E-13
MAX. ALLOWED	=	1.00E-04
COMPLETE INVERSIONS	=	5

Exhibit C-14

Variable-Factor Programming

FORTRAN Statements

```
      BLEXEC    ***BLENDING EXECUTIVE ROUTINE***
*     LIST8
*     LABEL
CBLEXEC
      COMMON N
      N=0
    1 N=N+1
      CALL BLEND
      GO TO 1
      END
```

```
*    LIST8
*    SYMBOL TABLE
*    LABEL
CBLEND
     SUBROUTINE BLEND
C    BLEND          ***BLENDING MAIN SUBROUTINE***
     COMMON NUMBER,ITERNO,ITERNN,MAXIT,MINIT,CV,EP,IREPET,JP,NP,KP,MP,
    1   MM,LP,NN,LL,AA,BA,GA,DA,KIND,KPP,LPP
     COMMON THETA
     COMMON VMU,X,F,Z,DEL,BB,OLDZ,R,AM,A,B,IB,JV,IR,IBB
     DIMENSION VMU(20),X(50),F(50),Z(20,50),DEL(20,20),BB(20),OLDZ(20,50)
    10),R(80,241),AM(40,50),A(40),B(20),IB(160),JV(160),IR(80),IBB(160)
     IREPET=0
     ITERNO=0
     ITERNN=0
     CALL BINPUT
     IF(IREPET-9)1,9,10
   1 CALL BFILLT
     IF(IREPET-9)2,9,10
   2 CALL BINVER
     IF(IREPET-9)3,9,10
   3 CALL BSOLVE
     IF(IREPET-9)4,9,10
   4 IF(IREPET-7)6,1,5
   5 CONTINUE
   6 CALL BCONVR
     IF(IREPET-9)7,9,10
   7 IF(IREPET-1)9,1,8
   8 IF(IREPET-3)10,3,10
   9 CALL BOUT
  10 RETURN
     END
```

```
C    BINPUT      ***BLENDING INPUT ROUTINE***
*    LIST8
*    LABEL
CBINPUT
     SUBROUTINE BINPUT
     COMMON NUMBER,ITERNO,ITERNN,MAXIT,MINIT,CV,EP,IREPET,JP,NP,KP,MP,
    1   MM,LP,NN,LL,AA,BA,GA,DA,KIND,KPP,LPP
     COMMON THETA
     COMMON VMU,X,F,Z,DEL,BB,OLDZ,R,AM,A,B,IB,JV,IR,IBB
     DIMENSION VMU(20),X(50),F(50),Z(20,50),DEL(20,20),BB(20),OLDZ(20,50)
    10),R(80,241),AM(40,50),A(40),B(20),IB(160),JV(160),IR(80),IBB(160)
     DIMENSION FMTA(12),FMTB(12),FMTMUX(12)
     READ INPUT TAPE 5,1,KIND,JP,NP,KP,MAXIT,MINIT,CV,EP,KRANDM
   1 FORMAT (6I5,9XE4.0,6XE4.0,6XI1)
     MP=NP+KP+JP
     MM=MP+1
     LP=MP+MP
     NN=MM+MP
     LL=NN+MP
     KPP=NP+KP
     LPP=MP+NP+KP
     IF(JP-50)10,10,21
  10 IF(NP-40)11,11,21
  11 IF(KP-20)12,12,21
  12 IF(MP-80)13,13,21
  13 IF(MAXIT)14,14,15
  14 MAXIT=100
  15 IF(MINIT)16,16,17
  16 MINIT=5
  17 IF(CV)18,18,19
  18 CV=1.0E-02
  19 IF(EP)20,20,9
  20 EP=1.0E-05
   9 IF(KRANDM)3,3,2
   2 CALL BRANDM
     RETURN
   3 READ INPUT TAPE 5,4,FMTA,FMTB,FMTMUX
   4 FORMAT (12A6)
     READ INPUT TAPE 5,FMTA,(A(N),(AM(N,J),J=1,JP),N=1,NP)
     READ INPUT TAPE 5,FMTB,(B(K),K=1,KP)
     READ INPUT TAPE 5,FMTMUX,(VMU(K),K=1,KP),(X(J),J=1,JP)
     READ INPUT TAPE 5,5,(IB(L),L=1,MP)
   5 FORMAT (72I1)
     DO 6 L=1,MP
     K=L+MP
     IF(IB(L))7,7,8
   7 IB(L)=K
     IB(K)=L
     GO TO 6
   8 IB(K)=K
     IB(L)=L
   6 CONTINUE

     CALL BINOUT(0)

     RETURN
```

```
*    LIST8
*    LABEL
CBINOUT
     SUBROUTINE BINOUT(KEY)
     COMMON NUMBER,ITERNO,ITERNN,MAXIT,MINIT,CV,EP,IREPET,JP,NP,KP,MP,
    1   MM,LP,NN,LL,AA,BA,GA,DA,KIND,KPP,LPP
     COMMON THETA
     COMMON VMU,X,F,Z,DEL,BB,OLDZ,R,AM,A,B,IB,JV,IR,IBB
     DIMENSION VMU(20),X(50),F(50),Z(20,50),DEL(20,20),BB(20),OLDZ(20,50)
    10),R(80,241),AM(40,50),A(40),B(20),IB(160),JV(160),IR(80),IBB(160)

C    BINOUT      ***BLENDING OUTPUT ROUTINE FOR INPUT DATA***

     WRITE OUTPUT TAPE 6,1,NUMBER,JP,NP,KP
   1 FORMAT (1H1/////48X20HBLENDING PROBLEM NO. I3////55X10HINPUT DATA/
    1// 45X20HNUMBER OF ACTIVITIES ,6X1H=I3/45X23HNUMBER OF FIXED FACTORS
    2RS ,3X1H=I3/45X27HNUMBER OF VARIABLE FACTORS=I3)
     WRITE OUTPUT TAPE 6,2,NP,JP
   2 FORMAT (///43X30HFIXED-FACTOR CONSTRAINTS, N=1, I3//2X1HN,49X12HA(
    1N,J), J=1, I3,45X4HA(N))
     DO 3 N=1,NP
   3 WRITE OUTPUT TAPE 6,4,N,A(N),(AM(N,J),J=1,JP)
   4 FORMAT (/1XI2,97XF20.6/(6X5F20.6))
     WRITE OUTPUT TAPE 6,5,KP,(B(K),K=1,KP)
   5 FORMAT (///43X30HVARIABLE-FACTOR SUPPLIES, K=1, I3//(6X5F20.6))

     CALL BDATA(KEY)

     WRITE OUTPUT TAPE 6,6
   6 FORMAT (/////50X19HINITIALIZATION DATA//)
     WRITE OUTPUT TAPE 6,7,KP,(VMU(K),K=1,KP)
   7 FORMAT (45X36HINITIAL VARIABLE-FACTOR PRICES, K=1, I3/(6X5F20.6))
     WRITE OUTPUT TAPE 6,8,JP,(X(J),J=1,JP)
   8 FORMAT (//44X29HINITIAL ACTIVITY LEVELS, J=1, I3/(6X5F20.6))
     WRITE OUTPUT TAPE 6,10,MP,(IB(K),K=1,MP)
  10 FORMAT (//49X19HINITIAL BASIS, L=1, I3/(20X20I4))

     DO 9 L=1,LP
   9 IBB(L)=IB(L)
     RETURN
     END
```

```
*    LIST8
*    SYMBOL TABLE
*    LABEL
CBFILLT
     SUBROUTINE BFILLT
C    BFILLT          ***ROUTINE TO FILL THE TABLEAU***
     COMMON NUMBER,ITERNO,ITERNN,MAXIT,MINIT,CV,EP,IREPET,JP,NP,KP,MP,
    1  MM,LP,NN,LL,AA,BA,GA,DA,KIND,KPP,LPP
     COMMON THETA
     COMMON VMU,X,F,Z,DEL,BB,OLDZ,R,AM,A,B,IB,JV,IR,IBB
     DIMENSION VMU(20),X(50),F(50),Z(20,50),DEL(20,20),BB(20),OLDZ(20,50)
    10),R(80,241),AM(40,50),A(40),B(20),IB(160),JV(160),IR(80),IBB(160)
     IF(IREPET-7)3,1,3
   1 DO 2 L=1,LP
   2 IB(L)=IBB(L)
     GO TO 14
   3 CALL BDATA(1)
     DO 4 J=1,JP
     DO 5 K=1,KP
   5 OLDZ(K,J)=Z(K,J)
   4 CONTINUE
  14 DO 8 L=1,LL
     DO 9 K=1,MP
   9 R(K,L)=0.0
   8 CONTINUE
     DO 10 K=1,MP
     L=MP+K
     J=NN+K
     R(K,L)=1.0
  10 R(K,J)=1.0
     DO 6 K=1,KP
     M=NP+K
     AA=B(K)
     DO 7 L=1,KP
     I=NP+L
     DA=DEL(L,K)
     R(I,M)=DA
   7 AA=AA+DA*VMU(L)
   6 R(M,NN)=AA
     DO 11 J=1,JP
     L=KPP+J
     DO 12 N=1,NP
     AA=AM(N,J)
     R(N,L)=AA
  12 R(L,N)=-AA
     DO 13 K=1,KP
     M=NP+K
     AA=Z(K,J)
     R(M,L)=AA
  13 R(L,M)=-AA
  11 R(L,NN)=-F(J)
     DO 15 N=1,NP
  15 R(N,NN)=A(N)
     DO 16 L=1,LP
  16 JV(L)=L
     RETURN
     END
```

```
*    LIST8
*    LABEL
CBINVER
     SUBROUTINE BINVER
C    BINVER          ***BLEND BASIS INVERSION ROUTINE***
     COMMON NUMBER,ITERNO,ITERNN,MAXIT,MINIT,CV,EP,IREPET,JP,NP,KP,MP,
    1   MM,LP,NN,LL,AA,BA,GA,DA,KIND,KPP,LPP
     COMMON THETA
     COMMON VMU,X,F,Z,DEL,BB,OLDZ,R,AM,A,B,IB,JV,IR,IBB
     DIMENSION VMU(20),X(50),F(50),Z(20,50),DEL(20,20),BB(20),OLDZ(20,50)
    10),R(80,241),AM(40,50),A(40),B(20),IB(160),JV(160),IR(80),IBB(160)
     DIMENSION KEY(80)
     DO 1 L=1,MP
   1 KEY(L)=0
     DO 2 L=1,MP
     K=IB(L)
     J=JV(K)
     IF(K)455,455,456
 455 STOP
 456 IF(J)457,457,458
 457 STOP
 458 CONTINUE
     AA=0.0
     DO 3 I=1,MP
     IF(KEY(I))4,4,3
   4 DA=ABSF(R(I,J))
     IF(DA-AA)3,3,5
   5 AA=DA
     IPR=I
   3 CONTINUE
     IF(AA-EP)6,6,8
   6 WRITE OUTPUT TAPE 6,7,ITERNN,AA,EP,(IB(I),I=1,LP)
   7 FORMAT (/////26X25HSINGULAR BASIS, ITERATION I3,9H,  PIVOT=E12.4,8
    1H,  MIN.=E12.4//57X5HBASIS/(10X20I5))
     IREPET=10
     RETURN
   8 IR(L)=IPR
     KEY(IPR)=1
   2 CALL BPIVOT(IPR,L,L)
     RETURN
     END
```

```
C    BSOLVE          ***SIMPLEX PROCEDURES***
*    LIST8
*    LABEL
CBSOLVE
     SUBROUTINE BSOLVE
     COMMON NUMBER,ITERNO,ITERNN,MAXIT,MINIT,CV,EP,IREPET,JP,NP,KP,MP,
    1   MM,LP,NN,LL,AA,BA,GA,DA,KIND,KPP,LPP
     COMMON THETA
     COMMON VMU,X,F,Z,DEL,BB,OLDZ,R,AM,A,B,IB,JV,IR,IBB
     DIMENSION VMU(20),X(50),F(50),Z(20,50),DEL(20,20),BB(20),OLDZ(20,50)
    10),R(80,241),AM(40,50),A(40),B(20),IB(160),JV(160),IR(80),IBB(160)
     KEY=0
     NUMB=0
     IF(IREPET-7)2,1,2
   1 KEY=1
   2 IREPET=0

  28 AA=0.0
     DO 3 M=1,MP
     K=IR(M)
     DA=R(K,NN)
     IF(DA)4,3,3
   4 J=IB(M)
     IF(J-KPP)6,6,5
   5 IF(J-LPP)34,34,6
   6 IF(DA-AA)7,3,3
   7 AA=DA
     IPR=M
   3 CONTINUE

     IF(AA)9,8,8
   8 KABOB=0
     RETURN

   9 IREPET=1
     LPR=IB(IPR)
     KPR=IR(IPR)
     JPC=IPR+MP

  10 ITERNO=ITERNO+1
     LPC=IB(JPC)
     J=JV(LPC)
     AA=R(KPR,NN)/R(KPR,J)
     GA=0.0
     IPC=0
     DO 11 M=1,MP
     L=IB(M)
     IF(L-KPP)13,13,12
  12 IF(L-LPP)15,15,13
  13 IF(LPR-KPP)11,11,14
  14 IF(LPR-LPP)15,15,11
  15 I=IR(M)
     IF(R(I,J))11,11,16
  16 DA=R(I,NN)/R(I,J)
     IF(DA)17,19,20
  17 WRITE OUTPUT TAPE 6,18,ITERNN,ITERNO,(IB(K),K=1,MP)
  18 FORMAT (/1X29HPRIMAL INFEASIBILITY OCCURRED/1X7HITERNN= I3,
```

```
  1   10H,  ITERNO= I3,9H,  BASIS=/(1X10I3))
    GO TO 11
 19 R(I,NN)=.0000001
    GO TO 16
 20 IF(GA)17,22,21
 21 IF(DA-GA)22,11,11
 22 IPC=M
    GA=DA
 11 CONTINUE

 23 IF(AA)25,25,29
 29 IF(GA)27,27,24
 24 IF(AA-GA)27,27,26
 25 IF(GA)30,30,26

 26 CALL BPIVOT(IPC,JPC,MM)
    JPC=IPC+MP
    GO TO 10

 27 CALL BPIVOT(IPR,JPC,MM)
    GO TO 28
 30 WRITE OUTPUT TAPE 6,31,ITERNN
 31 FORMAT (///1X38HSOLUTION UNBOUNDED IN GROSS ITERATION I3)
    IF(KABOB)32,32,33
 32 KABOB=1
    IREPET=7
    RETURN
 33 IREPET=9
    RETURN

 34 WRITE OUTPUT TAPE 6,41
 41 FORMAT (2H *)
    NUMB=NUMB+1
    IF(NUMB-LL)39,39,35
 35 WRITE OUTPUT TAPE 6,42
 42 FORMAT (3H **)
    IF(KEY)38,38,36
 36 WRITE OUTPUT TAPE 6,37
 37 FORMAT (///50X19HFEASIBLE BASIS LOST)
    IREPET=9
    RETURN

 38 IREPET=7
    RETURN
 39 AA=0.0
    DO 40 K=M,MP
    I=IB(K)
    IF(I-KPP)40,40,43
 43 IF(I-LPP)44,44,40
 44 I=IR(K)
    DA=R(I,NN)
    IF(DA-AA)45,40,40
 45 IPR=K
    AA=DA
 40 CONTINUE
    GO TO 9
    END
```

```
*    LIST8
*    LABEL
CBCONVR
     SUBROUTINE BCONVR
     COMMON NUMBER,ITERNO,ITERNN,MAXIT,MINIT,CV,EP,IREPET,JP,NP,KP,MP,
    1   MM,LP,NN,LL,AA,BA,GA,DA,KIND,KPP,LPP
     COMMON THETA
     COMMON VMU,X,F,Z,DEL,BB,OLDZ,R,AM,A,B,IB,JV,IR,IBB
     DIMENSION VMU(20),X(50),F(50),Z(20,50),DEL(20,20),BB(20),OLDZ(20,50)
    10),R(80,241),AM(40,50),A(40),B(20),IB(160),JV(160),IR(80),IBB(160)

C    BCONVR         ***ROUTINE FOR NEWTONS METHOD***

   1 ITERNN=ITERNN+1
     IF (ITERNN-MAXIT)3,3,2
   2 IREPET=9
     RETURN
   3 IF(ITERNN-MINIT)14,13,13
  13 CALL BDATA(2)
     KEY=0
     DA=0.0
     DO 4 M=1,MP
     AA=0.0
     DO 5 N=1,NP
     L=NN+N
   5 AA=AA+A(N)*R(M,L)
     N=NN+NP
     DO 6 K=1,KP
     L=N+K
   6 AA=AA+BB(K)*R(M,L)
     N=N+KP
     DO 7 J=1,JP
     L=N+J
   7 AA=AA-F(J)*R(M,L)
     GA=ABSF(AA-R(M,NN))
     IF(GA-DA)9,9,8
   8 DA=GA
   9 IF(AA)10,4,4
  10 KEY=1
   4 R(M,NN)=AA

     IF(KEY)12,12,11
  11 IREPET=3
     RETURN
  12 IF(DA-CV)15,15,1
  15 THETA=DA
     IF(IREPET-1)2,16,2
  16 MINIT=MINIT+1
  14 IREPET=1
     RETURN
     END
```

```
*    LIST8
*    LABEL
CBOUT
     SUBROUTINE BOUT
     COMMON NUMBER,ITERNO,ITERNN,MAXIT,MINIT,CV,EP,IREPET,JP,NP,KP,MP,
    1   MM,LP,NN,LL,AA,BA,GA,DA,KIND,KPP,LPP
     COMMON THETA
     COMMON VMU,X,F,Z,DEL,BB,OLDZ,R,AM,A,B,IB,JV,IR,IBB
     DIMENSION VMU(20),X(50),F(50),Z(20,50),DEL(20,20),BB(20),OLDZ(20,50)
    10),R(80,241),AM(40,50),A(40),B(20),IB(160),JV(160),IR(80),IBB(160)

C    BOUT          ***BLENDING OUTPUT ROUTINE***

     WRITE OUTPUT TAPE 6,1,NUMBER
   1 FORMAT (1H1/48X20HBLENDING PROBLEM NO. I3/////
    1   43X34HSOLUTION VALUES OF BASIC VARIABLES ///
    2   51X18HECONOMIC VARIABLES //)
     DO 3 J=1,JP
     L=KPP+J
     DO 4 K=1,MP
     I=IB(K)
     IF(I-L)4,5,4
   4 CONTINUE
     GO TO 3
   5 I=IR(K)
     WRITE OUTPUT TAPE 6,6,J,R(I,NN)
   6 FORMAT (39X17HACTIVITY LEVEL X(I2,2H)=F20.6)
   3 CONTINUE
     DO 7 L=1,LP
     IF(L-MM)40,8,11
  40 IF(L-KPP)11,11,7
   8 WRITE OUTPUT TAPE 6,10
  10 FORMAT (///52X15HSLACK VARIABLES //)
   9 CONTINUE
  11 DO 12 K=1,MP
     I=IB(K)
     IF(I-L)12,13,12
  12 CONTINUE
     GO TO 7
  13 M=IR(K)
     IF(I-NP)16,16,14
  14 IF(I-KPP)18,18,15
  15 IF(I-LPP)20,20,22
  16 WRITE OUTPUT TAPE 6,17,I,R(M,NN)
  17 FORMAT (30X26HFIXED-FACTOR PRICE LAMBDA( I2,2H)=F20.6)
     GO TO 7
  18 I=I-NP
     WRITE OUTPUT TAPE 6,19,I,R(M,NN)
  19 FORMAT (31X25HVARIABLE-FACTOR PRICE MU(I2,2H)=F20.6)
     GO TO 7
  20 I=I-MP
     WRITE OUTPUT TAPE 6,21,I,R(M,NN)
  21 FORMAT (41X15HPRIMAL SLACK S(I2,2H)=F20.6)
     GO TO 7
  22 I=I-LPP
     WRITE OUTPUT TAPE 6,23,I,R(M,NN)
  23 FORMAT (43X13HDUAL SLACK V(I2,2H)=F20.6)
```

```
 7 CONTINUE
   WRITE OUTPUT TAPE 6,78
78 FORMAT (/////37X46HSOLUTION VALUES OF NONBASIC VARIABLES ARE ZERO)
   KEY=0
25 WRITE OUTPUT TAPE 6,24,JP,KP,JP
24 FORMAT (/////34X40HVARIABLE-FACTOR INPUTS BY ACTIVITY, J=1, I3,
  1   7H,  K=1, I3//2X1HK,51X12HZ(K,J), J=1, I3)
   DO 26 K=1,KP
26 WRITE OUTPUT TAPE 6,27,K,(Z(K,J),J=1,JP)
27 FORMAT (/1XI2,3X5F20.6/(6X5F20.6))
   WRITE OUTPUT TAPE 6,28,JP,(F(J),J=1,JP)
28 FORMAT (////43X31HACTIVITY YIELD RATES F(J), J=1, I3//(6X5F20.6))
   AA=0.0
   DO 33 K=1,MP
   J=IB(K)
   IF(J-KPP)33,33,88
88 IF(J-MP)35,35,33
35 M=IR(K)
   J=J-KPP
   AA=AA+F(J)*R(M,NN)
33 CONTINUE
   WRITE OUTPUT TAPE 6,34,AA
34 FORMAT (/////36X27HTOTAL YIELD SUM(X(J)*F(J))= F20.6)
   IF(KEY)29,29,30
29 KEY=1
   CALL BDATA(3)
   WRITE OUTPUT TAPE 6,31
31 FORMAT (/////47X25HVALUES FOR NEXT ITERATION)
   GO TO 25
30 WRITE OUTPUT TAPE 6,36,ITERNO,ITERNN,MAXIT,THETA,CV,MINIT
36 FORMAT (/////48X23HSUMMARY OF COMPUTATIONS//
  1   40X30HNUMBER OF SIMPLEX ITERATIONS = I6/
  2   40X30HNUMBER OF GROSS ITERATIONS   = I6/
  3   45X25HMAX. ALLOWED              = I6/
  4   40X30HCONVERGENCE MEASURE           = 5XE7.1/
  5   45X25HMAX. ALLOWED              = 5XE7.1/
  6   40X30HNUMBER OF COMPLETE INVERSIONS= I6)

   WRITE OUTPUT TAPE 6,32,NUMBER
32 FORMAT (//////////39X38HEND OF OUTPUT FOR BLENDING PROBLEM NO. I3)
   RETURN
   END
```

```
*    BPIVOT          ***PIVOTING ROUTINE***
*    LIST8
*    LABEL
CBPIVOT
     SUBROUTINE BPIVOT(IPR,JPC,MMIN)
     COMMON NUMBER,ITERNO,ITERNN,MAXIT,MINIT,CV,EP,IREPET,JP,NP,KP,MP,
    1   MM,LP,NN,LL,AA,BA,GA,DA,KIND,KPP,LPP
     COMMON THETA
     COMMON VMU,X,F,Z,DEL,BB,OLDZ,R,AM,A,B,IB,JV,IR,IBB
     DIMENSION VMU(20),X(50),F(50),Z(20,50),DEL(20,20),BB(20),OLDZ(20,50)
    10),R(80,241),AM(40,50),A(40),B(20),IB(160),JV(160),IR(80),IBB(160)
     J=IB(JPC)
     J=JV(J)
     IF(MMIN-MM)1,2,2
   1 I=IPR
     GO TO 3
   2 I=IR(IPR)
   3 AA=R(I,J)

     DO 4 K=1,MP
   4 R(K,J)=-R(K,J)/AA
     R(I,J)=1.0/AA
     DO 5 L=MMIN,LL
     IF(L-JPC)6,5,6
   6 IF(L-NN)7,8,8
   7 N=IB(L)
     N=JV(N)
     GO TO 9
   8 N=L
   9 DA=R(I,N)
     IF(DA)10,5,10
  10 DO 11 M=1,MP
  11 R(M,N)=R(M,N)+DA*R(M,J)
     R(I,N)=DA/AA
   5 CONTINUE

     IF(MMIN-MM)12,13,13
  13 I=IB(IPR)
     J=IB(JPC)
     K=JV(I)
     L=JV(J)
     IB(IPR)=J
     IB(JPC)=I
     JV(I)=L
     JV(J)=K
  12 RETURN
     END
```

```
      LIST8
      LABEL
BDATA
      SUBROUTINE BDATA(KEY)
C     BDATA          ***ROUTINE TO CALL FUNCTION DATA***
      COMMON NUMBER,ITERNO,ITERNN,MAXIT,MINIT,CV,EP,IREPET,JP,NP,KP,MP,
     1   MM,LP,NN,LL,AA,BA,GA,DA,KIND,KPP,LPP
      COMMON THETA
      COMMON VMU,X,F,Z,DEL,BB,OLDZ,R,AM,A,B,IB,JV,IR,IBB
      DIMENSION VMU(20),X(50),F(50),Z(20,50),DEL(20,20),BB(20),OLDZ(20,50)
     10),R(80,241),AM(40,50),A(40),B(20),IB(160),JV(160),IR(80),IBB(160)
      IF(KEY-7)20,2,20
   20 CONTINUE
      IF(KEY)2,2,1
    1 DO 3 J=1,JP
    3 X(J)=0.0
      DO 4 K=1,KP
    4 VMU(K)=0.0
      DO 5 M=1,MP
      J=IB(M)
      IF(J-MP)6,6,5
    6 IF(J-KPP)8,8,7
    8 IF(J-NP)5,5,9
    7 I=IR(M)
      J=J-KPP
      X(J)=R(I,NN)
      GO TO 5
    9 I=IR(M)
      K=J-NP
      VMU(K)=R(I,NN)
    5 CONTINUE

    2 CALL DATA(KIND,KEY,KP,JP)
      IF(KEY-2)10,11,10
   11 DO 13 K=1,KP
      AA=B(K)
      DO 14 L=1,KP
   14 AA=AA+VMU(L)*DEL(L,K)
   13 BB(K)=AA
   17 DO 15 J=1,JP
      AA=F(J)
      DO 16 K=1,KP
   16 AA=AA+VMU(K)*(OLDZ(K,J)-Z(K,J))
   15 F(J)=AA
   10 RETURN
      END
```

```
      SUBROUTINE DATA
      LIST8
      LABEL
CDATA
      SUBROUTINE DATA(KIND,KEY,KP,JP)
C     THIS SUBROUTINE DOES NOT NEED COMMON
      IF(KIND-19)2,119,2
    2 CONTINUE
      WRITE OUTPUT TAPE 6,7
    7 FORMAT (///45X30HKIND OF PROBLEM NOT IDENTIFIED )
      CALL EXIT
C
  119 CALL DORFMN(KEY,KP,JP)
      GO TO 999
  999 RETURN
      END
```

```
*    LIST8
*    LABEL
CDORFMN
     SUBROUTINE DORFMN(KEY,KP,JP)
C    DORFMN          ***ROUTINE FOR PAKISTAN PROBLEM***
     COMMON NUMBER,ITERNO,ITERNN,MAXIT,MINIT,CV,EP,IREPET,JP,NP,KP,MP,
    1   MM,LP,NN,LL,AA,BA,GA,DA,KIND,KPP,LPP
     COMMON THETA
     COMMON VMU,X,F,Z,DEL,BB,OLDZ,R,AM,A,B,IB,JV,IR,IBB
     DIMENSION VMU(20),X(50),F(50),Z(20,50),DEL(20,20),BB(20),OLDZ(20,50)
    10),R(80,241),AM(40,50),A(40),B(20),IB(160),JV(160),IR(80),IBB(160)
     DIMENSION FMT(12),P(50),C(50),S(50),YM(50),D(50),T(50),V(50)

     IF(KEY)1,1,2
   1 READ INPUT TAPE 5,4,FMT
   4 FORMAT (12A6)
     READ INPUT TAPE 5,FMT,(P(J),C(J),YM(J),S(J),D(J),J=1,JP)
   8 WRITE OUTPUT TAPE 6,3
   3 FORMAT (/////45X29HPARAMETERS OF YIELD FUNCTIONS //
    1   2X1HJ,11X11HCROP PRICES,9X13HTILLING COSTS,7X11HMAX. YIELDS,
    2   9X15HSALT PARAMETERS ,5X26HPERCENT OF WATER IN KHARIF /)
     WRITE OUTPUT TAPE 6,5,(J,P(J),C(J),YM(J),S(J),D(J),J=1,JP)
   5 FORMAT (OP,1XI2,3X4F20.6,2PF20.6)
     DO 6 J=1,JP
     P(J)=S(J)/(P(J)*YM(J))
   6 V(J)=P(J)*S(J)*(0.405)
     RETURN
   2 IF(KEY-7)9,7,9
   7 DO 10 J=1,JP
     CALL RANDM(XX)
     P(J)=ABSF(XX)
     CALL RANDM(XX)
     C(J)=ABSF(XX)
     CALL RANDM(XX)
     S(J)=ABSF(XX)
     CALL RANDM(XX)
     YM(J)=ABSF(XX)
     CALL RANDM(XX)
  10 D(J)=1.0/(1.0+ABSF(XX) )
     GO TO 8
   9 DO 20 J=1,JP
     AA=D(J)*VMU(1)+(1.0-D(J))*VMU(2)
     BA=S(J)-V(J)*AA
     Z(1,J)=D(J)*BA-6.0
     IF(Z(1,J))30,31,31
  30 Z(1,J)=0.0
  31 Z(2,J)=(1.0-D(J))*BA-3.0
     IF(Z(2,J))32,33,33
  32 Z(2,J)=0.0
  33 DA=AM(1,J)*(Z(1,J)+6.0)+AM(2,J)*(Z(2,J)+3.0)
  20 F(J)=-C(J)+(S(J)/P(J))*(1.0-((1.0-DA/S(J))*(1.0-DA/S(J)))/(0.81))
     IF(KEY-1)60,61,60
  60 RETURN
  61 CONTINUE

     DO 59 J=1,JP
  59 T(J)=X(J)*V(J)
```

```
      AA=0.0
      BA=0.0
      GA=0.0
      DO 62 J=1,JP
      GA=GA-T(J)*AM(1,J)*AM(2,J)*D(J)*(1.0-D(J))
      BA=BA-T(J)*AM(2,J)*(1.0-D(J))*(1.0-D(J))
   62 AA=AA-T(J)*AM(1,J)*D(J)*D(J)
      DEL(1,1)=AA
      DEL(2,2)=BA
      DEL(1,2)=GA
      DEL(2,1)=GA
      RETURN
      END
```

```
*    LIST8
*    LABEL
CDATA
     SUBROUTINE DATA(KIND,KEY,KP,JP)
C    DATA
C         THIS SUBROUTINE DOES NOT NEED COMMON
   2 IF(KIND-1)3,101,3
   3 CONTINUE
     WRITE OUTPUT TAPE 6,7
   7 FORMAT (///45X30HKIND OF PROBLEM NOT IDENTIFIED )
     CALL EXIT

 101 CALL SHLAFR(KEY,KP,JP)
     GO TO 999
 999 RETURN
     END
```

```
  LIST8
  LABEL
*LAFR
  SUBROUTINE SHLAFR(KEY,KP,JP)
  COMMON NUMBER,ITERNO,ITERNN,MAXIT,MINIT,CV,EP,IREPET,JP,NP,KP,MP,
 1  MM,LP,NN,LL,AA,BA,GA,DA,KIND,KPP,LPP
  COMMON THETA
  COMMON VMU,X,F,Z,DEL,BB,OLDZ,R,AM,A,B,IB,JV,IR,IBB
  DIMENSION VMU(20),X(50),F(50),Z(20,50),DEL(20,20),BB(20),OLDZ(20,50)
 10),R(80,241),AM(40,50),A(40),B(20),IB(160),JV(160),IR(80),IBB(160)

  COMMON PI,GAMMA,POWER,SCALE,ORIGIN
  DIMENSION FMT(12),PI(50),GAMMA(50),POWER(10,50),SCALE(10,50),
 1  ORIGIN(10,50)

  SHLAFR        ***BLENDING FUNCTION INPUT FOR SCHLAIFERIAN UTILITIES***

  IF(KEY)12,12,1
 1 IF(KEY-7)18,2,18
 2 DO 4 J=1,JP
  DO 3 K=1,KP
  CALL RANDM(XX)
  IF(XX-0.5)7,6,6
 7 CALL RANDM(XX)
  POWER(K,J)=-ABSF(XX)
  GO TO 11
 6 CALL RANDM(XX)
  POWER(K,J)=XX
11 CALL RANDM(XX)
  SCALE(K,J)=ABSF(XX)
  CALL RANDM(XX)
 3 ORIGIN(K,J)=1.0-1.0/(1.0+ABSF(XX))
  CALL RANDM(XX)
  PI(J)=ABSF(XX)
  CALL RANDM(XX)
 4 GAMMA(J)=ABSF(XX)
  GO TO 14
12 READ INPUT TAPE 5,13,FMT
13 FORMAT (12A6)
  READ INPUT TAPE 5,FMT,(PI(J),GAMMA(J),(POWER(K,J),SCALE(K,J),ORIGIN(K,J)
 1N(K,J),K=1,KP),J=1,JP)
14 WRITE OUTPUT TAPE 6,15,JP
15 FORMAT (/////38X40HPARAMETERS OF YIELD FUNCTIONS F(J), J=1, I3//
 1  2X1HJ,16X5HPI(J),15X8HGAMMA(J),12X10HPOWER(K,J),10X10HSCALE(K,J),10X11H
 2),10X11HORIGIN(K,J) //)
  DO 16 J=1,JP
16 WRITE OUTPUT TAPE 6,17,J,PI(J),GAMMA(J),(POWER(K,J),SCALE(K,J),
 1  ORIGIN(K,J),K=1,KP)
17 FORMAT (1XI2,3X5F20.6/(46X3F20.6))
  DO 19 J=1,JP
  DO 20 K=1,KP
20 ORIGIN(K,J)=ORIGIN(K,J)/SCALE(K,J)
19 CONTINUE
  RETURN

18 DO 99 J=1,JP
99 F(J)=0.0
```

```
      IF(KEY-1)201,200,201
  200 CONTINUE
      DO 100 K=1,KP
      DO 101 L=1,KP
  101 DEL(K,L)=0.0
  100 CONTINUE
  201 CONTINUE
      DO 21 K=1,KP
      AA=0.0
      DO 22 J=1,JP
      BA=1.0
      SA=SCALE(K,J)
      IF(SA)24,23,24
   23 Z(K,J)=0.0
      GO TO 22
   24 PA=POWER(K,J)
   27 IF(PA-1.0)28,29,29
   29 IF(PA-2.5)30,31,31
   28 VA=VMU(K)/(SA*PA)
      IF(PA)25,30,26
   25 VA=-VA
   26 ARG=1.0/(PA-1.0)
      VAA=(VA**ARG)/SA
      Z(K,J)=VAA-ORIGIN(K,J)
      IF(Z(K,J))34,35,35
   34 Z(K,J)=0.0
      BA=0.0
      VAA=ORIGIN(K,J)
   35 IF(KEY-1)202,203,202
  203 AA=AA+X(J)*BA*VAA*ARG/VMU(K)
  202 CONTINUE
      IF(PA)32,33,33
   33 F(J)=F(J)+(SA*VAA)**PA
      GO TO 22
   32 F(J)=F(J)-(SA*VAA)**PA
      GO TO 22
   30 VA=SA/VMU(K)
      Z(K,J)=VA-SA*ORIGIN(K,J)
      IF(Z(K,J))40,41,41
   40 Z(K,J)=0.0
      BA=0.0
      VA=SA*ORIGIN(K,J)
   41 F(J)=F(J)+SA*LOGF(VA)
      IF(KEY-1)22,204,22
  204 CONTINUE
      AA=AA-BA*VA/VMU(K)
      GO TO 22
   31 VA=SA/VMU(K)
      Z(K,J)=LOGF(VA)/SA-ORIGIN(K,J)
      IF(Z(K,J))42,43,43
   42 Z(K,J)=0.0
      BA=0.0
      VA=EXPF(ORIGIN(K,J))
   43 IF(KEY-1)206,205,206
  205 AA=AA-BA*VA
  206 CONTINUE
      F(J)=F(J)-1.0/VA
   22 CONTINUE
```

```
      IF(KEY-1)21,207,21
  207 DEL(K,K)=AA
   21 CONTINUE
      RETURN
      END
```

```
C    BRANDM          ***BLENDING RANDOM DATA GENERATION ROUTINE***
*    LIST8
*    LABEL
CBRANDM
     SUBROUTINE BRANDM
     COMMON NUMBER,ITERNO,ITERNN,MAXIT,MINIT,CV,EP,IREPET,JP,NP,KP,MP,
    1   MM,LP,NN,LL,AA,BA,GA,DA,KIND,KPP,LPP
     COMMON THETA
     COMMON VMU,X,F,Z,DEL,BB,OLDZ,R,AM,A,B,IB,JV,IR,IBB
     DIMENSION VMU(20),X(50),F(50),Z(20,50),DEL(20,20),BB(20),OLDZ(20,50)
    10),R(80,241),AM(40,50),A(40),B(20),IB(160),JV(160),IR(80),IBB(160)
     DO 1 K=1,KP
     CALL RANDM(XX)
     VMU(K)=ABSF(XX)
     CALL RANDM(XX)
   1 B(K)=ABSF(XX)
     DO 2 N=1,NP
     CALL RANDM(XX)
     A(N)=ABSF(XX)
     DO 3 J=1,JP
     CALL RANDM(XX)
   3 AM(N,J)=XX
   2 CONTINUE
     DO 4 J=1,JP
     CALL RANDM(XX)
   4 X(J)=ABSF(XX)
     DO 5 L=1,MP
     K=MP+L
     IB(L)=K
   5 IB(K)=L
     CALL BINOUT(7)
     RETURN
     END
```

Exhibit C-15

Variable-Factor Programming

Sample Output

BLENDING PROBLEM NO. 1

INPUT DATA

NUMBER OF ACTIVITIES = 11
NUMBER OF FIXED FACTORS = 15
NUMBER OF VARIABLE FACTORS= 2

FIXED-FACTOR CONSTRAINTS, N=1, 15

A(N,J), J=1, 11

N						A(N)
1	1.000000	1.000000	1.000000	1.000000	1.000000	850.000000
	0.	0.	1.000000	0.	1.000000	
	0.					
2	1.000000	1.000000	0.	0.	0.	850.000000
	1.000000	1.000000	1.000000	1.000000	0.	
	1.000000					
3	-1.000000	-0.	-0.	-0.	-0.	-105.000000
	-0.	-0.	-0.	-0.	-0.	
	-0.					
4	0.	-1.000000	-0.	-0.	-0.	-65.000000
	-0.	-0.	-0.	-0.	-0.	
	-0.					
5	0.	0.	-1.000000	-0.	-0.	-50.000000
	-0.	-0.	-0.	-0.	-0.	
	-0.					
6	0.	0.	0.	-1.000000	-0.	-10.000000
	-0.	-0.	-0.	-0.	-0.	
	-0.					
7	0.	0.	0.	0.	-1.000000	-10.000000
	-0.	-0.	-0.	-0.	-0.	
	-0.					
8	0.	0.	0.	0.	0.	-320.000000

	-1.000000	-0.	-0.	-0.	-0.
	-0.				
9					-60.000000
	0.	0.	0.	0.	0.
	0.	-1.000000	-0.	-0.	-0.
	-0.				
10					-32.000000
	0.	0.	0.	0.	0.
	0.	0.	-1.000000	-0.	-0.
	-0.				
11					-6.000000
	0.	0.	0.	0.	0.
	0.	0.	0.	-1.000000	-0.
	-0.				
12					-80.000000
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	-1.000000
	-0.				
13					-117.000000
	-0.	-0.	-0.	-0.	-0.
	-0.	-0.	-0.	-0.	-0.
	-1.000000				
14					80.000000
	-0.	-0.	-0.	-0.	-0.
	-0.	-0.	-0.	-0.	1.000000
	-0.				
15					117.000000
	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.
	1.000000				

VARIABLE-FACTOR SUPPLIES, K=1, 2

12400.000000	12590.000000

PARAMETERS OF YIELD FUNCTIONS

	CROP PRICES	TILLING COSTS	MAX. YIELDS	SALT PARAMETERS	PERCENT OF WATER IN KHARIF
1	30.700000	10.000000	7.300000	38.300000	79.499999
2	15.900000	110.000000	31.700000	52.600000	76.299999
3	11.700000	10.000000	13.000000	22.800000	100.000000
4	15.100000	10.000000	13.700000	42.200000	100.000000
5	13.500000	10.000000	8.100000	23.900000	100.000000
6	12.800000	10.000000	12.400000	14.100000	-0.
7	13.200000	10.000000	10.200000	21.800000	-0.
8	22.900000	10.000000	7.000000	13.200000	17.300000
9	10.000000	10.000000	10.100000	14.100000	-0.

10	14.000000	10.000000	8.600000	26.000000	100.000000
11	14.000000	10.000000	8.600000	15.100000	-0.

INITIALIZATION DATA

INITIAL VARIABLE-FACTOR PRICES, K=1, 2

5.000000 5.000000

INITIAL ACTIVITY LEVELS, J=1, 11

99.000000	99.000000	99.000000	99.000000	99.000000
99.000000	99.000000	99.000000	99.000000	99.000000
99.000000				

INITIAL BASIS, L=1, 28

29 30 3 4 5 6 7 8 9 10 11 12 13 42 43 44 45 18 19 20
21 22 23 24 25 26 27 28

BLENDING PROBLEM NO. 1

SOLUTION VALUES OF BASIC VARIABLES

ECONOMIC VARIABLES

ACTIVITY LEVEL X(1)=	105.000000
ACTIVITY LEVEL X(2)=	210.000000
ACTIVITY LEVEL X(3)=	403.000000
ACTIVITY LEVEL X(4)=	10.000000
ACTIVITY LEVEL X(5)=	10.000000
ACTIVITY LEVEL X(6)=	320.000000
ACTIVITY LEVEL X(7)=	60.000000
ACTIVITY LEVEL X(8)=	32.000000
ACTIVITY LEVEL X(9)=	6.000000
ACTIVITY LEVEL X(10)=	80.000000
ACTIVITY LEVEL X(11)=	117.000000
FIXED-FACTOR PRICE LAMBDA(1)=	75.170653
FIXED-FACTOR PRICE LAMBDA(2)=	163.718975
FIXED-FACTOR PRICE LAMBDA(3)=	126.456400
FIXED-FACTOR PRICE LAMBDA(6)=	16.160088
FIXED-FACTOR PRICE LAMBDA(7)=	39.047183
FIXED-FACTOR PRICE LAMBDA(8)=	14.998978
FIXED-FACTOR PRICE LAMBDA(9)=	39.078978
FIXED-FACTOR PRICE LAMBDA(10)=	101.719509
FIXED-FACTOR PRICE LAMBDA(11)=	72.718975
FIXED-FACTOR PRICE LAMBDA(12)=	36.548780
FIXED-FACTOR PRICE LAMBDA(13)=	53.318976
VARIABLE-FACTOR PRICE MU(1)=	5.023493

SLACK VARIABLES

PRIMAL SLACK S(4)=	145.000000
PRIMAL SLACK S(5)=	353.000000
PRIMAL SLACK S(14)=	0.
PRIMAL SLACK S(15)=	0.
PRIMAL SLACK S(17)=	4339.171265

SOLUTION VALUES OF NONBASIC VARIABLES ARE ZERO

VARIABLE-FACTOR INPUTS BY ACTIVITY, J=1, 11, K=1, 2

K	Z(K,J), J=1, 11				
1	16.032002	27.632128	9.846534	18.685883	7.272335

	0.	0.	0.	0.	8.576977
	0.				
2	2.681208	7.446677	0.	0.	0.
	11.100000	18.800000	7.600007	11.100000	0.
	12.100000				

ACTIVITY YIELD RATES F(J), J=1, 11

192.969885	377.699436	124.634651	152.878973	72.655995
148.719997	124.639997	137.170134	91.000000	81.708258
110.399999				

TOTAL YIELD SUM(X(J)*F(J))= 231519.529297

VALUES FOR NEXT ITERATION

VARIABLE-FACTOR INPUTS BY ACTIVITY, J=1, 11, K=1, 2

K	Z(K,J), J=1, 11				
1	16.032002	27.632128	9.846534	18.685884	7.272335
	0.	0.	0.	0.	8.576977
	0.				
2	2.681208	7.446677	0.	0.	0.
	11.100000	18.800000	7.600007	11.100000	0.
	12.100000				

ACTIVITY YIELD RATES F(J), J=1, 11

192.969885	377.699436	124.634653	152.878975	72.655997
148.719997	124.639997	137.170134	91.000000	81.708260
110.399999				

TOTAL YIELD SUM(X(J)*F(J))= 231519.529297

SUMMARY OF COMPUTATIONS

NUMBER OF SIMPLEX ITERATIONS	=	9
NUMBER OF GROSS ITERATIONS	=	5
MAX. ALLOWED	=	100
CONVERGENCE MEASURE	=	0.6E-03
MAX. ALLOWED	=	1.0E-02
NUMBER OF COMPLETE INVERSIONS	=	5

END OF OUTPUT FOR BLENDING PROBLEM NO. 1

Bibliography

Barankin, E.W , and R. Dorfman, On Quadratic Programming, University of California Publications in Statistics, Vol. 2 (1958), pp. 285-317.

Beale, E.M.L., "On Optimizing a Convex Function Subject to Linear Inequalities," Jour. Royal Stat. Soc., Vol. 17 (1955), pp. 173-184.

_____, "On Quadratic Programming," Naval Res. Log. Quar., Vol. 6 (1959), pp. 227-243.

Birkhoff, G., and S. MacLane, A Survey of Modern Algebra, Revised Edition, New York: Macmillan Co., 1953.

Charnes, A., and W.W. Cooper, "Chance-Constrained Programming," Management Science, Vol. 6 (Oct. 1959).

_____, and _____, "Deterministic Equivalents for Optimizing and Satisficing Under Chance Constraints," Operations Research, Vol. 11 (Jan. 1963), pp. 18-39.

_____, _____, and G. L. Thompson, "Constrained Generalized Medians and Linear Programming under Uncertainty," Management Science (forthcoming).

Dantzig, George B., "Linear Programming under Uncertainty," Management Science, Vol. 1 (1955), pp. 197-206.

_____, "Maximization of a Linear Function of Variables Subject to Linear Inequalities," pp. 339-347 in Koopmans, Activity Analysis.

_____, "Programming of Interdependent Activities: II, Mathematical Model," Econometrica, Vol. 17 (1949), pp. 200-211; also in Koopmans, Activity Analysis.

_____, Alex Orden, and Philip Wolfe, "The Generalized Simplex Method for Minimizing a Linear Form under Linear Inequality Constraints," Pacific Jour. Math., Vol. 5 (1955), pp. 183-195.

Dorfman, Robert, Application of Linear Programming to the Theory of the Firm, Berkeley: University of California Press, 1951.

_____, P.A. Samuelson, and R.M. Solow, Linear Programming and Economic Analysis, New York: McGraw-Hill, 1958.

Dorn, W.S., "Non-linear Programming--A Survey," Management Science, Vol. 9 (Jan. 1963), pp. 171-207.

Dreyfus, S. and M. Freimer, "A New Approach to the Duality Theory of Mathematical Programming," pp. 340-347 in R. Bellman and S. Dreyfus, Applied Dynamic Programming, Princeton, 1962.

Eggleston, A., Convexity, Cambridge, England: Cambridge Univ. Press, 1958.

Fenchel, W., "Convex Cones, Sets, and Functions," Lecture Notes, Princeton Univ., 1953.

Gass, Saul I. Linear Programming: Methods and Applications, New York: McGraw-Hill, 1958.

Hildreth Clifford, "A Quadratic Programming Procedure," Naval Res. Log. Quar., Vol. 4 (1957), pp. 79-85.

Hochstrasser, U., "Numerical Methods for Finding Solutions of Nonlinear Equations," pp. 255-278 in J. Todd (ed.), Survey of Numerical Analysis, New York: McGraw-Hill, 1962.

Houthakker, H.S., "The Capacity Method of Quadratic Programming," Econometrica, Vol. 28 (1960), pp. 62-87.

Karlin, Samuel, Mathematical Methods and Theory in Games, Programming, and Economics, Vol. I, Reading, Mass.: Addison-Wesley, 1959.

Karreman, H.F., "Economic Analysis of the United States Manganese Problem, Part I, A Nonstochastic Model," Naval Res. Log. Quar., Vol. 7 (Sept. 1960), pp. 261-279.

Kelley, James E., Jr., "The Cutting-Plane Method for Solving Convex Programs," SIAM Journal, Vol. 8 (Dec. 1960), pp. 703-712.

Koopmans, Tjalling C. (edit.), Activity Analysis of Production and Allocation, Monograph No. 13 of the Cowles Commission, New York: Wiley, 1951.

Kuhn, Harold W., and Albert W. Tucker, "Non-Linear Programming," Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability (edit. J. Neyman), 1950, pp. 481-492.

Lemke, C.E., "A Method of Solution for Quadratic Programs," Management Science, Vol. 8 (July 1962), pp. 442-453.

Manne, A.S., Scheduling of Petroleum Refinery Operations, Cambridge, Mass.: Harvard Univ. Press, 1956.

Markowitz, H., Portfolio Selection, New York: Wiley, 1959.

_____, "The Optimization of a Quadratic Function Subject to Linear Constraints," Naval Res. Log. Quar., Vol. 3 (1956), pp. 111-133.

Orden, Alex, "Matrix Inversion and Related Topics by Direct Methods," pp. 39-55 in A. Ralston and H.S. Wilf, Mathematical Methods for Digital Computers, New York: Wiley, 1960.

Rosen, J.B., "The Gradient Projection Method for Nonlinear Programming Part I: Linear Constraints," SIAM Journal, Vol. 8 (Mar. 1960), pp. 181-217.

_____, "The Gradient Projection Method for Nonlinear Programming, Part II: Nonlinear Constraints," SIAM Journal, Vol. 9 (Dec. 1961), pp. 514-532.

Samuelson, P., Foundations of Economic Analysis, Cambridge, Mass.: Harvard Univ. Press, 1947, Chapter III, pp. 21-56.

Schlaifer, R., Probability and Statistics for Business Decisions, New York: McGraw-Hill, 1959, Chap. 4, p. 74 ff.

Theil, Henri, and C. van de Panne, "Quadratic Programming as an Extension of Classical Quadratic Maximization," Management Science, Vol. 7 (Oct. 1960), pp. 1-20.

Vajda, S., Mathematical Programming, Reading, Mass.: Addison-Wesley, 1961.

Wilde, D., "Differential Calculus in Nonlinear Programming," Operations Research, Vol. 10 (Nov. 1962), pp. 764-773.

Wolfe, Philip, "The Simplex Method for Quadratic Programming," Econometrica, Vol. 27 (1959), pp. 382-398.

_____, "Recent Developments in Non-linear Programming," A Report Prepared for United States Air Force Project RAND, RAND Corporation, 1962. (R-401-PR)

Zoutendijk, G., Methods of Feasible Directions, Amsterdam: Elsevier and van Nostrand (1960).