SEPARABLE PROGRAMMING ANALYSIS OF SPATIAL COMPETITIVE MARKET MODELS

By

SHIN AN CHIANG

Bachelor of Science in Engineering
National Cheng Kung University
Tainan, Taiwan
1973

Master of Science
University of Wisconsin
Madison, Wisconsin
1980

Submitted to the Faculty of the Graduate College
of the Oklahoma State University
in partial fulfillment of the requirements
for the Degree of
MASTER OF SCIENCE
December, 1986

SEPARABLE PROGRAMMING ANALYSIS OF SPATIAL

COMPETITIVE MARKET MODELS

Thesis Approved:

Donald W. Grace
Thesis Adviser

J P Chandler

Keith A. Teague

Norman N. Durham
Dean of the Graduate College

PREFACE

The objective of this study is to analyze spatial competitive market equilibrium models by separable programming. Separable programming is an application of grid linearization techniques for approximating nonlinear separable functions with linear segments. This paper uses a grid refinement program to generate different grid sizes, a matrix generator to convert MINOS input format to MPSX input format, a translator program to transform MPSX output standard format to a readable format, an inverse program to convert results of linear programs into the variables of the quadratic programs, an MPSX program to execute the MPSX package, a MINOS program to execute the MINOS package, and the MINIT program to execute linear programming problems.

I would like to express sincere gratitude to my major advisor Dr. Donald W. Grace for his guidance, motivation, and invaluable help. I am also thankful to Dr. Keith D. Willett, Dr. John P. Chandler, and Dr. Ramesh Sharda for their insightful suggestions during the course of this work.

My deepest gratitude to my parents for their encouragement and for their love.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Mathematical programming specifications of spatial competitive market equilibrium problems have appeared extensively throughout the economics literature. The basic structural foundations for these models were first provided by Samuelson [1]. Samuelson's original specification was for a single commodity with multiple regions. Takayama and Judge [2] extended Samuelson's work to multi-market equilibria using quadratic programming and have become the standard reference for such extensions. Furtan et al. [3] have utilized this conceptual model and applied quadratic programming to problems of international trade in Canadian agriculture.

A major concern in the use of mathematical programming specifications for spatial competitive market equilibrium models is generating numerical solutions. As noted previously, the Takayama and Judge models were based on a quadratic programming specification. Polito et al. [4] have pointed out that, in actual applications, relatively small quadratic programming problems have been solved. These authors have also noted that an extreme inefficiency may be achieved by always relying on quadratic programming, i.e., the algorithm fails to solve the problem or the wrong answer is given. This, in turn, has motivated the development of approximations or alternative solution procedures. Duloy and Norton [5], for example, have shown how a quadratic objective function can be approximated as a linear objective function with the use of separable programming. This approach has the advantage of allowing use of the simplex method for routine numerical solution, thereby expanding the size and scope of problems which can be considered.

The purpose of Dr. Willett's work [6] is to present a single commodity spatial equilibrium model stated as a linear programming problem. The linearization techniques employed by Duloy and Norton were used to develop the linear programming model. This is the technique to approximate nonlinear separable functions with linear segments. Separable functions are functions that can be expressed as sums of expressions of a single variable. The optimizing spatial competitive market equilibrium formulation is based on the assumption that producers are profit maximizers and that consumers' behavior is adequately described by a set of aggregate demand functions in the space of prices and quantities. Supply functions are represented in this model through producers' technology and behavior specifications, including resource limitations, and the objective function. The perceived contribution of this thesis is the implementation of Willett's methodology which allows models of spatial competitive market equilibria to be solved as standard linear programming problems.

CHAPTER II

BACKGROUND AND LITERATURE REVIEW

Separable Programming

Separable programming is a mathematical programming technique that solves a linear programming problem constructed to be a good approximation of a nonlinear problem. The data for the linear problem result from the evaluation of the objective and constraint functions of the nonlinear problem on a grid of points spanning a suitable portion of the space of the problem, and substituting a piecewise linear function for each nonlinear function.

Let x_1, x_2, . . . x_S be a collection of n-vectors. Any point x of the convex hull of this collection may be written

$$x = \sum_{s=1}^{S} D_s\, x_s \qquad \text{(Eq. 1)}$$

Where

$$\sum_s D_s = 1 \text{ and } D_s \geq 0 \text{ for all } s \qquad \text{(Eq. 2)}$$

Given any function g of x, the linearization of g on the grid x_1, . . . x_S is attained through the approximation by using the same D_s as in (Eq. 1).

$$g(x) = \sum_s D_s g(x_s) \qquad \text{(Eq. 3)}$$

Any mathematical programming problem becomes a linear problem in the nonnegative variables D_s if x, g(x), and f(x) are replaced throughout by their representations above. Using this representation, the mathematical programming problem may be stated in the approximate form:

Minimize $\sum_s D_s F(x_s)$ (Eq. 4)

subject to the constraints

$$\sum_s D_s = 1 \quad \text{(Eq. 5)}$$

$$\sum_s D_s g_i(x_s) \geq 0 \quad \text{(Eq. 6)}$$

for all i

The observations above make grid linearization an effective tool for problems having the proper convexity; but where convexity does not obtain, a more refined technique must be used [7].

Limitation of the Method

This method cannot be called a general-purpose nonlinear programming procedure, because it solves nonlinear programming problems with the following important constraints:

1. Each nonlinear function must be a function of only one variable or a linear combination of such functions, that is, "separable". However, in many cases nonseparable functions can be converted to separable forms by using appropriate transformations. The appropriate transformations depend on the particular functional forms. Hadley [8] discussed several possible transformations including transformation to logs and the definition of new variables (For example, Xe^Y can be transformed to natural logarithm expression LnX + Y).

2. Each function must be polygonal, or replaceable by a polygonal approximation to it. In other words, it must be able to be described by a piecewise linear function. This approximation automatically increases the number of variables and thus incurs a substantial computational burden.

3. Separable programming does not necessarily lead to the global optimum and furthermore gives no indication of how far the separable programming solution might be from the global optimum [9].

Despite these disadvantages, separable programming has been used for a number of practical problems [10], and computer programs are available for it [11].

Linear Programming

Linear programming (LP) is an optimization method applicable for the solution of problems in which the objective function and the constraints appear as linear functions of the decision variables. The constraint equations in a linear programming problem may be in the form of equalities or inequalities. The linear programming type of optimization problem was first recognized in the 1930s by economists while developing methods for the optimal allocation of resources. During World War II the United States Air Force sought more effective procedures of allocating resources and turned to linear programming. George B. Dantzig, who was a member of the Air Force group, formulated the general linear programming problem and devised the simplex method of solution in 1947. This was a significant step in bringing linear programming into wider usage. Afterwards, much progress was made in the theoretical development and in the practical applications of linear programming. The theoretical contributions made by Kuhn and Tucker had a major impact in the development of the duality theory in LP. the work of Charnes and Cooper was directed toward the industrial applications of LP. In the food processing industry, linear programming has been used to determine the optimal shipping plan for the distribution of a particular product from the different manufacturing plants to the various warehouses. The optimal routing of messages in a communication network and the routing of aircraft and ships can also be decided by using linear programming [12].

The general linear programming problem can be stated in the following standard form:

Minimize $F(x) = C^T X$ (Eq. 7)

subject to the constraints

$$A X \geq B \qquad \text{(Eq. 8)}$$

$$X \geq 0 \qquad \text{(Eq. 9)}$$

Where

$$X = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_n \end{bmatrix} \qquad B = \begin{bmatrix} b_1 \\ b_2 \\ . \\ b_m \end{bmatrix} \qquad C = \begin{bmatrix} C_1 \\ C_2 \\ . \\ . \\ C_n \end{bmatrix}$$

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots\cdots & a_{1n} \\ a_{21} & a_{22} & \cdots\cdots & a_{2n} \\ . & . & \cdots\cdots & . \\ a_{m1} & a_{m2} & \cdots\cdots & a_{mn} \end{bmatrix}$$

The case n = m is of no interest, for then there is either a unique solution X which satisfies Eqs. (8) and (9) (in which case there can be no optimization) or no solution, in which case the constraints are inconsistent. The case m < n corresponds to an underdetermined set of linear equations which, if they have one solution, have an infinite number of solutions. The problem of linear programming is to find one of these solutions satisfying Eqs. (8) and (9) and yielding the minimum of objective function.

Quadratic Programming

A quadratic programming (QP) problem is the most well-behaved nonlinear programming problem. In this problem, the objective function is assumed convex (to assure global minimum) and all the constraints are linear. Hence quadratic programming problems can be solved by suitably modifying the simplex method of linear programming. In some practical optimization problems, the objective and constraint functions are separable in the design variables. Separable programming techniques are useful for solving such problems.

A quadratic programming problem can be stated as:

Minimize $$f(X) = c^T X + 1/2\ x^T D \cdot X \qquad \text{(Eq. 10)}$$

subject to the constraints

$$A\ X \geq B \qquad \text{(Eq. 11)}$$

$$X \geq 0 \qquad \text{(Eq. 12)}$$

Where

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ x_n \end{bmatrix} \qquad B = \begin{bmatrix} b_1 \\ b_2 \\ \cdot \\ b_m \end{bmatrix} \qquad C = \begin{bmatrix} c_1 \\ c_2 \\ \cdot \\ \cdot \\ c_n \end{bmatrix}$$

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdot & \cdot & \cdots & \cdot \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \quad \text{and } D = \begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ \cdot & \cdot & \cdots & \cdot \\ d_{n1} & d_{n2} & \cdots & d_{nn} \end{bmatrix}$$

In Eq. (10), the term $X^T D X/2$ represents the quadratic part of the objective function with D being assumed a symmetric positive definite matrix. If D = 0, the problem reduces to a LP problem. The solution of the quadratic programming problem stated in Eqs. (10) to (12) can be obtained by using the Lagrange multiplier technique. Details are in the reference [13].

Availability of Quadratic Programming Software and Approximations

Quadratic Programming is both a special case of nonlinear programming and an extended case of linear programming. Consequently, software from both areas has been adapted for quadratic programs. The original approach to quadratic programming was by Wolfe, using the Kuhn-Tucker conditions. The Kuhn-Tucker conditions form a large linear program, with additional complementary slackness conditions. Wolfe then utilized a variant of the simplex algorithm which incorporated provisions to enforce the complementary slackness conditions. Many available algorithms follow these principles.

In the early 1960s, Cottle and Dantzig, and Lemke developed the complementary pivoting theory for solution of quadratic problems. This approach solves problems via a process which allows only one of a pair of variables in any basis [14].

The third algorithmic approach for quadratic programming is based on nonlinear gradients. This theory was presented in an article by Murtagh and Saunders [15]. Later, this work culminated in the Modular In-core Nonlinear Optimization System (MINOS) package [16].

Fourth, there is the decomposition procedure based on Bender's work [17].

Lemke's complementary pivoting algorithm [18] and MINOS package are currently available at Oklahoma State University. But according to author's experience, when running test problem 1, the complementary pivoting algorithm cannot find the feasible solution. The infeasible solution is given in Appendix F. Therefore, until this difficulty can be resolved, only nonlinear gradient theory is considered here in the comparison with separable programming method.

CHAPTER III

METHODOLOGY

Applicability of Mathematical Programming Models to Spatial Competitive Market Analysis

The spatial competitive market equilibrium which is to be modeled can be summarized in the following way. Two or more regions with known demand functions and production functions produce and consume a homogeneous product. Since goods can be shipped back and forth between regions, therefore, the regions are separated but can communicate for a price (transfer costs). Given this information, the problem is to determine the equilibrium levels of production, consumption, and prices in each region and equilibrium trade flows between regions.

An optimal solution to the problem described above is characterized by three equilibrium conditions. First, prices will differ between any two regions by an amount that is less than or equal to the transfer costs. For the second condition, assume that the quantity of a good which is produced and consumed in the same region is viewed as a transfer flow to the region itself. Then demand in each region equals the trade flows to that region. Finally, there is an implied condition that the equilibrium price and quantity must lie on the implicit supply function and the demand function.

The basic components of the spatial competitive market can match those of mathematical programming models. Mathematical programming models have three basic components: an objective function to be optimized; a set of alternative activities or processes which can be used for attaining the objective; and resource or other restrictions on the solution. The objective function of this model is to maximize the sum of consumer and producer surplus within a competitive market system. Activities available for attaining the objective include production and distribution of the various commmodities. Finally, limits on resources available, and institutional restrictions provide constraints on the system.

The mathematical programming model that provides a competitive market equilibrium solution to this spatial problem is driven by an objective function which Samuelson called the "net social payoff". This objective function is defined as the sum of consumers' plus producers' surplus less the total transportation cost for all possible trade flows. Assume that a single commodity is produced and consumed in each region. Also assume that the ith region has a known inverse demand function with demand price as the dependent variable:

$$p_i^D = a_i - b_i q_i \qquad \text{(Eq. 13)}$$

Where p_i^D = the demand prices in region i

q_i = quantity demand in region i

a_i = demand intercepts in region i

b_i = demand slopes in region i

The objective function (expressed in dollars) is formed by subtracting explicit production costs and the cost of shipping commodities between regions from the area under the demand curve. Let c_i (dollar/unit commodity) denote the explicit cost for purchased inputs, Y_i represent the amount of the commodity produced in region i, and let t_{ij}

(dollar/unit commodity) denote the unit cost of shipping the commodity from region i to region j. Also let X_{ij} represent the amount of the commodity from region i to region j. Then the objective function is written as:

$$\sum_i (a_i - 1/2\ b_i q_i) q_i - \sum_i c_i Y_i - \sum_{ij}\sum t_{ij} X_{ij} \qquad \text{(Eq. 14)}$$

The search for optimal demands, production levels, and prices is bounded by several constraints. For each region, the quantity of the commodity demanded is less than or equal to the quantity supplied by that region plus the quantity shipped from other regions. This constraint is written as:

$$q_j \leq \sum_i X_{ij} \qquad \text{(Eq. 15)}$$

for all j

For each region, total shipments is less than or equal to total production. This constraint is written as:

$$\sum_j X_{ij} \leq Y_i \qquad \text{(Eq. 16)}$$

for all i

There are also resources in each region, such as land and certain types of labor, whose availability is constrained. This, in turn, means that an additional constraint must be imposed on the production possibilities set for each region. Let d_{ri} represent the amount of resource r necessary to produce one unit of the commodity in region i and let B_{ri} denote the maximum amount of the rth resource available in region i. Then the resource availability constraint in the ith region can be written as:

$$d_{ri} Y_i \leq B_{ri} \qquad \text{(Eq. 17)}$$

for all r and i.

The constraints (Eqs. 15-17) can be combined with the objective function (Eq. 14) to form the single commodity spatial competitive equilibrium model. This model is written as follows:

$$\max \sum_i (a_i - 1/2\ b_i q_i) q_i - \sum_i c_i Y_i - \sum_{ij}\sum t_{ij} X_{ij} \qquad \text{(Eq. 18)}$$

Subject to

$$q_j \leq \sum_i X_{ij} \qquad \text{(Eq. 19)}$$

for all j

$$\sum_j X_{ij} \leq Y_i \qquad \text{(Eq. 20)}$$

for all i

$$d_{ri}\ Y_i \leq B_{ri} \qquad \text{(Eq. 21)}$$

for all r and i.

Development of the Separable Programming Model

The mathematical model used in this study is formulated within a general linear programming framework. The advantages of linear programming arise from the fact that the simplex algorithm is a very powerful solution technique. It allows a greater amount of detail in the specification of regional factor supplies and production processes without making the model prohibitively large or expensive. If the results of interregional analyses are to be of use to the policy makers, considerable regional detail is needed.

A major limitation of the quadratic programming formulation is that the solution algorithms are much more expensive than the simplex algorithm for equivalent-sized problems. The modeler is thus faced with the tradeoff of greatly increased solution costs or of giving up some detail in the specification of regional resources and production activities.

The terms in the objective function representing the area under the demand function must be linearized before setting up the linear programming model, Following Duloy and Norton, this is done by grid linearization which requires prior specification of the relevant range of values on the demand curve and the use of variable interpolation weights on the grid points. The interpolation weights become special variables in the model and their values are jointly constrained by a set of convex combination constraints. The principal advantage of this technique is that the demand functions can be approximated as closely as required without requiring additional constraints in the model other than the convex constraints.

First, a function representing the area under the demand curve in the ith region is defined as follows:

$$A = (a_i - 1/2\ b_i q_i) q_i \qquad \text{(Eq. 22)}$$

For each region, the initial demand curve, defined in its own price-quantity space, must pass through the point $(\bar{p}_i^D , \bar{q}_i)$ as illustrated in Figure 1. The relevant range of the demand curve is defined and truncated at point a and b. Next, the relevant range of the demand curve is partitioned into segments s=1, . . .,S. For each segment, the area under the demand curve is written as:

$$A_{is} = (a_i - 1/2\ b_i q_{is}) q_{is} \qquad \text{(Eq. 23)}$$

For each segment endpoint, the parameters q_{is} and A_{is} represent the cumulative quantity demanded and the cumulative area under the demand function in the ith region, respectively. The quantity demanded and the value of the area under the demand curve for the good in the ith region can be expressed as a weighted combination of the q_{is} and A_{is} respectively:

$$q_i = \sum_s q_{is}\ D_{is} \qquad \text{(Eq. 24)}$$

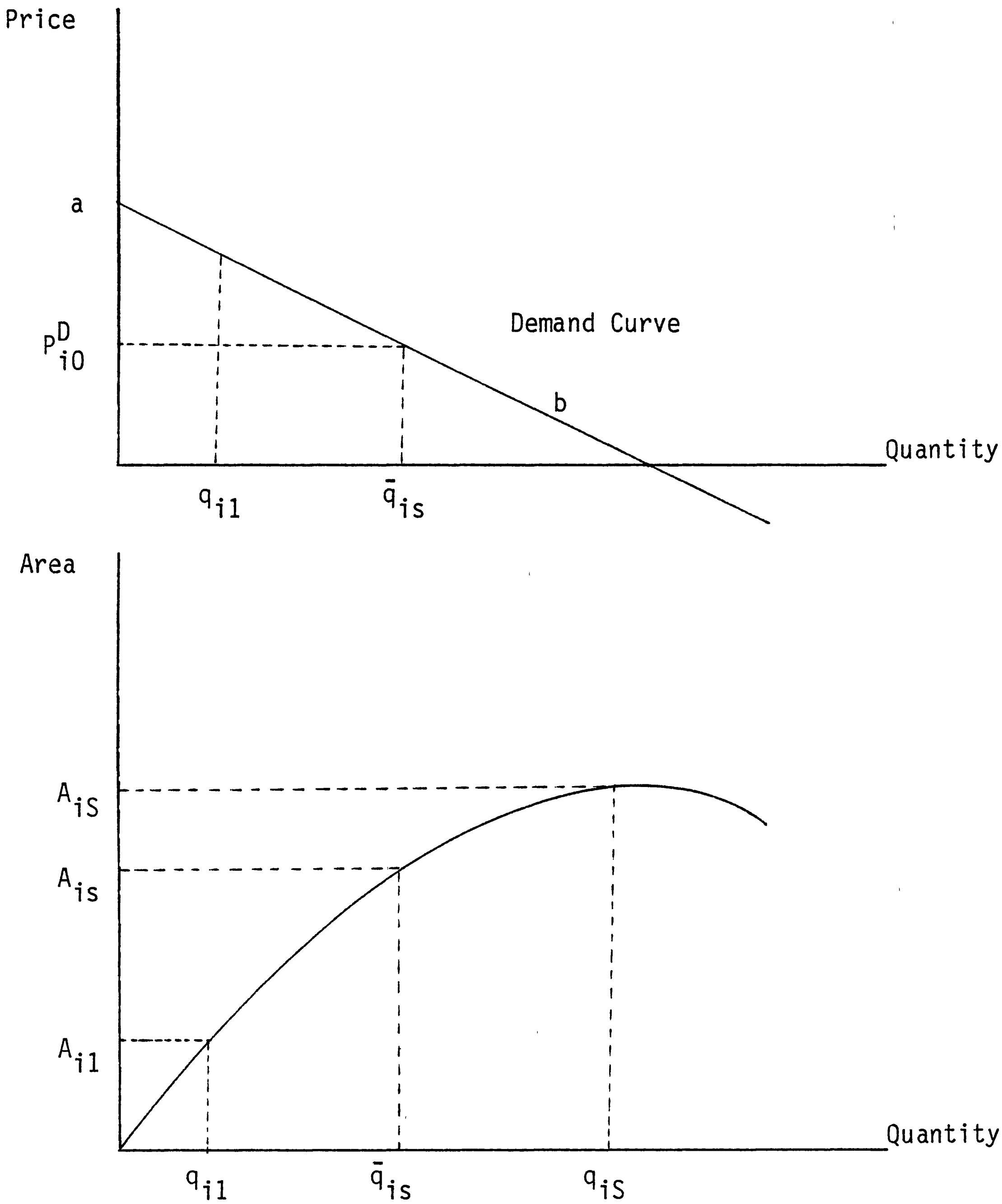

Figure 1. Grid Linearization of the Demand Curve

$$A_i = \sum_s A_{is} D_{is} \qquad \text{(Eq. 25)}$$

The D_{is} are special variables and are defined such that

$$\sum_s D_{is} = 1 \qquad \text{(Eq. 26)}$$

With all of the necessary steps completed, the linear programming model which will yield a spatial competitive market equilibrium can be written as follows:

$$\text{Maximize} \quad \sum_i \sum_s A_{is} D_{is} - \sum_i c_i Y_i - \sum_i \sum_j t_{ij} X_{ij} \qquad \text{(Eq. 27)}$$

subject to the constraints

$$\sum_s q_{js} D_{js} \leq \sum_i X_{ij} \qquad \text{(Eq. 28)}$$

for all j

$$\sum_j X_{ij} \leq Y_i \qquad \text{(Eq. 29)}$$

for all i

$$d_{ri} Y_i \leq B_{ri} \qquad \text{(Eq. 30)}$$

for all r and i.

$$\sum_s D_{is} = 1 \qquad \text{(Eq. 31)}$$

for all i

The Conceptual Model of the Program

There are seven programs involved in this thesis: (a) the grid refinement program, which calculates the cumulative area and the cumulative quantity demanded under the demand function in the i region; (b) the matrix generator program, which converts MINOS input format to the Mathematical Programming System Extended (MPSX) input format; (c) the translator program, which translates MPSX output Standard format to a readable format; (d) the inverse program, which compares the accuracy of the quadratic part of the objective function between LP and QP systems; (e) the MINOS program, which executes the MINOS package; (f) the MPSX program, which executes MPSX package; and (g) the MINIT program, which solves the linear programming problems. A schematic of these programs and datasets is given in Figure 2.

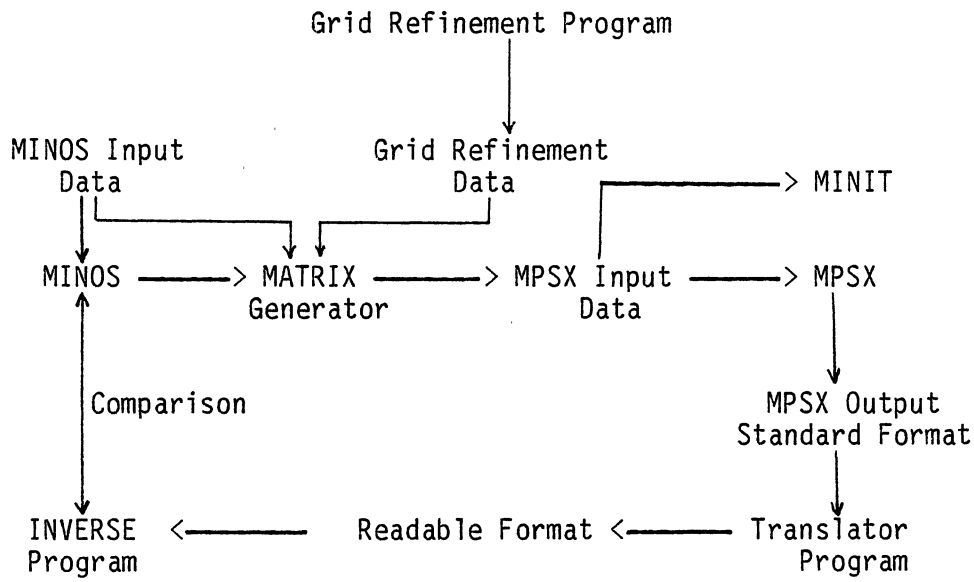

Figure 2. Relationship of Programs and Datasets

Grid Refinement Program

This program is a starter program for the MPSX package; the user needs to specify the number of intervals and the basic point value (Pb) for the grid linearization method. The grid size is calculated by the formula:

$$K = (q_e - q_s) / N \qquad \text{(Eq. 32)}$$

Where q_e: the end node of the interval

$(1.45 \text{ pb} \leq q_e \leq 2.0 \text{ Pb})$

q_s: the starting node of the interval

$(0.25 \text{ pb} \leq q_s \leq 0.36 \text{ Pb})$

N : number of intervals

K : grid size

Note that N intervals generate N+1 points q_k, where q_k are equally spaced, and $q_1 \leq q_2 \leq \ldots \leq q_n = q_e$ in the interval $q_s \leq q_k \leq q_e$. The q_k is calculated by the formula:

$$q_k = q_{k-1} + K \qquad \text{(Eq. 33)}$$

The Grid refinement program is given in Appendix A.

Matrix Generator Program

The matrix generator which starts with reading MINOS input data in the MINOS format, and grid refinement dataset and automatically builds a mathematical programming model in a format acceptable to the input procedures of MPSX package. This program can be used to modify the existing MINOS input format to the MPSX input format. The matrix generator program is given in Appendix B.

Translator Program

This program is designed to convert MPSX output standard format to a readable format [19]. A standard format is composed of sections corresponding to various sections of the printed output. The translator program is given in Appendix C.

Inverse Program

The inverse program converts results of linear programs into the variables of the original quadratic programs. These results will be substituted into equation (24) and compared with the outputs from running the quadratic programs directly.

MINOS Program

This program reads MINOS input format and executes the MINOS package. The MINOS package is a Stanford University product designed to solve large-scale optimization problems.

MPSX Program

This program reads MPSX input format and executes the MPSX package. The MPSX package is an IBM program product intended for the study of linear programming applications.

MINIT Program

This program reads the generated MPSX input data and executes the linear programming problems. The MINIT algorithm was presented as algorithm 333 in the Communications of the ACM [20].

All of the seven programs are programmed in FORTRAN on an IBM 3081k mainframe. The translator program is delivered to the users in load module form. The MINOS package and MPSX package, are also written in standard FORTRAN. However, source code for MPSX is not available and MINOS cannot legally be exported to some countries.

The comparisons of these approaches will be described in Chapter IV.

CHAPTER IV

A COMPARISON OF FEATURES

General

The two packages and MINIT program compared in this thesis are listed in Table I. All of them offer linear programming; the MINOS package has the capabilities to solve quadratic programming problems. In this study, the results derived from the MPSX package and MINIT program are compared with the results obtained from the MINOS package.

TABLE I

SUMMARY OF THE PROGRAMS

Code Name	Basic Theory	Maximize Program Size*	User Interface
MINOS	Gradient	L	Available
MPSX	Revised Simplex Method	VL	Available
MINIT	Dual Simplex Method		

*Problem size refers to the number of variables. L (large, 500-3000; VL (very large), over 3000.

Performance Tests

The comparisons presented are based on the program capabilities and demonstration runs. These three programs are compared by attempting to solve two test problems. Tables II and III exhibit major statistics of the test problems 1 and 2 employed in this study; description of test problem 1 is in reference [21], and test problem 2 is in references [22] and [23]. The mathematical statement of test problems 1 and 2 are given in Appendices D and E respectively. These problems are realistic problems in that neither is completely randomly generated. They both include realistic coefficients and structure.

TABLE II

A DESCRIPTION OF TEST PROBLEM 1

Name	Number of* Intervals	Rows	Columns	Density
MINOS		28	42	7.483
MPSX	5	35	66	6.36
MPSX	6	35	72	6.48
MPSX	9	35	90	6.78
MPSX	10	35	96	6.87
MPSX	15	35	126	7.18
MPSX	20	35	156	7.40

*Number of intervals is applies for grid linearization method

?

Test Problem 1 -> World Energy Model

This is a world petroleum model developed by Takayama to determine the optimal crude oil quantity processed and the final product optimal price.

Test Problem 2 -> Electrical Energy Model

This model tries to enhance the likelihood that economic efficiency will be obtained in the pricing and allocation of electrical energy in the USA.

TABLE III

A DESCRIPTION OF TEST PROBLEM 2

Name	Number of* Intervals	Rows	Columns	Density
MINOS		18	135	
MPSX	5	46	243	5.62
MPSX	6	46	270	5.70
MPSX	9	46	351	5.87
MPSX	10	46	378	5.91

*Number of intervals is applied for grid linearization method

Test Criteria

A good program should provide a fast, accurate solution to a problem. The program should take minimum time to prepare input. These criteria are not equally important for all users. While accurate solutions are probably critical to all, fast execution of the simplex algorithm may be important to somebody who has to solve rather large problems regularly. On the othe hand, ease of preparing input may be more important than solution time to a particular user. Three criteria for comparison are defined here: accuracy, computational efficiency, and human efficiency.

Accuracy

The word "approximation" implies that error is being introduced into the process. In one sense, this is always true; in another sense, this may never be true. Generally, if all problems solved by QP represent truly quadratic realities, then solution of a quadratic programming problem by any other procedure will introduce error. In this sense, error always occurs when approximations are used. However, the real test of approximation adequacy should not involve closeness of the approximated solution to the quadratic programming solution. Rather, the criteria should involve the real world purpose of the modeling effort. In this sense, the quadratic program itself may be an approximation.

Computational Efficiency

One facet of computational efficiency involves model size. In some cases, the number of rows and columns introduced by an approximation introduced by an approximation yields a larger problem than the associated quadratic problem. If the number of quadratic variables is large relative to the total number of variables, then the approximations of the problem size are likely to be larger than the Kuhn-Tucker system. Conversely, when relatively few quadratic variables are involved, the approximation may be much smaller. Thus approximation may yield either larger or smaller problems. However, size and solution time are not perfectly correlated [24]. Nevertheless, when the approximation is significantly smaller, a computational advantage will likely exist.

A second computational efficiency consideration involves algorithm characteristics. Unfortunately, two solution packages employing the same basic algorithm rarely, if ever, perform the same. Solution packages are quite different in numberical tactics employed to manage round-off error and data storage, etc. These affect computational efficiency. Thus, codes may possess characteristics which lead to differences in computational performance (for instance, codes may be good on large problems; good on certain structures, numerically stable or unstable). Programming language and style also affect computational efficiency. Crowder et al. [25] in discussing computational efficiency comparisons state that (a) results derived from small problems are not, in general, representative of results for larger applied problems, only a conjecture may be made; (b) results on one problem structure are not true on all problem structures; (c) comparing computer codes written by different programmers for different uses leads to conclusions which are valid only on the codes used, not on the methods themselves. Thus, computational efficiency depends on a complex set of issues involving the problem and algorithms at hand.

Human Efficiency

The packages accept input in a number of ways. Many approximations require numerous time-consuming steps once a QP problem has been formulated - forming a separable grid, for instance. Thus, solution via quadratic algorithms may reduce the human time spent on the problem.

When contemplating an approximation, one should ask whether or not the approximation procedure needs to be performed multiple times in the analysis. When the procedure is done repeatedly, the necessary human time increases. However, many approximations can be handled easily with a utility program. Thus, human efficiency problems may be mitigated by computerizing the approximation. However, this option itself has costs. Obviously this indicator is difficult to measure, but the importance should not be ignored.

MINOS, MPSX, and MINIT all support the MINOS format; the input formats for these problems are quite similar. As can be seen from Table IV, only nonzero coefficients need to be entered. The matrix generator and the starter program can convert MINOS input format to an external file accepted by MPSX Package and MINIT program. For a large problem, when the format becomes quite cumbersome, the matrix generator is proven to be powerful.

TABLE IV

MINOS INPUT FORMAT

```
MPSX/370 R1 6 PTF9     MPSCL EXECUTION

NAME            MPSX1
ROWS
 N   OBJECTVE
 G   CS1
 G   CS2
 G   PR111
 G   PR112
 G   PR121
 G   PR122
 G   PR211
 G   PR212
 G   PR221
 G   PR222
 G   PR311
 G   PR312
 G   PR321
 G   PR322
 G   PD11
 G   PD12
 G   PD21
 G   PD22
 G   PD31
 G   PD32
 G   ASR1
 G   ASR2
 G   DIS11
 G   DIS12
 G   DIS21
 G   DIS22
 G   DIS31
 G   DIS32
 E   CD1
 E   CD2
 E   CD3
 E   CD4
 E   CD5
 E   CD6
COLUMNS
     X1          OBJECTVE   -      1.00000    CS1         -       1.00000
     X1          PR111              .50000    PR112                .60000
     X1          ASR1       -      1.00000
     X2          OBJECTVE   -      1.00000    CS1         -       1.00000
     X2          PR211              .50000    PR212                .60000
     X2          ASR2       -      1.00000
     X3          OBJECTVE   -      1.20000    CS1         -       1.00000
     X3          PR311              .50000    PR312                50000
     X4          OBJECTVE   -      1.00000    CS2         -       1.00000
     X4          PR121              .70000    PR122                .40000
     X4          ASR1       -      1.00000
     X5          OBJECTVE   -      1.20000    CS2         -       1.00000
     X5          PR221              .70000    PR222                .40000
     X5          ASR2       -      1.00000
     X6          OBJECTVE   -      1.00000    CS2         -       1.00000
     X6          PR321              .60000    PR322                .50000
```

CHAPTER V

RESULTS AND DISCUSSION

The accuracy and speed of the software are important for the large problems. For comparison purposes, two test problems are solved on all of the systems. Table V exhibits results of the optimal solutions obtained by MINOS, MPSX, and MINIT for test problem 1. Table VI describes results of the optimal solutions obtained by MINOS, MPSX, and MINIT for test problem 2.

TABLE V

ACCURACY AND SPEED OF MINOS, MPSX, AND MINIT FOR TEST PROBLEM 1

Code Name	Number of Intervals (a)	Format Convert Time (b)	CPU Time (c)	Objective Value
MINOS			0.00079	6584.97
MPSX	5	0.00013	0.00046	6533.78
MPSX	6	0.00013	0.00047	6574.95
MPSX	9	0.00013	0.00048	6574.95
MPSX	10	0.00013	0.00048	6580.95
MPSX	15	0.00013	0.00050	6582.54
MPSX	20	0.00013	0.00052	6574.90
MINIT	5	0.00013	0.00048	6574.42
MINIT	6	0.00013	0.00059	6574.89
MINIT	9	0.00013	0.00065	6571.43
MINIT	10	0.00013	0.00067	6574.90
MINIT	15	0.00013	0.00096	6580.46
MINIT	20	0.00013	0.00129	6582.32

TABLE VI

ACCURACY AND SPEED OF MINOS, MPSX, AND MINIT
FOR TEST PROBLEM 2

Code Name	Number of Intervals	Format Convert Time (b)	CPU Time (c)	Objective Value
MINOS			Fail	Fail (d)
MPSX	5	0.00018	0.00053	318426.22
MPSX	6	0.00018	0.00054	328874.43
MPSX	9	0.00018	0.00057	346285.67
MPSX	10	0.00018	0.00058	349767.64
MINIT	5	0.00018	0.00081	318424.80
MINIT	6	0.00018	0.00090	328873.00
MINIT	9	0.00018	0.00114	346284.30
MINIT	10	0.00018	0.00124	349767.40

(a) The basic point values are available in reference [21]. In this case, these values are 12.7, 7.7, 4.3, 4.3, 18.0, and 19.0 respectively.
(b) This is the CPU time of the starter program.
(c) CPU time is measured in hours. All these jobs are run during weekend to minimize the effect of other jobs affecting CPU time.
(d) Failed to solve the problem because of unbounded (or badly scaled).

CPU time is the central processing time needed for executing the algorithm. Generally, CPU time increases with the increased number of variables. For LP problems, it is evident that MPSX take less CPU time than MINIT. For MPSX, the CPU time keeps almost steady for different number of intervals. For MINIT, the CPU time increases with the increased number of intervals. For MINOS, the CPU time is longer than that of the other two programs. Therefore, the solution algorithm of MINOS is much more expensive than the simplex algorithm for the same problem and the solution algorithm of MINIT is much more expensive than the MPSX package algorithm for the LP problems.

Numerical accuracy is a measurement of the algorithm's ability to compute a "correct" answer in the face of numerical instability. Table V and VI indicate that MINIT is able to obtain the same optimal solution as MPSX in two test problems. The purpose of this study is to approximate nonlinear separable functions with linear segments. Table V also indicates that the average accuracy differences between linear programming and quadratic programming is within 2%.

The ratio of the largest coefficient (147.9043) to the smallest coefficient (0.00023) in test problem 2 is about 10^6. This gives MINOS numerical difficulty, which means that MINOS is sensitive to scaling.

There is no fixed rule for arriving at either the optimal grid size or the optimal number of grids for a problem. However, the use of large grid sizes (large, relative to the total range of validity of the separable problem) may produce less reliable results. As illustrated in Table V, when the number of intervals is 5, the objective value obtained from the MPSX package is 6533.78.

In order to test the impact of basic point values, Table VII lists the test problem 1 objective values by using three different basic points on the MPSX package. Figure 3 interprets these results graphically.

The accuracy of nonlinear variables depends on the number of intervals and different basic point values, Table VII and Figure 3 indicate that there is no systematic pattern and Table VII also indicates that when using different basic points, it does not necessarily give closer values to MINOS's result. However, the use of finer subdivisions gives closer answers.

TABLE VII

THE IMPACT OF DIFFERENT BASIC POINTS
FOR TEST PROBLEM 1

Number of	Factor	Objective	Nonlinear Variables (b)					
Intervals	(a)	Value	1	2	3	4	5	6
5	0.85	6559.17	12.31	7.46	4.17	4.17	18.45	19.45
	1.0	6533.78	10.92	8.78	3.7	4.9	20.37	17.32
	1.2	6553.63	11.98	7.95	4.44	4.44	18.58	18.61
6	0.85	6571.59	11.48	8.07	4.51	4.51	18.87	18.56
	1.0	6574.95	12.7	7.7	4.3	4.3	18.0	19.0
	1.2	6534.6	11.68	7.39	3.96	3.96	19.16	19.85
9	0.85	6581.22	11.63	8.07	4.5	4.51	18.87	18.42
	1.0	6574.95	11.71	8.30	3.97	4.63	19.32	18.05
	1.2	6561.95	11.68	7.08	3.96	3.96	19.96	19.96
10	0.85	6572.71	11.02	7.46	4.17	4.17	19.58	19.60
	1.0	6574.95	12.7	7.7	4.3	4.3	18.0	19.0
	1.2	6573.94	11.88	7.95	4.44	4.44	18.58	18.71
MINOS		6584.97	11.69	7.76	4.45	4.3	18.74	19.06

(a) Factor 1 means the basic point values are the same as in reference [21]. Factor 0.85 means the basic point values which are the products of 0.85 and factor 1's basic point values (i.e., 10.795, 6.545, 3.655, 3.655, 15.3, and 16.15).

(b) Nonlinear variables are obtained by inverse program.

In separable programming, data are given as in a linear program, with the addition that there is one set of special variables for each nonlinear function (see Eqs. 24, 25, and 26). The simplex algorithm is modified to inhibit pricing (caluclation of the reduced cost coefficients) of the special variables within each set. Table VIII gives the results of the simplex algorithm applied to test problem 1.

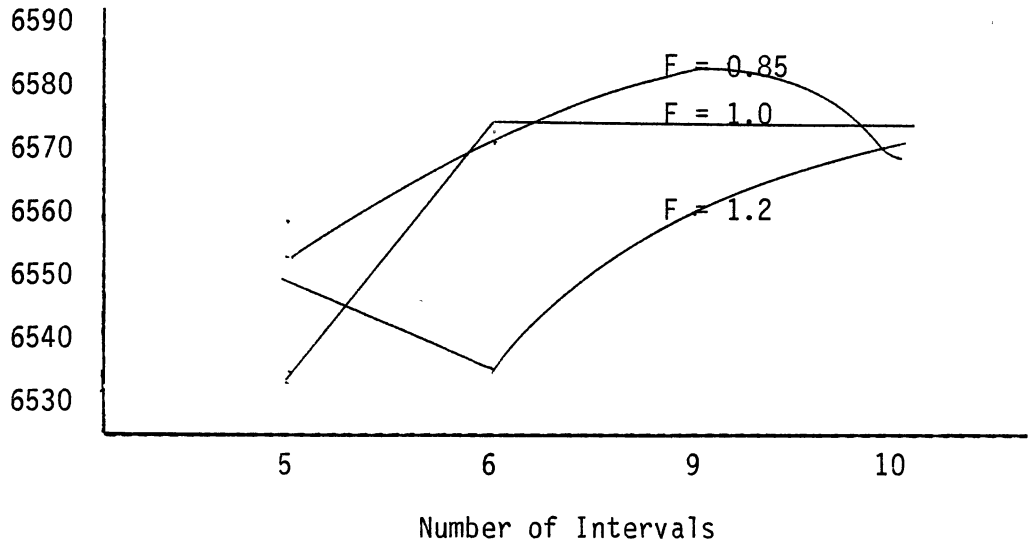

Figure 3. Objective Values vs. Number of Intervals

TABLE VIII

TEST PROBLEM 1 PARTIAL RESULTS OF MPSX SOLUTION OUTPUT, COLUMNS SECTION

Variable (1)	Activity	Quantity (q) (2)	Reduced Cost Coefficient
D_{11}	0.	3.81	96.71
D_{12}	1.	7.37	0.
D_{13}	0.	10.92	72.36
D_{14}	0.	14.48	312.76
D_{15}	0.	18.03	142.61
D_{51}	0.	5.40	43.71
D_{52}	0.	10.44	762.05
D_{53}	0.	15.48	254.02
D_{54}	0.02778	20.52	0.
D_{55}	0.97222	25.56	0.

(1) The number of intervals for this example is 5. Therefore, there are five special variables for each nonlinear function in this problem. D_{11} means the first special variable (Eq. 26) in the first set.

(2) Quantity values are the point values along the q Axis (see Figure 1).

To have a workable separable programming algorithm, it must be shown that the process terminates after a finite number of iterations and that the terminal solution is optimal in a local sense [9]. That is, no other feasible solution sufficiently close to it will have a better objective value. Consider the terminal solution and examine a particular set of special variables $S = (D_{i1}, \ldots, D_{in})$. In view of Eqs. (24, 25, 26) and Table VIII, there must be at least one element of S in the basis. Two cases can arise:

Case 1 Two (say D_{is}, $D_{i(s+1)}$) of S are basic, and $D_{is} \neq 0$, $D_{i(s+1)} \neq 0$.

Case 2 One (say D_{is}) of S is basic. Necessarily $D_{is} = 1$.

If case 1 occurs (D_{is}, $D_{i(s+1)}$ basic), express the nearby solution using only q_{is} (D_{is}) and $q_{i(s+1)}(D_{i(s+1)})$ - i.e., stay between A_{is} and $A_{i(s+1)}$ on the graph of A=f(q), Figure 1. If case 2 occurs stay between $A_{i(s-1)}$ and A_{is} or A_{is} and $A_{i(s+1)}$, using only $q_{i(s-1)}$ $(D_{i(s-1)})$ and $q_{is}(D_{is})$ or q_{is} (D_{is}) and $q_{i(s+1)}$ $(D_{i(s+1)})$. But all of these special variables were already priced at the last simplex iteration and found to have disadvantageously reduced cost coefficients. Hence, evaluating any nearby feasible solution via the reduced objective functional shows it to have a less desirable (at any rate, no better) objective value than the terminal one. So the terminal one is a local optimum if this is not a convex programming problem.

MPSX has the capability to check the sensitivity of the solution by ranging and parametric programming. MINOS has the capability to solve the nonlinear problems. A summary of the features is presented in Table IX.

TABLE IX

SUMMARY OF PROS AND CONS FOR MPSX, MINOS, AND MINIT

Code Name	Pros	Cons
MPSX	Sensitive analysis Post-optimal analysis CPU time is shorter	Non-portable
MINOS	Nonlinear constrained optimization Unconstrained optimization	CPU time is longer Sensitive to scaling
MINIT	Portable	CPU time depends strongly upon the number of variables

CHAPTER V

CONCLUSIONS AND RECOMMENDATIONS

Conclusions

Using grid linearization techniques to approximate nonlinear functions is proven to be useful. The analyzed results indicate that the approximation error is with 2%.

For LP problems, MINIT algorithm is much more expensive than the MPSX package. The CPU time is pretty steady for the MPSX package but not for the MINIT program. However, MINIT is portable but MPSX is not. For Quadratic programs, MINOS is sensitive to scaling, therefore may give numerical difficulties for large problems.

Quadratic programs should not always be approximated, nor should they always be solved as QPs. For small problems, considering computational efficiency and human effort, a quadratic programming solver will be better. Large problems with few quadratic variables seem to be candidates for approximation. The solution with linear programming is generally simpler and more reliable.

Recommendations

The benefits from approximation increase with problem size. Basically, linear programming codes can be utilized on problems which are larger than can be solved with any quadratic codes. Thus, future research work could be continued in three areas: First, a critical point at which approximation will always be better should be found. Second, criteria to help the user choose a method should be investigated. Third, if numerical difficulties arise for MINOS package, an automatic scaling subroutine should be conducted.

LITERATURE CITED

1. Samuelson, P. A. "Spatial Price Equilibrium and Linear Programming" American Economic Review, 42, (1952), pp. 283-303.

2. Takayama, T., and Judge, G. G. Spatial and Temporal Price and Allocation Models. North-Holland, New York, 1971.

3. Furtan, W. H., Nagy, J. G. and Storey, G. G., "The Impact on the Canadian Rapeseed Industry from Changes in Transport to Tariff Rates" American Journal of Agricultural Economics, 61, (1979), pp. 238-248.

4. Polito, J., McCarl, B. A. and Morin, T. L. "Solution of Spatial Equilibrium Problems with Benders Decomposition" Management Science, 26, (1980), pp. 593-605.

5. Duloy, J. H., and Norton, R. D. "Prices and Incomes in Linear Programming Models" American Journal of Agricultural Economics, 57, (1975), pp. 591-600.

6. Willett, Keith. "Moels of Spatial Competitive Market Equilibria: A Linear Programming Perspective" Oklahoma State University, 1986.

7. Wolfe, P. "Methods of Nonlinear Programming" in R. L. Graves and P. Wolfe, Recent Advances in Mathematical Programming. McGraw-Hill Book Company, New York, 1963.

8. Hadley, G. Nonlinear and Dynamic Programming, Addison-Wesley Publishing Company, Reading, Mass., 1964.

9. Miller, C. "The simplex method for local separable programming" in R,. L. Graves and P. Wolfe Recent Advances in Mathematical Programming. McGraw-Hill Book Company, New York, 1963.

10. Boggess, W. C., and Heady, E. O. A Separable Programming Analysis of U.S. Agricultural Export, Price and Income, and Soil Conservation Policies in 1985. Iowa State University, Iowa, 1980.

11. Neufville, R. D., and Stafford, J. H. Systems Analysis for Engineers and Managers McGraw-Hill Book Company, New York, 1971.

12. Rao, S. S. Optimization Theory and Applications Wiley Eastern Limited Company, Indian, 1984.

13. Wolfe, P. "The simplex method for quadratic programming" Econometrica, 27, (1959), pp. 382-398.

14. Cottle, R. W., and Dantzig, G. B. "Complementary Pivot Theory of Mathematical Programming" Linear Algebra and Its Application, 1, (1968), pp. 103-126.

15. Murtagh, B., and Saunders, M. "Large-Scale Linearly Constrained Optimization" Mathematical Programming, 14, (1978), pp. 41-72.

16. Murtagh, B., and Saunders, M. Modular In-core Nonlinear Optimization System (MINOS) Users' Manual Stanford University, 1983.

17. Benders, J. F., "Partitioning Procedures for Solving Mixed Variables Programming Problems." Numerische Mathematik, 4, (1962), pp. 238-252.

18. Ravindran, A., "Algorithm 431 A Computer Routine for Quadratic and Linear Programming Problems" Communications of the ACM, 15(9) (1972), pp. 818-820.

19. IBM IBM Mathematical Programming System Extended/370 - General Information Manual CH-19-1090-4, File No. S370-82 IBM Corporation, New York, 1979.

20. Rodolfo, C. S., and Subrata, K. S. "Algorithm 333" Communications of the ACM, 11 (6), (1968), pp. 437-440.

21. Takayama, T. "An Application of Spatial and Temporal Price Equilibrium Model to World Energy Modeling" Regional Science Association Papers, 41, (1979), pp. 43-58.

22. Uri, N. D. "The Pricing and Allocation of Electrical Energy" Energy Economics, 6, (1984), pp. 94-101.

23. Uri, N. D. Towards an Efficient Allocation of Electrical Energy D.C. Heath and Company, Lexington, Massachusetts, 1975.

24. McCarl, B. A. and Tice, T. "Should Quadratic Programming Problems be Approximated?" American Journal of Agricultural Economics, 62, (1984), pp. 585-589.

25. Crowder, H. P., Dembo, R. S., and Mulvey, J. M. "Reporting Computational Experiments in Mathematical Programming" Mathematical Programming, 15, (1978), pp. 316-329.

APPENDIXES

APPENDIX A

GRID REFINEMENT PROGRAM

```
**** TSO FOREGROUND HARDCOPY ****
DSNAME=U10822A.SHW6.CNTL

//U10822A JOB (10822,398-82-0158),'CHIANG',TIME=(0,5),
//  CLASS=A,MSGCLASS=X,NOTIFY=*
/*PASSWORD ????
// EXEC WATFIV
//FT06F001 DD SYSOUT=A
//FT12F001 DD DSN=U10822A.INPUT12.CNTL,DISP=SHR
$JOB     ,LIST
C THIS PROGRAM IS DESIGNED FOR COMPUTING AREA FOR DIFFERENT
C NUMBER OF SEGMENT AT SPECIFIC QUANTITY, ALSO THIS PROGRAM
C INITIALIZES SOME GIVEN VARIABLES (N,NC,NL,NV)
C N  -> NUMBER OF SEGEMNTS
C NC -> NUMBER OF CONSTRAINTS
C NL -> NUMBER OF NONLINEAR VARIABLES
C NV -> NUMBER OF LINEAR VARIABLES
C
      DIMENSION PM(6),W(6),WS(6),SG(6),S(20,6),A(20,6)
      DATA N,NC,NL,NV/6,28,6,36/
      DATA PM/12.7,7.7,4.3,4.3,18.0,19.0/
      DATA W/200.,115.,220.,165.,230.,75./
      DATA WS/13.33,10.,40.,30.,10.,2./
      DATA FACT/1.00/
      IOUT=6
      IND=36
      DO 90 I=1,NL
   90 PM(I)=PM(I)*FACT
      WRITE(IOUT,105) N
  105 FORMAT(1H1,17HNUMBER OF SEGMENT,I2)
      WRITE(12,126) N,NC,NL,NV,FACT
  126 FORMAT(5X,I2,1X,I2,1X,I2,1X,I2,1X,F5.2)
      DO 100 J=1,NL
      SG(J)=(1.7*PM(J)-0.3*PM(J))/N
      WRITE(IOUT,106) J
  106 FORMAT(5X,6HREGION,I2)
      WRITE(IOUT,107) PM(J),SG(J)
  107 FORMAT(5X,17HEQUILIBRIUM PRICE,F5.1,2X,17HLENGHT OF SEGMENT,F10.5)
      WRITE(IOUT,108)
  108 FORMAT(15X,8HQUANTITY,4X,4HAREA)
      DO 110 I=1,N
        S(I,J)=0.
  110 CONTINUE
      DO 120 I=1,N
        IF (I .EQ. 1) GO TO 130
        SUM=SUM+SG(J)
        GO TO 140
  130   S(I,J)=0.3*PM(J)
        SUM=S(I,J)
  140   S(I,J)=SUM
        A(I,J)=(W(J)-0.5*WS(J)*S(I,J))*S(I,J)
        IND=IND+1
        WRITE(IOUT,124) IND,S(I,J),A(I,J)
  124   FORMAT(5X,1HX,I3,2X,F10.2,F10.2)
        S(I,J)=-S(I,J)
        WRITE(12,125) S(I,J),A(I,J)
  125   FORMAT(5X,F10.2,F10.2)
  120 CONTINUE
  100 CONTINUE
      STOP
      END
C
```

APPENDIX B

MATRIX GENERATOR PROGRAM

```
**** TSO FOREGROUND HARDCOPY ****
DSNAME=U10822A.SHW8.CNTL

//U10822A JOB (10822,398-82-0158),'CHIANG',TIME=(0,5),
// CLASS=A,MSGCLASS=X,NOTIFY=*
/*PASSWORD ????
// EXEC WATFIV
//FT11F001 DD DSN=U10822A.INPUT11.CNTL,DISP=SHR
//FT12F001 DD DSN=U10822A.INPUT12.CNTL,DISP=SHR
//FT16F001 DD DSN=U10822A.INPUT6.CNTL,DISP=SHR
$JOB     ,LIST
C THIS MATRIX GENERATOR PROGRAM WILL CONVERT INPUT DATA FORMAT
C FOR MINOS TO INPUT DATA FORMAT ACCEPTED BY MPSX
C OVALUE ARE THE OBJECTIVE COEFFICIENTS FOR LINEAR VARIABLES
C NI -> THE BEGINNING OF X VARIABLE
      DIMENSION OVALUE(24)
      DATA OVALUE/-1.0,-1.0,-1.2,-1.0,-1.2,-1.0,
     *      -0.,-2.,-3.,-2.,-0.,-1.,-3.,-1.,-0.,
     *      -0.,-1.5,-2.,-1.5,-0.,-1.,-2.,-1.,-0./
      IOU=11
      IN1=12
      IN2=16
      IV=0
      CDVAL=1.0
      READ(IN1,35) N,NC,NL,NV,FACT
   35 FORMAT(5X,I2,1X,I2,1X,I2,1X,I2,1X,F5.2)
C     WRITE ROWS SECTION
      WRITE(IOU,45)
   45 FORMAT(4HNAME,10X,5HMPSX1/4HROWS/1X,1HN,2X,8HOBJECTVE)
      READ(IN2,55)
   55 FORMAT(//////////)
      DO 50 I=1,NC
      READ(IN2,56) CON,CNAM1,CNAM2
      WRITE(IOU,56) CON,CNAM1,CNAM2
   56 FORMAT(A2,A4,A4)
   50 CONTINUE
      DO 60 I=1,NL
      WRITE(IOU,65) I
   65 FORMAT(1X,3HE  ,2HCD,I1)
   60 CONTINUE
C     WRITE COLUMN SECTION
      READ(IN2,66) COL1,COL2
      WRITE(IOU,66) COL1,COL2
   66 FORMAT(A4,A4)
C SLASHES NUMBER CORRESPONDSTO NONLINEAR VARIABLES
      READ(IN2,67)
C 36 VARIABLE OCCPUY 82 ROWS IN COLUMN SECTION
   67 FORMAT(/////)
      DO 200 I=1,82
      READ(IN2,68)X,IVALUE,DRES1,DRES2,VALUE
   68 FORMAT(A5,I2,7X,A4,A4,F10.2)
      IVALUE=IVALUE-6
C FIRST 24 VARIABLES INVOLVED IN THE OBJECTIVE FUNCTION
      IF (IVALUE .GT. 24) GO TO 80
      IF (IVALUE .EQ. IV) GO TO 80
      IV=IV+1
      IF (IV .LT. 10) GO TO 76
C WRITE THE OBJECTIVE COEFFICIENTSFOR THE COLUMN SECTION
      WRITE(IOU,75) IV,OVALUE(IV)
   75 FORMAT(4X,1HX,I2,7X,8HOBJECTVE,F10.2)
      GO TO 80
   76 WRITE(IOU,77) IV,OVALUE(IV)
```

```
 77   FORMAT(4X,1HX,I1,8X,8HOBJECTVE,F10.2)
 80   IF (IVALUE .LT. 10) GO TO 90
      WRITE(IOU,68)X,IVALUE,DRES1,DRES2,VALUE
      GO TO 200
 90   WRITE(IOU,69)X,IVALUE,DRES1,DRES2,VALUE
 69   FORMAT(A5,I1,8X,A4,A4,F10.2)
200   CONTINUE
      IN=6*N
      DO 250 I=1,IN
      READ(IN1,105) QUANT,AREA
105   FORMAT(5X,F10.2,F10.2)
      IVN=I-1
      IK=(IVN/N)+1
      IV=I+NV
      IF (IV .GE. 100) GO TO 107
      WRITE(IOU,106) IV,AREA
106   FORMAT(4X,1HX,I2,7X,8HOBJECTVE,F10.2)
      GO TO 109
107   WRITE(IOU,108) IV,AREA
108   FORMAT(4X,1HX,I3,6X,8HOBJECTVE,F10.2)
109   GO TO (110,120,130,140,150,160),IK
      IF (IV .GE. 100) GO TO 116
110   WRITE(IOU,115) IV,QUANT,IV,CDVAL
115   FORMAT(4X,1HX,I2,7X,5HDIS11,3X,F10.2/4X,1HX,I2,7X,3HCD1,5X,F10.2)
      GO TO 250
116   WRITE(IOU,117) IV,QUANT,IV,CDVAL
117   FORMAT(4X,1HX,I3,6X,5HDIS11,3X,F10.2/4X,1HX,I3,6X,3HCD1,5X,F10.2)
      GO TO 250
120   IF (IV .GE. 100) GO TO 126
      WRITE(IOU,125) IV,QUANT,IV,CDVAL
125   FORMAT(4X,1HX,I2,7X,5HDIS12,3X,F10.2/4X,1HX,I2,7X,3HCD2,5X,F10.2)
      GO TO 250
126   WRITE(IOU,127) IV,QUANT,IV,CDVAL
127   FORMAT(4X,1HX,I3,6X,5HDIS12,3X,F10.2/4X,1HX,I3,6X,3HCD2,5X,F10.2)
      GO TO 250
130   IF (IV .GE. 100) GO TO 136
      WRITE(IOU,135) IV,QUANT,IV,CDVAL
135   FORMAT(4X,1HX,I2,7X,5HDIS21,3X,F10.2/4X,1HX,I2,7X,3HCD3,5X,F10.2)
      GO TO 250
136   WRITE(IOU,137) IV,QUANT,IV,CDVAL
137   FORMAT(4X,1HX,I3,6X,5HDIS21,3X,F10.2/4X,1HX,I3,6X,3HCD3,5X,F10.2)
      GO TO 250
140   IF (IV .GE. 100) GO TO 146
      WRITE(IOU,145) IV,QUANT,IV,CDVAL
145   FORMAT(4X,1HX,I2,7X,5HDIS22,3X,F10.2/4X,1HX,I2,7X,3HCD4,5X,F10.2)
      GO TO 250
146   WRITE(IOU,147) IV,QUANT,IV,CDVAL
147   FORMAT(4X,1HX,I3,6X,5HDIS22,3X,F10.2/4X,1HX,I3,6X,3HCD4,5X,F10.2)
      GO TO 250
150   IF (IV .GE. 100) GO TO 156
      WRITE(IOU,155) IV,QUANT,IV,CDVAL
155   FORMAT(4X,1HX,I2,7X,5HDIS31,3X,F10.2/4X,1HX,I2,7X,3HCD5,5X,F10.2)
      GO TO 250
156   WRITE(IOU,157) IV,QUANT,IV,CDVAL
157   FORMAT(4X,1HX,I3,6X,5HDIS31,3X,F10.2/4X,1HX,I3,6X,3HCD5,5X,F10.2)
      GO TO 250
160   IF (IV .GE. 100) GO TO 166
      WRITE(IOU,165) IV,QUANT,IV,CDVAL
165   FORMAT(4X,1HX,I2,7X,5HDIS32,3X,F10.2/4X,1HX,I2,7X,3HCD6,5X,F10.2)
      GO TO 250
166   WRITE(IOU,167) IV,QUANT,IV,CDVAL
167   FORMAT(4X,1HX,I3,6X,5HDIS32,3X,F10.2/4X,1HX,I3,6X,3HCD6,5X,F10.2)
250   CONTINUE
      READ(IN2,254) RHS
254   FORMAT(A3)
      WRITE(IOU,255) RHS
```

```
 255  FORMAT(A3)
      DO 300 I=1,4
      READ(IN2,305) RTH1,RTH2,RTH3,RNAME,RVAL
 305  FORMAT(A4,A4,A2,4X,A4,4X,F10.2)
      WRITE(IOU,305) RTH1,RTH2,RTH3,RNAME,RVAL
 300  CONTINUE
      DO 350 I=1,NL
      WRITE(IOU,355) I,CDVAL
 355  FORMAT(4X,6HRTHDSD,4X,2HCD,I1,5X,F10.2)
 350  CONTINUE
      WRITE(IOU,365)
 365  FORMAT(6HENDATA)
      STOP
      END
SENTRY
SIBSYS
//
```

APPENDIX C

TRANSLATOR PROGRAM

```
**** TSO FOREGROUND HARDCOPY ****
DSNAME=U10822A.SHW11.CNTL

C      THIS PROGRAM IS DESIGNED TO CONVERT MPSX OUTPUT
C      STANDARD FORMAT TO A READABLE FORMAT
C
       INTEGER*4      FILE,LIST,NOCOL,NOCOL2,I,J,L,M,N,P
       CHARACTER*8    NAME
       CHARACTER*8    COLUMN(50),VALUES(50)
       INTEGER*4      TYPE(100),VALNUM(100)
       CHARACTER*4    VALALF(100)
       CHARACTER*8    ENDSEC,ENDATA
       DATA           ENDSEC/'$ENDSEC$'/,ENDATA/'ENDATA  '/
       EQUIVALENCE   (VALUES(1),VALNUM(1),VALALF(1))
       FILE=4
       LIST=11
C
C      SKIP THE NAME,XDATA,RECORD
C
       READ(FILE)
C
       READ(FILE) NAME,NOCOL
       NOCOL2=2*NOCOL
       READ(FILE)(COLUMN(N),N=1,NOCOL)
       READ(FILE) (TYPE(N),N=1,NOCOL2)
       J=0
       DO 2I=2,NOCOL2,2
       J=J+TYPE(I)
  2    CONTINUE
       J=J/4
       READ(FILE) (VALALF(N),N=1,J)
C
C     PRINT THE IDENTIFICATION ARRAY, ONE VALUE PER LINE
C
       WRITE(LIST,9) NAME
       WRITE(6,9) NAME
  9    FORMAT(1H1,35X,'PRINTOUT OF THE FIELD ',A8)
C
       J=0
       DO 20 N=1,NOCOL
       L=J/4+1
       M=L+1
       P=L+19
       IF (TYPE(2*N-1) -2)10,14,12
C
C     NUMERIC - INTEGER -VALUE
C
 10    WRITE(LIST,11) COLUMN(N),VALNUM(L)
       WRITE(6,11) COLUMN(N),VALNUM(L)
 11    FORMAT(1H0,35X,A8,' = ',I8)
       GO TO 19
C
C     NUMERIC - INTEGER -VALUE
C
 12    WRITE(LIST,13) COLUMN(N),VALALF(L)
       WRITE(6,13) COLUMN(N),VALALF(L)
 13    FORMAT(1H0,35X,A8,' = ',F18.8)
       GO TO 19
C
C     ALPHAMERIC VALUE. LENGHT MAY BE 4,8 OR 80
C
 14    IF (TYPE(2*N)-8) 15,17,18
```

```
C
C     ALPHAMERIC VALUE - LENGHT = 4
C
   15 WRITE(LIST,16) COLUMN(N),VALALF(L)
      WRITE(6,16) COLUMN(N),VALALF(L)
   16 FORMAT(1H0,35X,A8,' = ',20A4)
      GO TO 19
C
C     ALPHAMERIC VALUE - LENGHT = 8
C
   17 WRITE(LIST,16) (COLUMN(N),VALALF(K),K=L,M)
      WRITE(6,16) (COLUMN(N),VALALF(K),K=L,M)
      GO TO 19
C
C     ALPHAMERIC VALUE - LENGHT = 8
C
   18 WRITE(LIST,16) (COLUMN(N),VALALF(K),K=L,P)
      WRITE(6,16) (COLUMN(N),VALALF(K),K=L,P)
C
   19 J=J-TYPE(2*N)
   20 CONTINUE
C
C      SKIP THE PENDSECP OF THE IDENTIFICATION ARRAY
C
      READ(FILE)
C
C      GET THE ROW AND COLUMN SECTION
C
   21 READ(FILE) NAME,NOCOL
      IF (NAME .EQ. ENDATA) GO TO 31
   22 WRITE(LIST,9) NAME
      WRITE(6,9) NAME
C
      READ(FILE) (COLUMN(N),N=1,NOCOL)
      READ(FILE)
      WRITE(LIST,23) (COLUMN(N),N=1,NOCOL)
   23 FORMAT(1H0/1H ,A8,12X,A8,12X,A8,8X,A8,8X,A8,11X,A8,4X,A8,4X,A8/)
C
   24 READ(FILE) (VALUES(N),N=1,NOCOL)
      IF (VALUES(1) .EQ. ENDSEC) GO TO 21
C
   25 WRITE(LIST,26) (VALUES(N),N=1,NOCOL)
      WRITE(6,26) (VALUES(N),N=1,NOCOL)
   26 FORMAT(1H ,D15.8,D20.8,D16.4,D16.4,D20.8,F11.0,A12,A12)
      GO TO 24
C
   31 RETURN
      END
```

APPENDIX D

MATHEMATICAL STATEMENT OF TEST PROBLEM 1

Test Problem 1 (Source: Reference 21)

Semi-positive definite matrix

Objective Function:

Maximize

$$200X_1 - 6.67X_1^2 + 115X_2 - 5X_2^2 + 220X_3 - 20X_3^2 +$$
$$165X_4 - 15X_4^2 + 230X_5 - 5X_5^2 + 75X_6 - X_6^2 - X_7 - X_8 -$$
$$1.2X_9 - X_{10} - 1.2X_{11} - X_{12} - 2X_{14} - 3X_{15} + 2X_{16} - X_{18} -$$
$$3X_{19} - X_{20} - 1.5X_{23} - 2X_{24} - 1.5X_{25} - X_{27} - 2X_{28} - X_{29}$$

Subject to the constraints:

Crude supply constraints:

$$- X_7 - X_8 - X_9 = - 20$$
$$- X_{10} - X_{11} - X_{12} = - 40$$

Try $X_7 + X_8 + X_9 = 20$, $X_{10} + X_{11} + X_{12} = 40$

Production-distribution constraints:

$$X_{31} + X_{33} - X_{13} - X_{14} - X_{15} \geq 0$$
$$X_{32} + X_{34} - X_{22} - X_{23} - X_{24} \geq 0$$
$$X_{35} + X_{37} - X_{16} - X_{17} - X_{18} \geq 0$$
$$X_{36} + X_{38} - X_{25} - X_{26} - X_{27} \geq 0$$
$$X_{39} + X_{41} - X_{19} - X_{20} - X_{21} \geq 0$$
$$X_{40} + X_{42} - X_{28} - X_{29} - X_{30} \geq 0$$

Distribution and final regional demand constraints:

$$X_{13} + X_{16} + X_{19} - X_1 \geq 0$$
$$X_{22} + X_{25} + X_{28} - X_2 \geq 0$$
$$X_{14} + X_{17} + X_{20} - X_3 \geq 0$$
$$X_{23} + X_{26} + X_{29} - X_4 \geq 0$$
$$X_{15} + X_{18} + X_{21} - X_5 \geq 0$$
$$X_{24} + X_{27} + X_{30} - X_6 \geq 0$$

Refinery process constraints:

$$0.5X_7 - X_{31} \geq 0$$
$$0.6X_7 - X_{32} \geq 0$$
$$0.7X_{10} - X_{33} \geq 0$$
$$0.4X_{10} - X_{34} \geq 0$$

$$0.5X_8 - X_{35} \geq 0$$
$$0.6X_8 - X_{36} \geq 0$$
$$0.7X_{11} - X_{37} \geq 0$$
$$0.4X_{11} - X_{38} \geq 0$$
$$0.5X_9 - X_{39} \geq 0$$
$$0.5X_9 - X_{40} \geq 0$$
$$0.6X_{12} - X_{41} \geq 0$$
$$0.5X_{12} - X_{42} \geq 0$$

Refinery capacity constraints:

$$- X_7 - X_{10} \geq - 15$$
$$- X_8 - X_{11} \geq - 15$$

And

$$X_j \geq 0 \quad j = 1, 2, \ldots\ldots, 42$$

APPENDIX E

MATHEMATICAL STATEMENT OF TEST PROBLEM 2

Test Problem 2 (Source: References 22 and 23)

Objective Function:

Maximize

$$147.9043X_1 - 0.002915X_1^2 + 147.8661X_2 -$$
$$0.00102X_2^2 + 130.4007X_3 - 0.00675X_3^2 + 136.4894X_4 -$$
$$0.00159X_4^2 + 105.6432X_5 - 0.0053X_5^2 + 73.14445X_6 -$$
$$0.00675X_6^2 + 117.9889X_7 - 0.00101X_7^2 + 119.1421X_8 -$$
$$0.00288X_8^2 + 91.9915X_9 - 0.00605X_9^2 + 57.6892X_{10} -$$
$$0.00114X_{10}^2 + 57.77284X_{11} - 0.00345X_{11}^2 + 52.26184X_{12} -$$
$$0.00028X_{12}^2 + 52.30450X_{13} - 0.00705X_{13}^2 + 43.20152X_{14} -$$
$$0.00024X_{14}^2 + 37.8308X_{15} - 0.00735X_{15}^2 + 40.74765X_{16} -$$
$$0.00032X_{16}^2 + 40.9598X_{17} - 0.0620X_{17}^2 + 36.24478X_{18} -$$
$$0.00018X_{18}^2 + 52.2091X_{19} - 0.001045X_{19}^2 + 48.31183X_{20} -$$
$$0.00023X_{20}^2 + 43.9390X_{21} - 0.0015X_{21}^2 + 49.52139X_{22} -$$
$$0.00060X_{22}^2 + 48.5061X_{23} - 0.00025X_{23}^2 + 77.90477X_{24} -$$
$$0.00047X_{24}^2 + 53.8488X_{25} - 0.00036X_{25} + 54.78305X_{26} -$$
$$0.00096X_{26}^2 + 102.4274X_{27} - 0.00062X_{27}^2 - 0.00518X_{28} -$$
$$0.88974X_{29} - 4.0758X_{30} - 5.9595X_{31} - 4.40342X_{32} -$$
$$4.87846X_{33} - 7.46253X_{34} - 8.18329X_{35} - 12.95419X_{36} -$$
$$0.88974X_{37} - 0.00518X_{38} - 3.44514X_{39} - 4.95627X_{40} -$$
$$3.53524X_{41} - 4.01028X_{42} - 6.57797X_{43} - 7.64682X_{44} -$$
$$12.4177X_{45} - 4.0758X_{46} - 3.44154X_{47} - 0.00518X_{48} -$$
$$2.06919X_{49} - 2.99467X_{50} - 2.67934X_{51} - 3.84238X_{52} -$$
$$4.15361X_{53} - 8.96137X_{54} - 5.95959X_{55} - 4.95627X_{56} -$$
$$2.06916X_{57} - 0.00518X_{58} - 3.30181X_{59} - 2.92505X_{60} -$$
$$3.84238X_{61} - 2.48687X_{62} - 7.66279X_{63} - 4.40342X_{64} -$$
$$3.53524X_{65} - 2.99467X_{66} - 3.30181X_{67} - 0.00518X_{68} -$$

$0.63584X_{69} - 3.36324X_{70} - 5.78350X_{71} - 10.4193X_{72} -$
$4.87846X_{73} - 4.01028X_{74} - 2.67934X_{75} - 2.92505X_{76} -$
$0.63584X_{77} - 0.00518X_{78} - 2.72849X_{79} - 5.33712X_{80} -$
$9.79679X_{81} - 7.46253X_{82} - 6.57797X_{83} - 3.84238X_{84} -$
$3.84238X_{85} - 3.36324X_{86} - 2.72849X_{87} - 0.00518X_{88} -$
$3.21581X_{89} - 7.22501X_{90} - 8.18329X_{91} - 7.64602X_{92} -$
$4.15361X_{93} - 2.48687X_{94} - 5.78350X_{95} - 5.33712X_{96} -$
$3.21581X_{97} - 0.00518X_{98} - 5.18151X_{99} - 12.9542X_{100} -$
$12.41772X_{101} - 8.96136X_{102} - 7.66729X_{103} - 10.41926X_{104} -$
$9.79679X_{105} - 7.22501X_{106} - 5.18151X_{107} - 0.00518X_{108} -$
$1.03X_{109} - 4.23X_{110} - 3.47X_{111} - 1.03X_{112} - 4.23X_{113} -$
$3.47X_{114} + 1.03X_{115} - 4.23X_{116} - 3.47X_{117} - 1.03X_{118} -$
$4.23X_{119} - 3.47X_{120} + 1.03X_{121} - 4.23X_{122} - 3.47X_{123} -$
$1.03X_{124} - 4.23X_{125} - 3.47X_{126} + 1.03X_{127} - 4.23X_{128} -$
$3.47X_{129} - 1.03X_{130} - 4.23X_{131} - 3.47X_{132} + 1.03X_{133} -$
$4.23X_{134} - 3.46X_{135}$

Subject to the constraints:

Production-distribution constraints:

$$- X_1 - X_{10} - X_{19} + X_{28} + X_{37} + X_{46} + X_{55} + X_{64} + X_{73} + X_{82} + X_{91} + X_{100} \geq 0$$

$$- X_2 - X_{11} - X_{20} + X_{29} + X_{38} + X_{47} + X_{56} + X_{65} + X_{74} + X_{83} + X_{92} + X_{101} \geq 0$$

$$- X_3 - X_{12} - X_{21} + X_{30} + X_{39} + X_{48} + X_{57} + X_{66} + X_{75} + X_{84} + X_{93} + X_{102} \geq 0$$

$$- X_4 - X_{13} - X_{22} + X_{31} + X_{40} + X_{49} + X_{58} + X_{67} + X_{76} + X_{85}$$

$$+ X_{94} + X_{103} \geq 0$$

$$- X_{5} - X_{14} - X_{23} + X_{32} + X_{41} + X_{50} + X_{59} + X_{68} + X_{77} + X_{86} + X_{95} + X_{104} \geq 0$$

$$- X_{6} - X_{15} - X_{24} + X_{33} + X_{42} + X_{51} + X_{60} + X_{69} + X_{78} + X_{87} + X_{96} + X_{105} \geq 0$$

$$- X_{7} - X_{16} - X_{25} + X_{34} + X_{43} + X_{52} + X_{61} + X_{70} + X_{79} + X_{88} + X_{97} + X_{106} \geq 0$$

$$- X_{8} - X_{17} - X_{26} + X_{35} + X_{44} + X_{53} + X_{62} + X_{71} + X_{80} + X_{89} + X_{98} + X_{107} \geq 0$$

$$- X_{9} - X_{18} - X_{27} + X_{36} + X_{45} + X_{54} + X_{63} + X_{72} + X_{81} + X_{90} + X_{99} + X_{108} \geq 0$$

Distribution and final regional demand constraints:

$$- X_{28} - X_{29} - X_{30} - X_{31} - X_{32} - X_{33} - X_{34} - X_{35} - X_{36} + X_{109} + X_{110} + X_{111} \geq 0$$

$$- X_{37} - X_{38} - X_{39} - X_{40} - X_{41} - X_{42} - X_{43} - X_{44} - X_{45} + X_{112} + X_{113} + X_{114} \geq 0$$

$$- X_{46} - X_{47} - X_{48} - X_{49} - X_{50} - X_{51} - X_{52} - X_{53} - X_{54} + X_{115} + X_{116} + X_{117} \geq 0$$

$$- X_{55} - X_{56} - X_{57} - X_{58} - X_{59} - X_{60} - X_{61} - X_{62} - X_{63} + X_{118} + X_{119} + X_{120} \geq 0$$

$$- X_{64} - X_{65} - X_{66} - X_{67} - X_{68} - X_{69} - X_{70} - X_{71} - X_{72} + X_{121} + X_{122} + X_{123} \geq 0$$

$$- X_{73} - X_{74} - X_{75} - X_{76} - X_{77} - X_{78} - X_{79} - X_{80} - X_{81} + X_{124} + X_{125} + X_{126} \geq 0$$

$$- X_{82} - X_{83} - X_{84} - X_{85} - X_{86} - X_{87} - X_{88} - X_{89} - X_{90} + X_{127} + X_{128} + X_{129} \geq 0$$

$$- X_{90} - X_{91} - X_{92} - X_{93} - X_{94} - X_{95} - X_{96} - X_{97} - X_{99} + X_{130} + X_{131} + X_{132} \geq 0$$

$$- X_{100} - X_{101} - X_{102} - X_{103} - X_{104} - X_{105} - X_{106} - X_{107} - X_{108} + X_{133} + X_{134} + X_{135} \geq 0$$

And

$$X_j \geq 0 \quad J = 1, 2, \ldots\ldots, 135$$

APPENDIX F

INFEASIBLE SOLUTION OF LEMKE'S ALGORITHM FOR TEST PROBLEM 1

Test Problem 1 (Source: Reference 21)

z (1) = 15.03759	z (2) = 11.5	z (3) = 5.5
z (4) = 5.5	z (5) = 23.0	z (6) = 37.5
W (7) = 1.0	W (8) = 1.0	W (9) = 1.2
W(10) = 1.0	W(11) = 1.2	W(12) = 1.0
W(13) = 0.0	W(14) = 2.0	W(15) = 3.0
W(16) = 2.0	W(17) = 0.0	W(18) = 1.0
W(19) = 3.0	W(20) = 1.0	W(21) = 0.0
W(22) = 0.0	W(23) = 1.5	W(24) = 2.0
W(25) = 1.5	W(26) = 0.0	W(27) = 1.0
W(28) = 2.0	W(29) = 1.0	W(30) = 0.0
W(31) = 0.0	W(32) = 0.0	W(33) = 0.0
W(34) = 0.0	W(35) = 0.0	W(36) = 0.0
W(37) = 0.0	W(38) = 0.0	W(39) = 0.0
W(40) = 0.0	W(41) = 0.0	W(42) = 0.0
W(43) = 20.0	W(44) = 40.0	W(45) = 0.0
W(46) = 0.0	W(47) = 0.0	W(48) = 0.0
W(49) = 0.0	W(50) = 0.0	W(51) = 15.03759
W(52) = 11.5	W(53) = 5.5	W(54) = 5.5
W(55) = 23.0	W(56) = 37.5	W(57) = 0.0
W(58) = 0.0	W(59) = 0.0	W(60) = 0.0
W(61) = 0.0	W(62) = 0.0	W(63) = 0.0
W(64) = 0.0	W(65) = 0.0	W(66) = 0.0
W(67) = 0.0	W(68) = 0.0	W(69) = 15.0
W(70) = 15.0		

VITA

Shin An Chiang

Candidate for the Degree of

Master of Science

Thesis: SEPARABLE PROGRAMMING ANALYSIS OF SPATIAL COMPETITIVE MARKET MODELS

Major Field: Computing and Information Sciences

Biographical:

Personal Data: Born in Taipei, Taiwan, June 5, 1950, the son of Chiang Yung Ching and Yu Wen Quey.

Education: Receivd Bachelor of Science Degree in Engineering from National Cheng Kung University in July, 1973; received Master of Science Degree from University of Wisconsin, Madison, in December 1980; completed requirements for the Master of Science Degree At Oklahoma State University in December, 1986.

Professional Experience: Research Assistant, Water Resources Research Center, Oklahoma State University, June, 1984 to October, 1986.