September 1989 A.E. Res. 89-17

A GUIDE TO USING THE GENERAL ALGEBRAIC MODELLING SYSTEM (GAMS) FOR APPLICATIONS IN AGRICULTURAL ECONOMICS

by
Robert W. Jefferson
and
Richard N. Boisvert

Department of Agricultural Economics
Cornell University Agricultural Experiment Station
New York State College of Agriculture and Life Sciences
A Statutory College of the State University
Cornell University, Ithaca, New York 14853

It is the policy of Cornell University actively to support equality of educational and employment opportunity. No person shall be denied admission to any educational program or activity or be denied employment on the basis of any legally prohibited discrimination involving, but not limited to, such factors as race, color, creed, religion, national or ethnic origin, sex, age or handicap. The University is committed to the maintenance of affirmative action programs which will assure the continuation of such equality of opportunity.

A GUIDE TO USING THE GENERAL ALGEBRAIC MODELLING SYSTEM (GAMS) FOR APPLICATIONS IN AGRICULTURAL ECONOMICS

by

Robert W. Jefferson

and

Richard N. Boisvert

Abstract

This manual was prepared to introduce graduate students in agricultural economics to GAMS, a mathematical programming software combination consisting of a modelling language, GAMS 2.05, and linear, non-linear and integer programming solvers. It begins with a tutorial using a problem in farm planning. In later sections, this same example, combined with five years of data on gross margins, is used to illustrate how to solve a variety of risk programming applications (e.g. MOTAD, Mean-Gini and EV models). Other non-linear programming applications include a spatial allocation model and a non-stochastic dynamic programming problem. The manipulation of an input-output system is also discussed. The GAMS code for each of these applications is included in an appendix.

Table of Contents

INTRODUCTION

The purpose of this manual is to introduce users to a very general, yet accessible, mathematical programming software combination, known simply as GAMS. The basic package consists of the mathematical modelling system, GAMS 2.05, a modelling language, and a linear programming solver called BDM-LP 1.01 (after Brooke, Drud, and Meeraus, who created it). In addition, there are two optional solvers, MINOS 5.2, which is used for solving non-linear problems, and ZOOM, which solves integer programming problems.

Linear programming problems are solved with BDM-LP in conventional fashion. Linear inequality constraints are converted to equalities by the introduction of slack variables. Solutions are obtained through the adjustment of variables in the basis, and through the exchange of basic and non-basic variables. This process continues until the optimum feasible value of the objective function is obtained.

MINOS (for Modular In-core Non-linear Optimization System) was developed at the Department of Operations Research at Stanford University. It is capable of solving optimization problems of the form:[1]

(1) $\quad \text{Max} \quad F(X) + c'X + d'Y$

subject to

(2) $\quad f(X) + A_1Y \leq b_1$

(3) $\quad A_2X + A_3Y \leq b_2$

(4) $\quad L \leq \begin{bmatrix} X \\ Y \end{bmatrix} \leq u$

The non-linear and linear choice variables are represented by the vectors "X" and "Y" respectively, with lower and upper bounds of L and u. "F(X)" and "f(X)" are non-linear, twice differentiable functions of the variables contained in "X". The vectors "c" and "d" contain constant objective function coefficients. The "A" matrices contain coefficients which describe the linear components of the constraint set. Finally, the "b" vectors represent the right-hand-side values of the constraints.

For problems involving optimization of a non-linear function, subject to linear constraints, MINOS uses a version of Wolfe's reduced-gradient algorithm. Again, all inequalities are converted to equalities, and the set of choice variables is partitioned into non-

---

[1] The description of MINOS is adapted from G. Fox, Mathematical Programming with MINOS: A Handbook for Economists, Ontario Agricultural College, University of Guelph, 1986.

basic, basic, and "superbasic" variables. The number of superbasics in the optimal solution is proportional to the degree of non-linearity of the objective function. Optimization proceeds through changing the values of the superbasic variables, with the basic and non-basic variables being adjusted to ensure feasibility. When a basic or superbasic variable encounters one of its bounds, it is transferred out of the basis. When no further improvement in the objective function can be obtained for a given set of basic and superbasic variables, non-basic variables are considered as candidates. An optimum is identified when it is no longer possible to improve the value of the objective function through substitutions of this sort.

A problem with non-linear constraints is solved by MINOS using a variant of the Projected Augmented Lagrangian algorithm, developed by Murtagh and Saunders.[2] A linear approximation to the constraint set is constructed, and the algorithm proceeds to solve the resulting linearly-constrained problem. Following an arbitrary amount of progress toward solution of the approximation to the problem, the values of the choice variables are used to recompute a linear approximation to the constraint set.

Each linear approximation is called a major iteration. Within each major iteration, the values of the choice variables are adjusted in a series of minor iterations, attempting to optimize an augmented Lagrangian (augmented by a quadratic "penalty" term, which penalizes poor approximation). The process continues until a point arbitrarily close to a constrained optimum is obtained.

GAMS (for General Algebraic Modelling System) was developed at the World Bank as a general mathematical modelling language. It is used here as a sort of pre- and post-processer, attached to the solvers. Using GAMS, one can write a model in a computer code which differs only slightly from standard mathematical notation and submit it to the solvers. Once the solution is obtained, GAMS can be used to perform further calculations on the results, or as a report writer.

To date, there exists extensive documentation on GAMS, including a tutorial which takes the reader through the solution of a small transportation problem. This is an extremely simple example, which does not involve the use of some of the most essential modelling features of GAMS. These, and other features which are useful in agricultural economics applications, are described throughout the several hundred pages of documentation.[3]

---

[2] This algorithm is described in B. A. Murtagh and M. A. Saunders, "Large-Scale Linearly Constrained Optimization", Mathematical Programming 14 (1978) 41-72 and B. Murtagh, and M. A. Saunders, "A Projected Lagrangian Algorithm and its Implementation for Sparse Nonlinear Constraints", Mathematical Programming Study, 16 (1982) 84-117.

[3] The major GAMS documents are: A. Meeraus, General Algebraic Modelling System: Preliminary User's Guide, Version 1.0, The World Bank, February, 1982, and D. Kendrick and A. Meeraus, GAMS, An

The Users Guide by Brooke, Kendrick and Meeraus (footnote 3) is the documentation that is supplied with the commercial available micro-computer version of GAMS. It includes introductory material, as well as material for the advanced user. It is more readable than the earlier documentation and became available shortly after a draft of this bulletin had been prepared and used in an agricultural economics graduate course in mathematical programming. The orientation of our discussion and the topics emphasized in this bulletin are those most important to agricultural and resource economics applications.

To facilitate the discussion, a simple farm planning problem example is used as the basis for an extensive tutorial. In later sections, this same example, combined with five years of data on gross margins, is used to illustrate how to prepare the data inputs efficiently and to solve a variety of risk programming applications (e.g. E.V., MOTAD, Mean-Gini). Other non-linear programming applications include a spatial allocation model and a non-stochastic dynamic programming problem. The manipulations of an input-output system are also discussed.

The manual is divided into four main sections, with two appendices. Section I consists of the linear programming tutorial. Section II provides further information for users interested specifically in linear programming applications. Section III discusses some of the special features of GAMS, such as the conditional operator "$", and operations on sets. Section IV contains a discussion of non-linear programming, including some of the special aspects of dynamic programming, such as leads and lags.

Introduction, The World Bank, December, 1987, and A. Brooke, D. Kendrick, and A. Meeraus, GAMS - A User's Guide, The Scientific Press, Redwood City, California, 1988. The major MINOS document is: B. A. Murtagh and M. A. Saunders, MINOS 5.0 User's Guide, Systems Optimization Laboratory, Department of Operations Research, Stanford University, 1983.

SECTION I

A GAMS Tutorial, Using Linear Programming

The purpose of this section is to outline the essentials of formulating and running linear programming problems in GAMS. GAMS was originally designed as a mainframe package. Thus, in using it, it is necessary to prepare a complete file, and submit the file to GAMS to be solved in a "batch" mode. The file may be prepared using any text editor, but the use of tab characters or other format characters is prohibited. An attractive feature of GAMS, however, is that one can embed comments in the input file by beginning each comment line with an asterisk (*). This feature is especially helpful to document the structure of large models.

In outlining the essential features of this package, an example problem, in which a farmer wishes to choose levels of three activities (the production of standard wheat (X_1), oats (X_2) and new wheat (X_3)), subject to four resource constraints, that maximize net farm revenue, is used. Written in standard notation, the problem is:

$$
\begin{aligned}
(5) \qquad \text{Max} \quad & 108.3X_1 + 66.36X_2 + 127.56X_3 \\
\text{s.t.} \quad & X_1 + X_2 + X_3 \leq 12 \\
& X_1 + X_3 \leq 8 \\
& 30X_1 + 20X_2 + 40X_3 \leq 400 \\
& 5X_1 + 5X_2 + 8X_3 \leq 80 \\
& X_1, X_2, X_3 \geq 0.
\end{aligned}
$$

The components of the GAMS representation of this example are discussed in the following order:

INPUT
- Sets
- Data Input: PARAMETER, TABLE, and SCALAR
- Variables
- Equations
- MODEL and SOLVE Statements

SUBMITTING THE FILE

OUTPUT
- Compiler Output
  - Echo Print
  - Reference Maps
- Execution Output
  - Equation Listing
  - Column Listing
  - Model Statistics
  - Solution Report

INPUT

Sets

When formulating a problem in GAMS, one begins by declaring the sets over which parameters and variables are defined. The structure of the command is:

```
SETS  (identifier1)  explanatory text
      /element1,element1,element1, ... ,element1/
      (identifier2)  explanatory text
      /element2,element2,element2, ... ,element2/

      (identifierN)  explanatory text
      /elementN,elementN,elementN, ... ,elementN/
```

The declaration of sets begins with the keyword SETS (or SET if there is only one), which can be typed anywhere on a line except in the first column. (The first column is reserved for indicators of comment statements and system commands). Following the keyword, the set name, or identifier, is typed, followed by a space and optional explanatory text. An identifier can be up to ten characters long, the first character of which must be a letter. Identifiers cannot include embedded blanks, nor can they contain special characters such as ?&#-+=@%()_"'!.

The elements of a set must be separated either by commas or by end-of-lines, and enclosed within slashes. Element names (called "labels") can be up to ten characters in length. They must also begin with a letter, and must not contain embedded blanks or special characters other than "+" and "-". (A blank in an element list serves to separate the element name, or label, from associated explanatory text).

For example, set declaration for the set of activities in the example problem might look as follows:

```
SETS
 J ACTIVITIES
   /STD-WHT, OATS, NEW-WHT/
```

The name of the set is "J", which may also be considered to be the name of its generic element. The descriptive text is "ACTIVITIES", and the three elements follow on the next line. Note how multi-word labels are incorporated, by use of the hyphen.

It is possible to append descriptive text to set elements, in much the same way as one does with set identifiers. This is shown below, using the farm problem, naming the activities by the conventional X1, X2, and X3.

```
SETS
 J ACTIVITIES
   /X1 STANDARD, X2 OATS, X3 NEW/
```

In this case, the GAMS compiler considers X1 to be the label for the first element and the remaining characters between the first blank and the comma to be the description of the first element. This is the reason for excluding embedded blanks from the element labels themselves.

This set can also be entered using end-of-lines, rather than commas as separators, as follows:

```
SETS
 J ACTIVITIES
   /X1   STANDARD
    X2   OATS
    X3   NEW/
```

Large, unordered sets pose no entry problems, since elements may be separated using either commas or end-of-lines. Thus, if the set of activities contained twenty-five elements, the set could be entered as:

```
SETS
 J ACTIVITIES
   /AY,BEE,CEE,DEE,EE,EFF,GEE,AITCH,AI,JAY,KAY,ELL,EMM,ENN
    OH,PEE,KYOO,ARR,ESS,TEE,YOO,VEE,DUBBELYOO,EX,WHY/
```

In the case of large, ordered sets, such as time periods in dynamic problems, GAMS accepts a more compact method of set element entry. For example, if a problem contains constraint equations for twenty different resources, which are simply numbered consecutively, they may be declared by entering:

```
SETS
 R RESOURCES
   /RES1*RES20/
```

This is almost identical to the English convention of representing an ordered set as "RES1-RES20", which is generally understood to mean "RES1, RES2, RES3,...,RES20".

More simply, the resources might be entered as:

```
 R RESOURCES /1*20/
```

While this is an easy way to specify set element labels, it sometimes causes confusion for beginning users. The difficulty arises when the user attempts to perform operations on the numbers used as labels. Since these are only names for elements, it is no more sensible to use them directly in equations than it would be to multiply Peter by Paul.

In the farm problem, there are four resource constraints. The set of resource rows is declared as part of the SETS portion of the GAMS input, using the convention just described. The two sets needed for this problem could then appear as:

```
SETS
  J ACTIVITIES
    /STD-WHT, OATS, NEW-WHT/
  I INPUTS
    /RES1*RES4/
```

Subsets of primary sets can be created through assignment statements. This is discussed in more detail in Section III.

Once the sets have been defined, they can be used in entering data for the problem.

Data Input: PARAMETER, TABLE, and SCALAR

There are three forms of information that are entered in the GAMS language. These are scalars, lists, and tables. (A scalar is simply a one-element list, so there are really only two types of data input. However, scalars are entered using a unique keyword.)

When preparing any form of data, the commands are structured in essentially the same manner. The structure of the command is:

```
(Keyword)  Name(domain)   Descriptive text
     /Actual Data/
```

The keywords used are:

PARAMETER	(For entering parameter lists)
TABLE	(For two-dimensional tables)
SCALAR	(For scalars, or one-entry lists) [4]

Names of parameters follow the same convention as that of set names. Normally, these will be short--one or two letters is generally all that is required--although parameter names can be up to ten characters in length.

If domain-checking is desired, the domain of the parameter is appended to the parameter name inside parentheses.[5]

---

[4] All of these keywords refer to different ways of entering the "parameters" of the problem being studied. The commands in the GAMS language are actually PARAMETER LIST, PARAMETER TABLE, and PARAMETER SCALAR. However, the use of defaults allows the modeller to condense PARAMETER LIST to PARAMETER, PARAMETER TABLE to TABLE, and PARAMETER SCALAR to SCALAR.

[5] Once the input file for the problem is complete, it is checked for consistency before being submitted for solution. If the domain(s) of a parameter is included in the parameter declaration itself, the GAMS compiler will check to ensure that the list or table of data matches the set(s) to which the data belong. For example, if the domain of the parameter "R" was mistakenly declared to be "I", the set of resources, rather than "J", the set of activities, GAMS would have

When lists of parameters are prepared as input, the actual data are entered in the same format as that of the original sets themselves, except that each element contains a pair of entries. Elements of parameter lists are separated either by commas or by end-of-lines. The following statements are equivalent ways of entering the parameter "R", the objective function coefficients:

```
(1)  PARAMETER R(J)  REVENUE OF ACTIVITIES
      /STD-WHT 108.3, OATS 66.36, NEW-WHT 127.58/

(2)  PARAMETER R(J)  REVENUE OF ACTIVITIES
      /STD-WHT     108.3
      OATS          66.36
      NEW-WHT      127.58/
```

The right-hand-side values are entered in similar fashion. These are defined over the rows, or set "I", and are named "B", in accordance with linear programming convention:

```
PARAMETER B(I) AVAILABILITY OF INPUTS
  /RES1   12
   RES2    8
   RES3  400
   RES4   80/
```

While the right-hand-side values are entered simply as another list, the "A" matrix can be entered in a tabular form, initialized by the keyword TABLE, as follows:[6]

```
TABLE A(I,J)  USE OF INPUTS PER ACTIVITY
        STD-WHT     OATS     NEW-WHT
RES1      1          1         1
RES2      1                    1
RES3     30         20        40
RES4      5          5         8
```

There are a number of things to note about table input. First, a table is two-dimensional, so the parameter "A" is defined over the two sets "I" and "J". **The Order of Domain Entry Matters!** The first domain indexed corresponds to the rows of the table, and the second

---

expected entries for "RES1-RES4". Not finding these, it would have signaled an error. While error messages are annoying, in this case they serve to ensure that the GAMS job is internally consistent before being submitted to the solver.

[6] In this simple example, all rows of the "A" matrix can be entered on one line. In larger problems, this is not possible. This issue is discussed in Section II, in the subsection entitled "Large Tables".

corresponds to the columns. The use of domain-checking is recommended, as an error will be noted if the domains are entered in the wrong order.

Tables can be entered fairly freely, subject only to the rule that at least one digit of a number must be directly below one character of the corresponding column heading, and no digits of a number for one column may be directly below any character of another column heading. It is not necessary to enter zeros. **Note that tabular data are not enclosed within slashes.**

Scalars are entered in exactly the same way as lists, replacing the keyword PARAMETER with SCALAR. Once again, the actual number will be enclosed within slashes. (There are no scalars in the example problem. Examples of the SCALAR Keyword are found in Appendix B, B.2, B.3, B.4 and B.7.)

In addition to the three methods of direct data input, there is one other type of input which is extremely useful, dealing with data which are computed. Later examples of this include computation of the variance-covariance matrix for quadratic programming, and the set of differences for the Mean-Gini programming formulation. Since this type of data input is not required for the example problem, discussion of it is deferred until Section II, in a subsection entitled "Manipulation of Data Using Assignment Statements".

Variables

Once the data have been entered, it would seem natural to proceed with writing the model equations themselves. However, one rule of GAMS is that all components of the structural equations must be declared before the equations themselves are entered. Although the parameters have now been entered, the choice variables and variable representing the function to be optimized must still be declared.

The format for the declaration of variables is similar to that for sets and data, beginning with the keyword VARIABLE(S). Following this is a variable name (up to ten characters), the domain of the variable (if domain checking is desired), and explanatory text.

The structure of the command is:

```
VARIABLES  Name1(domain)  Descriptive Text
           Name2(domain)  Descriptive Text

           NameN(domain)  Descriptive Text ;
```

Following the declaration of variables, restrictions can be put on some, or all of them. In this example, we need to incorporate the restriction that the activity levels be non-negative. For reasons of its own, GAMS calls this restriction POSITIVE. The variables listing for the example problem is:

```
VARIABLES
  Z     NET REVENUE
  X(J)  ACTIVITY LEVELS
POSITIVE VARIABLE X;
```

Note the semicolon. The general rule is that all GAMS statements should end with a semicolon, but this can be relaxed when the next statement begins with a keyword. (The semicolon is not required after the line "Z NET REVENUE" because the next line is simply another element in the list of variables, and is therefore part of the same statement.)

Variables can be of five types, POSITIVE (which means non-negative), NEGATIVE (which, of course, means non-positive), INTEGER, BINARY, or FREE. Unless otherwise specified, GAMS will consider all variables to be free (able to take on all values).

Equations

The structural equations are the center of the model. These are entered in two parts. First the declaration names each equation for future reference; then the equation is defined. The structure of the commands is:

```
EQUATIONS
  Name1   Descriptive Text
  Name2   Descriptive Text            (Declarations)

  NameN   Descriptive Text ;
  Name1..  (Definition) ;
  Name2..  (Definition) ;             (Definitions)

  NameN..  (Definition) ;
```

The declarations section is simply a list of equation names, with optional explanatory text. An equation name is a single word of up to ten characters, and the text is a series of words. At the end of the list, a semicolon is required to alert the GAMS compiler that the list of names is complete, and the definitions are about to begin. The semicolon is necessary because the lines which follow are the equation definitions, which do not begin with a keyword.

> WARNING--omission of the semicolon at the end of the equations list is one of the most common errors in GAMS. Identification of this error can be difficult, as it results in the generation of a lengthy and confusing list of error messages in the output.

The equations for the example problem (equation (5) above) consist of an objective function, and four constraint functions, all of which are less-than-or-equal-to constraints.

It is common in mathematical statements to make use of the Greek letter Sigma to denote summation. GAMS translates this symbol with the word "SUM". Since in computer code it is not possible to write the

index of summation below the word SUM, in the GAMS language it is written beside it, as the first term inside parentheses.[7] The objective function for the example problem would be written as follows:

```
SUM(J, X(J)*R(J)) =E= Z;
```

This is almost, but not quite, self-explanatory. The summation is over the index "J", and sums the product of X(J) and R(J). (Like most computer codes, GAMS uses the asterisk to denote multiplication). This sum is set equal to the variable "Z" using the weird-looking expression "=E=". This overdone equal-to expression is used only in structural equations. The more conventional "=" is used in data transformations and assignments, of which there are no examples in this problem. These features of GAMS are discussed in Section III.

Since they are all of the same form, the constraint equations are entered using a single expression for the entire group:

```
SUM(J, X(J)*A(I,J)) =L= B(I) ;
```

The constraints use the expression "=L=", which stands for "less than or equal to" in structural equations. (The remaining symbol is "=G="). Since there are four of these, one corresponding to each resource, the constraint equations are named and defined over the set "I".

The complete equations section of the GAMS input for the example problem is:

```
EQUATIONS
  REVENUE      OBJECTIVE FUNCTION
  SUPPLY(I)    RESOURCE CONSTRAINTS ;
  REVENUE..    SUM(J, X(J)*R(J)) =E= Z ;
  SUPPLY(I)..  SUM(J, X(J)*A(I,J)) =L= B(I) ;
```

Note the two dots after the equation names in the definition statements. These tell GAMS that the equation name is complete and the expression defining it is about to begin. Note also that, while the list of equation declarations ends with a semicolon, each equation definition statement also ends with a semicolon.

MODEL and SOLVE Statements

The model statement is used to name the model and to identify the equations which it includes. In the simple farm problem, there is only one model, but one need not look far to find examples of jobs in which more than one model is to be solved. An example might be where one wishes to solve the farm problem first using all constraints, and second using a subset of constraints. (This is discussed in Section II, in a subsection entitled "Multiple Models".)

---

[7] Indexed operators, of which "SUM" is an example, are discussed in more detail in Section III.

The structure of the MODEL statement is:

MODEL (identifier (text)) /equation names/;

The model statement for the example can be written:

MODEL FARMPRIM FARM PROBLEM PRIMAL /REVENUE, SUPPLY/ ;

or, when the model incorporates all of the equations in the GAMS job, this may also be written:

MODEL FARMPRIM FARM PROBLEM PRIMAL /ALL/ ;

Note that this statement also concludes with a semi-colon.

Next comes the SOLVE statement, which indicates
(a) the model to be solved,
(b) the solution procedure to be used,
(c) the direction of solution (max or min),[8] and
(d) the name of the objective variable.

The SOLVE statement for the example is:

SOLVE FARMPRIM USING LP MAXIMIZING Z;

The statement begins with the keyword SOLVE, followed by the name of the model to be solved. The solution procedure to be used is specified with the word USING, and the statement concludes with the direction and the objective variable.

A small amount of flexibility is permitted with the SOLVE statement. The statement above could have been written:

SOLVE FARMPRIM MAXIMIZING Z USING LP;

Thus, it is not necessary to remember whether the objective and direction precedes or follows the specification of the solution process.

The entire GAMS job file for the example problem is shown on the next page. For reference, name this file FARMLP.GMS.

SUBMITTING THE FILE

A GAMS job file can be submitted in default fashion, or by specifying any of the options available. Most of the option calls are useful only if large models are to be solved, and will be discussed later in the manual. Two other option calls (PW and PS) can be used to control the lengths of lines and pages respectively, in the output files. These are well documented in the GAMS.DOC file, section 9.

---

[8]Non-American users may note that the directions "minimizing" and "maximizing" have the American spelling (using a "z" as opposed to an "s").

```
SETS
  J ACTIVITIES
    /STD-WHT, OATS, NEW-WHT/
  I INPUTS
    /RES1*RES4/
PARAMETER R(J) REVENUE OF ACTIVITIES
  /STD-WHT   108.3
   OATS       66.36
   NEW-WHT   127.58/
PARAMETER B(I) AVAILABILITY OF INPUTS
  /RES1   12
   RES2    8
   RES3  400
   RES4   80/
TABLE A(I,J)  USE OF INPUTS PER ACTIVITY
         STD-WHT     OATS     NEW-WHT
  RES1      1          1         1
  RES2      1                    1
  RES3     30         20        40
  RES4      5          5         8
VARIABLES
 Z     NET REVENUE
 X(J)  ACTIVITY LEVELS
 POSITIVE VARIABLE X;
EQUATIONS
  REVENUE     OBJECTIVE FUNCTION
  SUPPLY(I)   RESOURCE CONSTRAINTS;
  REVENUE..   SUM(J, X(J)*R(J)) =E= Z;
  SUPPLY(I).. SUM(J, X(J)*A(I,J)) =L= B(I);
MODEL FARMPRIM   FARM PROBLEM PRIMAL  /ALL/ ;
SOLVE FARMPRIM USING LP MAXIMIZING Z;
```

If the file is to be submitted in the simplest manner, it must be named either with no filename extension (with a fileprefix only), or using the extension ".GMS" (in the form "<fileprefix>.GMS").

```
cd\gams205    (This accesses the GAMS system files)
gams <drive>:<fileprefix>
```

For example, assume that you have created the file FARMLP.GMS, and have saved it on a floppy disk. You now wish to submit it using an IBM PC or PC compatible machine. Assuming the computer designates the floppy drive as drive "a", the sequence would be:

```
cd/gams205
gams a:farmlp
```

The computer will now compile, and if error-free, execute the file FARMLP.GMS. Whether or not the file is error-free, a new file, FARMLP.LST, will be created and written to the drive from which the input file originated (in this example, this is the "a" drive). If there are errors in the file, a message will appear on the screen directing the user to examine the "<fileprefix>.LST" file for the string "****", which highlights any errors. When the computer has completed writing the .LST file, two beeps will sound, and the words "ALL DONE" will appear on the screen.

If the input file has been named using an extension other than ".gms", the entire filename must be included in the submission command. For example, had the file above been named FARMLP.IN, the command would be:

```
cd\gams205
gams a:farmlp.in
```

The output will again be written to a newly-created file named FARMLP.LST. The ".LST" file is created using the original fileprefix, irrespective of the extension.

OUTPUT

Output from GAMS/MINOS is separated into the output from the compiler and that from execution.

Compiler Output

There are two sections of compiler output: Echo Print and Reference Maps.

Echo Print . The first part of the GAMS compiler output is called the Echo Print. This simply retypes the GAMS commands, appending line numbers for further reference. It is here that any GAMS errors are highlighted. The Echo Print of the example problem is shown at right.

```
PAGE     1
G E N E R A L  A L G E B R A I C  M O D E L I N G  S Y S T E M
C O M P I L A T I O N


   1   SETS
   2     J ACTIVITIES
   3       /STD-WHT, OATS, NEW-WHT/
   4     I INPUTS
   5       /RES1*RES4/
   6     PARAMETER R(J) REVENUE OF ACTIVITIES
   7       /STD-WHT    108.3
   8        OATS        66.36
   9        NEW-WHT    127.58/
  10     PARAMETER B(I)  AVAILABILITY OF INPUTS
  11       /RES1   12
  12        RES2    8
  13        RES3  400
  14        RES4   80/
  15     TABLE A(I,J)  USE OF INPUTS PER ACTIVITY
  16             STD-WHT     OATS     NEW-WHT
  17     RES1      1          1         1
  18     RES2      1                    1
  19     RES3     30         20        40
  20     RES4      5          5         8
  21   VARIABLES
  22    Z   NET REVENUE
  23    X(J)  ACTIVITY LEVELS
  24    POSITIVE VARIABLE X;
  25   EQUATIONS
  26     REVENUE     OBJECTIVE FUNCTION
  27     SUPPLY(I)   RESOURCE CONSTRAINTS;
  28     REVENUE..   SUM(J, X(J)*R(J)) =E= Z;
  29     SUPPLY(I).. SUM(J, X(J)*A(I,J)) =L= B(I);
  30   MODEL FARMPRIM   FARM PROBLEM PRIMAL  /ALL/ ;
  31   SOLVE FARMPRIM USING LP MAXIMIZING Z;
```

Reference Maps. The reference map of symbols on the next page consists of (a) a cross referenced list of all symbols by location and use, alphabetically arranged, and (b) a list of identifiers by type, including any explanatory text.[9]

The reference map for the example problem is at the right. Some explanatory comments follow.

Alphabetically, the elements of the "A" matrix, here called simply "A", will be shown first. Under TYPE it is shown that "A" is a parameter. It is first introduced (declared) on line 15, where the "A" matrix is initialized. It is referenced on line 29, in the definition of the constraint equations.

The set "I" is introduced on line 4, referenced on lines 10 and 15 (as a domain of both B(I) and A(I,J)), and appears twice on line 29. It is in a controlling position on line 29, as it is the set over which the four SUPPLY equations are defined.

The set "J" is in a controlling position on lines 28 and 29; it is used as the index of summation in REVENUE and SUPPLY equations.

Variables "X" and "Z" are implicitly assigned (IMPL-ASN) values on line 31, through the SOLVE statement. Once the solution is complete, values for "X(J)" and "Z" are determined.

The last part of the Reference Map consists of a list of identifiers, with any explanatory text. This is simply a listing of symbol definitions, for the benefit of users of the output.

---

[9] The reference map and symbol listing may be suppressed, reducing the amount of output, by beginning the GAMS input file with:

```
$OFFSYMLIST OFFSYMXREF
```

The "$" symbol must be in the first column, and this line must be the first statement in the file.

PAGE 2
G E N E R A L A L G E B R A I C M O D E L I N G S Y S T E M
SYMBOL LISTING

SYMBOL	TYPE	REFERENCES				
A	PARAM	DECLARED	15	REF	29	
B	PARAM	DECLARED	10	REF	29	
FARMPRIM	MODEL	DECLARED	30	REF	31	
I	SET	DECLARED	4	REF	10	15
27						
		2*29	CONTROL	29		
J	SET	DECLARED	2	REF	6	15
23						
		2*28	2*29	CONTROL	28	29
R	PARAM	DECLARED	6	REF	28	
REVENUE	EQU	DECLARED	26	DEFINED	28	
SUPPLY	EQU	DECLARED	27	DEFINED	29	
X	VAR	DECLARED	23	IMPL-ASN	31	REF
24						
		28	29			
Z	VAR	DECLARED	22	IMPL-ASN	31	REF
28						
		31				

SETS

I	INPUTS
J	ACTIVITIES

PARAMETERS

A	USE OF INPUTS PER ACTIVITY
B	AVAILABILITY OF INPUTS
R	REVENUE OF ACTIVITIES

VARIABLES

X	ACTIVITY LEVELS
Z	NET REVENUE

EQUATIONS

REVENUE	OBJECTIVE FUNCTION
SUPPLY	RESOURCE CONSTRAINTS

MODELS

FARMPRIM	FARM PROBLEM PRIMAL

Execution Output

There are four sections to the execution output: Equation Listing; Column Listing; Model Statistics; Solution Report.

Equation Listing. This portion of the output lists in painstaking detail some of the equations described in compact notation in the GAMS input statements. This is extremely useful for checking that the equations the computer is using in the model are correct.

Equations are listed with all endogenous variables grouped on the left-hand-side, and parameters on the right. Thus, the objective function, which is entered in the input file as:

```
REVENUE..   SUM(J, X(J)*R(J)) =E= Z;
```

is listed as:

```
REVENUE..  - Z + 108.3*X(STD-WHT) + 66.36*X(OATS) +
127.58*X(NEW-WHT) =E= 0 ;
```

The right-hand-side is zero, since the objective function is written entirely in terms of endogenous variables.

Note that the second supply equation (corresponding to the RES2 row) contains only two terms. GAMS does not print terms which are equal to zero.

When listing equations which have been entered through sets, such as the SUPPLY equations here, GAMS will list only the first three, unless told otherwise. If you wish more or fewer equations listed, there is an option statement which can be included in the file which overrides the three-equation default. This statement should be entered before the SOLVE statement, and takes the form:

```
OPTION LIMROW = n;
```

If the user wishes to suppress the equation listing entirely, this can be done by setting "n" equal to zero, and entering:

```
OPTION LIMROW = 0;
```

```
PAGE     3
G E N E R A L  A L G E B R A I C  M O D E L I N G  S Y S T E M
EQUATION LISTING    SOLVE FARMPRIM USING LP FROM LINE 31


---- REVENUE      =E=  OBJECTIVE FUNCTION


REVENUE..  - Z + 108.3*X(STD-WHT) + 66.36*X(OATS) + 127.58*X(NEW-WHT)
=E= 0 ;


---- SUPPLY       =L=  RESOURCE CONSTRAINTS


SUPPLY(RES1)..  X(STD-WHT) + X(OATS) + X(NEW-WHT) =L= 12 ;


SUPPLY(RES2)..  X(STD-WHT) + X(NEW-WHT) =L= 8 ;


SUPPLY(RES3)..  30*X(STD-WHT) + 20*X(OATS) + 40*X(NEW-WHT) =L= 400 ;

REMAINING ENTRY SKIPPED
```

Column Listing. The column listing on the next page shows the coefficients for each variable, and indicates the rows in which these occur.

In the listing for the example, the variable "Z" occurs only in the objective function. Since it is a free variable, it has a lower limit (.LO) of minus infinity (-INF), an upper limit (.UP) of plus infinity (+INF), and, as no initial value is entered, takes an initial level (.L) of zero.

The "X(J)" variables, of which there are three, enter into the objective function and all four constraint rows. Since "X(J)" is declared to be POSITIVE, all "X(J)" variables have a lower bound (.LO) of zero, but, since no further restrictions are imposed, they have no finite upper bound. Again, since no initial values are introduced, each "X(J)" takes an initial value (.L) of zero.[10]

In this example, there are only three columns. In models containing more than three columns, as with equations GAMS will list only the first three. Again, this three-column default is overridden with an option statement, entered before the SOLVE statement, of the form:

```
OPTION LIMCOL = n;
```

As was the case with the equation listing, the entire column listing can be suppressed by entering:

```
OPTION LIMCOL = 0;
```

---

10 Note that GAMS assigns starting values of zero to all variables, unless told otherwise. This is fine in the LP solution procedure, but in the non-linear routines, it can cause fatal errors. More is said about this in Section IV.

```
PAGE      4
G E N E R A L  A L G E B R A I C  M O D E L I N G  S Y S T E M
COLUMN LISTING      SOLVE FARMPRIM USING LP FROM LINE 31


---- Z           NET REVENUE


Z
  .LO =          -INF , .L =              0, .UP =          +INF
       -1                 REVENUE


---- X           ACTIVITY LEVELS


X(STD-WHT)
  .LO =                0, .L =              0, .UP =          +INF
      108.3                 REVENUE
        1                   SUPPLY(RES1)
        1                   SUPPLY(RES2)
       30                   SUPPLY(RES3)
        5                   SUPPLY(RES4)

X(OATS)
  .LO =                0, .L =              0, .UP =          +INF
       66.36                REVENUE
        1                   SUPPLY(RES1)
       20                   SUPPLY(RES3)
        5                   SUPPLY(RES4)

X(NEW-WHT)
  .LO =                0, .L =              0, .UP =          +INF
      127.58                REVENUE
        1                   SUPPLY(RES1)
        1                   SUPPLY(RES2)
       40                   SUPPLY(RES3)
        8                   SUPPLY(RES4)
```

Model Statistics. Next, GAMS prints a summary of the model statistics.

In this case, there are five rows or equations in the problem; the objective function and the four constraint rows. These are entered using two statements; one for the objective and one for the block of constraints. Thus, there are two blocks of equations, and a total of five single equations.

Likewise, there are four columns associated with endogenous variables; the objective variable "Z", and the three activity variables "X(J)". These are entered using two statements; one for "Z" and one for the block of "X(J)"s. Thus, there are two blocks of variables, associated with four individual columns, or single variables.

In the problem, there are five equations, each of which can include coefficients on each of the four endogenous variables. There are therefore a total of 20 possible coefficients to be entered. The coefficient on "Z" in each of the constraints is zero, as is the coefficient on "X(OATS)" on the second constraint equation. Thus, there are 15 of 20 possible coefficients which are not equal to zero. This is reported in the last line of the MODEL STATISTICS listing.

```
PAGE     5
G E N E R A L  A L G E B R A I C  M O D E L I N G  S Y S T E M

MODEL STATISTICS    SOLVE FARMPRIM USING LP FROM LINE 31


MODEL STATISTICS

BLOCKS OF EQUATIONS     2     SINGLE EQUATIONS      5
BLOCKS OF VARIABLES     2     SINGLE VARIABLES      4
NON ZERO ELEMENTS      15

GENERATION TIME      =        0.040 MINUTES


EXECUTION TIME       =        0.091 MINUTES
```

Solution Report. Finally, we come to the answers to the problem (next page). The Solution Report consists of three parts; the Solve Summary, the results themselves, and the Report Summary.

First, we examine the Solve Summary for the example. The model which was solved was FARMPRIM. It was solved using BDM-LP, the linear programming solver. The objective variable is Z, which is maximized. While all this information can be read from line 31 of the Echo Print, it is reprinted here for the benefit of those who examine only the execution output.

Next, GAMS informs the user that the problem was solved, and that there is an optimal solution, the objective value of which is 1260.3733. The solver needed only a negligible number of resource units, of a possible 1000, and required two, of a possible 1000, iterations. The solver would have stopped, and reported non-optimal values if either the limit on resource units or the limit on iterations had been reached before an optimum was obtained. These resource and iteration limits can be altered entering the following option statements before the SOLVE statement:

```
OPTION RESLIM = n;      OPTION ITERLIM = m;   .
```

Following the message "EXIT -- OPTIMAL SOLUTION FOUND", the second part of the Solution Report, the tables of results themselves, are printed.

The LOWER and UPPER bounds on the equation values, and the optimal LEVELs of these in solution are reported under EQU SUPPLY. All available amounts of RES1, RES2, and RES4 are used, showing that those constraints are binding at the optimum. The third constraint is not binding, as is shown by the fact that less than all available RES3 is used in the optimal solution. The column MARGINAL shows the amounts by which the objective variable would change if the amount of each resource was increased by one unit. (These, of course, correspond to the dual variables.) The marginal value of RES3, which is not used to capacity, is zero, a comforting result in view of the complementary slackness theorems of linear programming. The slack variable associated with RES3 is included in the final basis.

The listing for EQU REVENUE is not very informative. It says that the left-hand-side of

```
-Z + SUM(J, X(J)*R(J)) =E= 0
```

has upper and lower bounds of zero, and consequently takes a level of zero as well. The marginal value says that, if the right-hand-side was increased by one unit, the variable "Z" would **decrease** by one unit (corresponding to a one-unit increase in "-Z").

```
PAGE      6
G E N E R A L  A L G E B R A I C  M O D E L I N G  S Y S T E M
SOLUTION REPORT     SOLVE FARMPRIM USING LP FROM LINE 31

                S O L V E     S U M M A R Y

     MODEL   FARMPRIM            OBJECTIVE  Z
     TYPE    LP                  DIRECTION  MAXIMIZE
     SOLVER  BDMLP               FROM LINE  31

**** SOLVER STATUS     1 NORMAL COMPLETION
**** MODEL STATUS      1 OPTIMAL
**** OBJECTIVE VALUE               1260.3733

 RESOURCE USAGE, LIMIT          0.000     1000.000
 ITERATION COUNT, LIMIT         2         1000
          . . .
 WORK SPACE NEEDED (ESTIMATE)   --     3910 WORDS.
 WORK SPACE AVAILABLE           --    42144 WORDS.

 EXIT -- OPTIMAL SOLUTION FOUND.

                       LOWER      LEVEL      UPPER     MARGINAL

---- EQU REVENUE         .          .          .        -1.000

  REVENUE     OBJECTIVE FUNCTION


---- EQU SUPPLY      RESOURCE CONSTRAINTS

        LOWER      LEVEL      UPPER     MARGINAL

RES1     -INF      12.000     12.000     34.227
RES2     -INF       8.000      8.000     41.940
RES3     -INF     386.667    400.000       .
RES4     -INF      80.000     80.000      6.427

                       LOWER      LEVEL      UPPER     MARGINAL

---- VAR Z               -INF    1260.373     +INF         .

  Z           NET REVENUE


---- VAR X           ACTIVITY LEVELS

           LOWER      LEVEL      UPPER     MARGINAL

STD-WHT      .         1.333      +INF         .
OATS         .         4.000      +INF         .
NEW-WHT      .         6.667      +INF         .

**** REPORT SUMMARY :        0     NONOPT
                             0 INFEASIBLE
                             0  UNBOUNDED
```

The values of the objective variable "Z", and the choice variables "X(J)" follow. Here, "Z" is shown to be a free variable, with a lower bound of minus infinity (-INF) and an upper bound of plus infinity (+INF). The choice variables X(J) are constrained to be non-negative, so have a lower bound of zero, but no upper bound, and the "Level Column" gives their value in the current solution. The "MARGINAL" column shows the Z_j-C_j value or the penalty costs, of bringing the corresponding activity into solution. Since a one-unit increase in "Z" costs exactly one unit of "Z", this marginal value is equal to zero. The marginal values for "X(J)" variables are potentially more interesting because they indicate the "cost" of deviating from the optimal solution. However, in this case they are all included in the final basis and their associated Z_j-C_j values, shown in the MARGINAL column, are all zero.

The third part of the solution report is the three-line "REPORT SUMMARY". In this file, there were no problems of infeasibility or unboundedness. Had either of these problems occurred, the number of non-optimal values of variables would have been reported. (This information is often of little use, since these are fatal errors!)

SECTION II

Extended Modelling Capacity of GAMS for Linear Programming

In this section, a number of extensions to the modelling capabilities of GAMS discussed in the tutorial are outlined. These extensions are useful in solving larger problems, and in manipulating and transforming data into the forms needed for the models. The capacity for manipulating data is among the most powerful features of GAMS. Although these extensions are discussed within the context of linear programming applications, they are important for non-linear and dynamic programming applications as well. Other special features are described in subsequent sections.

LARGE TABLES

In realistic programming problems, data tables may contain more columns than will fit on the width of one page or monitor screen. In these cases, the table may be extended simply by writing the extended portion under the first portion. For example, if the farm problem involved twelve activities and five constraint rows, the A(I,J) matrix could be entered as:

```
SETS
   J ACTIVITIES   /ACT1*ACT12/
   I INPUTS   /RES1*RES5/

TABLE A(I,J) USE OF INPUTS PER ACTIVITY
          ACT1  ACT2  ACT3  ACT4  ACT5  ACT6  ACT7  ACT8
RES1       xx    xx    xx    xx    xx    xx    xx    xx
RES2       xx    xx    xx    xx    xx    xx    xx    xx
RES3       xx    xx    xx    xx    xx    xx    xx    xx
RES4       xx    xx    xx    xx    xx    xx    xx    xx
RES5       xx    xx    xx    xx    xx    xx    xx    xx

 +        ACT9  ACT10  ACT11  ACT12
RES1       xx    xx     xx     xx
RES2       xx    xx     xx     xx
RES3       xx    xx     xx     xx
RES4       xx    xx     xx     xx
RES5       xx    xx     xx     xx
```

The table is extended by typing a plus ("+") in any column except the first one. The row labels must be retyped, but only for those rows which contain non-zero entries in that particular portion of the table. Since columns need not be entered in any particular order, one may save considerable effort by ordering data so as to group columns which contain only zero entries in some row(s).[11]

---

[11] Lengthy parameter lists can be entered in the same way as large, unordered sets. Since elements can be separated using commas or end-of-lines, a parameter list may be continued on successive lines.

MANIPULATION OF DATA USING ASSIGNMENT STATEMENTS

When formulating models in GAMS, it is possible to enter all parameters separately. However, this can entail much tedious calculation by the user prior to running a GAMS job. This effort can be considerably reduced by entering data in its most primitive form, and using assignment statements to generate more refined data from the original input. (This practice has the additional advantage of informing any other users of the model of any data transformations undertaken in the modelling process.)

As an example, consider a MOTAD version of the farm problem used in the tutorial.[12] In this version, it is assumed that the gross returns from the activities are not known with certainty but that there exists a time series of observations on gross margins for each activity. The objective of the problem is to minimize the sum of absolute deviations of annual revenue from the mean, subject to attaining some specified level of expected revenue. In formulating the problem, it is necessary to have calculated the mean revenue and the deviations from mean revenue for each of the activities.

The GAMS job file for an example in which there are five years of observations on gross margins is shown on subsequent pages.[13] An algebraic representation of the model is:

$$
\begin{aligned}
(6)\quad & \max.\ Y_1 + Y_2 + Y_3 + Y_4 + Y_5 \\
& \text{s.t.} \\
& X_1 + X_2 + X_3 \le 12 \\
& X_1 + X_3 \le 8 \\
& 30X_1 + 20X_2 + 40X_3 \le 400 \\
& 5X_1 + 5X_2 + 8X_3 \le 80 \\
& -8.5X_1 + 1.94X_2 - 14.88X_3 + Y_1 \le 0 \\
& 25.0X_1 + 64.04X_2 + 110.82X_3 + Y_2 \le 0 \\
& 34.4X_1 - 33.06X_2 - 33.68X_3 + Y_3 \le 0 \\
& 46.0X_1 + 8.04X_2 - 44.38X_3 + Y_4 \le 0 \\
& -96.9X_1 - 40.96X_2 - 17.88X_3 + Y_5 \le 0 \\
& 108.3X_1 + 66.36X_2 + 127.58X_3 \le \lambda \\
& X_1,\ldots,X_3 \ge 0;\ Y_1,\ldots,Y_5 \ge 0
\end{aligned}
$$

---

[12] See P. Hazell, "A Linear Alternative to Quadratic and Semivariance Programming for Farm Planning Under Uncertainty", American Journal of Agricultural Economics, 53(1971):53-62, for a discussion of MOTAD.

[13] The example is used by L. Tauer, "Target MOTAD", American Journal of Agricultural Economics, 65(1983):606-610, in this comparison of MOTAD with his target MOTAD formulation.

Rather than enter these data as is, we start with the raw data on gross margins and have GAMS calculate means and deviations.

The statements used to calculate the mean revenue and deviations from mean revenue for each activity are:

```
PARAMETER RBAR(J)  AVERAGE REVENUE PER ACTIVITY;
   RBAR(J) = SUM(T, R(T,J)) / (CARD(T)) ;
 DISPLAY RBAR;
PARAMETER D(T,J)  DEVIATIONS FROM MEAN REVENUES;
   D(T,J) = R(T,J)-RBAR(J);
 DISPLAY D;
```

Each parameter created is constructed in three lines, the third of which is an optional display of the result. The declaration is the same as for any parameter, but must end with a semicolon, as the following line does not begin with a keyword. (The term CARD(T) which indicates the number of elements in the set, is discussed in Section III.)

The assignment itself takes the form:[14]

```
parameter name(domain) = expression ;
```

The final line is the DISPLAY statement, which requests the values of the created parameter to be printed as part of the GAMS output. This is not required, but it is highly recommended, as it enables the user to confirm that the parameter is calculated correctly.

The output from the example above will now include:

```
----      15 PARAMETER RBAR        AVERAGE REVENUE PER ACTIVITY

STD-WHT 108.300,    OATS      66.360,    NEW-WHT 127.580


----      18 PARAMETER D           DEVIATIONS FROM MEAN REVENUES

            STD-WHT        OATS     NEW-WHT

Y1           -8.500       1.940     -14.880
Y2           25.000      64.040     110.820
Y3           34.400     -33.060     -33.680
Y4           46.000       8.040     -44.380
Y5          -96.900     -40.960     -17.880
```

The entire input file for the MOTAD problem is given on the next page. Other examples of parameter creation are found in the sample job files in Appendix B.

---

[14] Assignment statements are explained in greater detail in Section III.

```
*FARM PROBLEM, MOTAD FORMULATION
*
 SETS
     J ACTIVITIES
     /STD-WHT, OATS, NEW-WHT/
     I INPUTS
     /RES1*RES4/
     T TIME PERIODS
     /1*5/ ;
  PARAMETER B(I) AVAILABILITY OF INPUTS
     /RES1    12
      RES2     8
      RES3   400
      RES4    80/ ;
  TABLE A(I,J)  USE OF INPUTS PER ACTIVITY
          STD-WHT    OATS    NEW-WHT
  RES1        1         1        1
  RES2        1                  1
  RES3       30        20       40
  RES4        5         5        8 ;
   TABLE R(T,J)   REVENUES PER ACTIVITY PER TIME PERIOD
            STD-WHT    OATS    NEW-WHT
    1         99.8     68.3     112.7
    2        133.3    130.4     238.4
    3        142.7     33.3      93.9
    4        154.3     74.4      83.2
    5         11.4     25.4     109.7 ;
  PARAMETER RBAR(J)  AVERAGE REVENUE PER ACTIVITY;
     RBAR(J) = SUM(T, R(T,J)) / (CARD(T)) ;
   DISPLAY RBAR;
  PARAMETER D(T,J)  DEVIATIONS FROM MEAN REVENUES;
     D(T,J) = R(T,J)-RBAR(J);
   DISPLAY D;
 SCALAR XPINC  EXPECTED INCOME  /945/;
  VARIABLES   SAD     SUM OF INCOME DEVIATIONS
              X(J)    ACTIVITY LEVELS
              Y(T)  TIME COEFFICIENTS
              POSITIVE VARIABLES X,Y;
  EQUATIONS
     OBJMOTAD   OBJECTIVE FUNCTION
     SUPPLY(I)  RESOURCE CONSTRAINTS
     ANNUAL(T)  YEARLY NONNEGATIVITY CONSTRAINTS
     EXPREV     EXPECTED REVENUE;
   OBJMOTAD..   SUM(T, Y(T))  =E= SAD;
   SUPPLY(I)..  SUM(J, X(J)*A(I,J)) =L= B(I);
   ANNUAL(T)..  (SUM(J, X(J)*D(T,J))) + Y(T) =G= 0;
   EXPREV..     SUM(J, X(J)*RBAR(J)) =E= XPINC;
    MODEL MOTAD /ALL/ ;
   SOLVE MOTAD USING LP MINIMIZING SAD;
```

MULTIPLE MODELS

Some commercial linear programming software emphasizes options to run multiple models and perform "basis change" parametric analysis on the model's parameters. GAMS does not, and this is certainly one limitation to this software. However, GAMS does allow the analyst to solve several versions of a model in one job. This can be done by creating subsets of the activities set, and/or the constraint set, and defining equations over the subsets. For example, if we wished to solve the farm problem first subject to all four constraints, and second, subject to only the first three, the GAMS input could look like:

```
SETS
  J ACTIVITIES
    /STD-WHT, OATS, NEW-WHT/
  I ALL INPUTS
    /RES1*RES4/
  I3(I) FIRST THREE INPUTS;
I3(I) = YES$(ORD(I) LE 3);
PARAMETER R(J)   REVENUE OF ACTIVITIES
     .                    .
     .                    .
     .                    .
POSITIVE VARIABLE X;
EQUATIONS
  REVENUE     OBJECTIVE FUNCTION
  SUPPLY(I)   ALL RESOURCE CONSTRAINTS
  SUPPLY3(I3) THREE RESOURCE CONSTRAINTS;
  REVENUE..   SUM(J, X(J)*R(J) =E= Z;
  SUPPLY(I).. SUM(J, X(J)*A(I,J)) =L= B(I);
  SUPPLY3(I3).. SUM(J, X(J)*A(I3,J)) =L= B(I3);
MODEL FARMALL FARM PROBLEM GLOBAL  /REVENUE, SUPPLY/ ;
MODEL FARM3  FARM PROBLEM 3 CONSTRAINTS  /REVENUE, SUPPLY3/ ;
SOLVE FARMALL USING LP MAXIMIZING Z;
SOLVE FARM3 USING LP MAXIMIZING Z;
```

The subset of "I" is created in two steps. First, it is declared in the SETS section. Second, it is defined by an assignment statement. The assignment statement says that an element of "I" is also an element of "I3" if the ordinal position of the element of "I" is less than or equal to 3. The symbol "ORD" and the "$" sign are discussed further in Section III.

Note that two sets of equations need to be declared and defined; one for each set of constraints. Equations defined on different sets must have different names, which is why one set is called SUPPLY, and the other SUPPLY3.

Subsets of activities can be defined and used in similar fashion. A general discussion of subsets and supersets is contained in Section III.

RE-SOLVING MODELS WITH DIFFERENT PARAMETER VALUES

Another type of multiple run is the case where the modeller wishes to solve the model using two or more sets of parameter values. This is done through the use of a sequence of assignment and SOLVE statements. As an example, consider solving the farm problem as it is originally written in the tutorial, then changing the revenue coefficients in the objective function and re-solving subject to the same set of constraints.

Following the original SOLVE statement, the revenue coefficients can be changed by assigning new values to the parameter "R(J)":[15]

```
R("STD-WHT") = 85;
R("OATS") = 70;
R("NEW-WHT") = 120;
```

The "MODEL" remains the same, containing a redefined "REVENUE" equation and an unchanged set of constraint equations. Thus, only an additional SOLVE statement is needed, which is a duplicate of the first:

```
SOLVE FARMPRIM USING LP MAXIMIZING Z;
```

When the augmented job file is submitted, the resulting output file will contain the compiler output (Echo Print and Reference Lists) for the entire job, as well as two sets of execution output.

When models, or variations of a model, are solved sequentially, the values resulting from the solution of one model are carried forward as a starting basis for the next model. This feature of GAMS allows the second (and subsequent) variations of a model to be solved in less time and using fewer iterations than the first.

Sequences of models may also be constructed and submitted using separate GAMS job files. This is useful in cases where the solution to one model raises questions which can be answered through solving one or more variations on the original model. This procedure is illustrated using a second variation of the farm problem.

The original file, FARMLP.GMS, contained only one SOLVE statement. Rather than using the default submission, "GAMS FARMLP", the researcher should submit it using the option "SAVE". The submission command (assuming that the original input file is stored on drive "a") would be:

```
GAMS a:FARMLP SAVE=FARMWORK, or
GAMS a:FARMLP S=FARMWORK
```

The original file will be executed as before, and the output written to FARMLP.LST. In addition, however, eight intermediate files, to be

---

[15] Note that single element labels are enclosed within quotation marks. This distinguishes them from set identifiers, which are entered without quotes.

named FARMWORK.G01 through FARMWORK.G08, will be stored on the drive from which FARMLP.GMS was originally read. These eight files are referred to as the restart files. Restart files are named using the fileprefix specified in the "SAVE =" optional statement.

The user now wishes to solve a variation of the model incorporating new objective function coefficients. Rather than appending the changes to the original file and re-solving the first model as well as the second, a new file, containing only the new assignments and the second SOLVE statement can be constructed and submitted. For reference, call this file FARMLP1.GMS.

The second file is submitted using the command:

```
GAMS a:FARMLP1  RESTART=FARMWORK, or
GAMS a:FARMLP1  R=FARMWORK
```

This command will result in the new input file being compiled by GAMS, and the new compilation added to that of FARMLP.GMS, which already exists in the eight temporary restart files. The new SOLVE statement will then be executed, and a new output file, called FARMLP1.LST, will be written to the specified drive. This new file will contain an Echo Print and Symbols List of only the additional lines of input contained in FARMLP1.GMS, as well as the complete execution output of the second model.

In order to prevent the output file of the original job being overwritten, it is important to give any continuation file a different fileprefix than was used for the original file. For example, if the original file was named FARMLP.GMS, and the continuation file is named FARMLP.NEW (which DOS will recognize as separate files), the execution of FARMLP.NEW will result in the creation of a new FARMLP.LST file, which will overwrite the first FARMLP.LST file.[16]

STARTING WITH A USER-ENTERED BASIS

The above procedure for using the results of an early version of a model to solve a later version requires the original model to be solved in GAMS, and that the restart files be saved. In cases where a large model has been solved outside GAMS, or when restart files are not available, the solution to the first model can be incorporated into a

---

[16] This problem may also be avoided by specifying the name of the file to which output is to be written. This is done with the optional statement "O = <filename>". A complete path may be specified using this statement, allowing the user to write the output file to a different drive than that from which the input file is read. For example, to execute the tutorial file, FARMLP.GMS, which is stored in a subdirectory called "AGEC712" in drive "c", and write the output to a file called FARMLP.OUT in the root directory on drive "a", the command sequence is:

```
GAMS C:\AGEC712\FARMLP  O=A:FARMLP.OUT
```

new GAMS input file as a starting basis for a different version of the model. This is done by assigning starting values of the variables (and equations) in the model.

For example, if a user wished to solve a variation on the farm problem from the tutorial, having previously solved the original problem, the optimal values from the first model can be entered with the following assignment statements:

```
X.L("STD-WHT") = 1.333;
X.L("OATS")  =   4.000;
X.L("NEW-WHT") = 6.667;
X.M(J) = 0;
SUPPLY.M("RES1") = 34.227;
SUPPLY.M("RES2") = 41.940;
SUPPLY.M("RES3") = 0;
SUPPLY.M("RES4") = 6.427;
SUPPLY.L("RES1") = 12;
SUPPLY.L("RES2") = 8;
SUPPLY.L("RES3") = 386.667;
SUPPLY.L("RES4") = 80;
```

The identifier extension ".L" indicates that the **level** of the variable is assigned. The identifier extension ".M" refers to the marginal value (in LP problems, these are the Z_j-C_j values for the columns of a tableau, and dual variables for the rows). Note that in this case, we have assigned starting values to equations. This is to illustrate the fact that this can be done, but for purposes of restarting a modified problem with a previous optimal basis, only the starting values for the variables need to be supplied.[17,18]

---

[17] The farm problem is small and easy to solve, so it is a poor example of the benefits to be had from entering a starting basis. In large problems, it may take several hundred iterations to reach the frontier of the constraint set from a cold start. The amount of computation can be significantly reduced by entering a starting basis.

[18] In linear programming problems, the assignment of starting values potentially reduces the amount of computation, but is not strictly necessary. In constrast, starting values are essential when solving non-linear programming problems. A more complete discussion of this is found in Section IV.

SECTION III

Special Features

There are a number of special features and operators in GAMS which facilitate modelling of large and complex systems. Some of these were used in the examples in Section II. More complete discussions of the most important GAMS features are contained in this section.

THE "$" OPERATOR

The "$" symbol has two purposes in GAMS. When it is placed in the first column of a line, it forms part of an output-control statement. For example, the statement

```
$OFFSYMLIST OFFSYMXREF
```

results in the suppression of the symbols list and reference map in the output file. The other, more important function, is that of an "if-then-else" or "such-that" operator, in expressions of the form

```
(expression)$(condition)
```

This sentence represents something which is equal to the (expression) if and only if the condition is met. (An example of how this is used was seen in the expression for creating a subset of constraints for an LP problem in Section II.) In general, the condition is some relation, such as "A greater than B" (written in GAMS as "A GT B").[19] However, a short form exists for the condition "$(<identifier> NE 0)" (read as "not equal to zero"), which is simply "$<identifier>". Thus, the condition "$(B(J) NE 0)" can be written more compactly as "$B(J)". For example, if a parameter is to be created which is the ratio of two previously-defined parameters, it will be undefined for zero values of the denominator. The new parameter can be safely defined as:

```
PARAMETER RATIO(J);
  RATIO(J) = A(J)/B(J)$B(J);
```

The parameter "RATIO" will now be defined only for those cases in which "B(J)" exists and is not equal to zero.

Conditions are well-defined **if an only if** they are written in terms of identifiers the values of which have been previously assigned. Thus, endogenous variables may not be used in condition statements for data input. However, the optimal values of variables from one model, which have fixed values through the execution of a SOLVE statement, can

---

[19] Relational operators are discussed in a note to the subsection below dealing with assignments and are listed in Appendix "A".

be used as parameter values for later models. These optimal values are referenced using the identifier extension ".L", to distinguish them from undetermined variables, which carry no identifier extension.[20]

In certain non-linear models, some of the structural equations may be undefined if a variable takes on certain "illegal" values. Examples are (a) division by a variable, (b) the square root of a variable, and (c) the logarithm of a variable. Since conditions may not be written in terms of unassigned variables, the "$" operator may not be used to rule out the illegal values. In such cases, the only solution is to impose bounds on the variables in question. This is an issue peculiar to non-linear models and discussion of it is deferred until Section IV.

As the "$" symbol is used in numerous instances, examples are presented throughout this section, rather than being grouped here.

IDENTICAL SETS: THE ALIAS STATEMENT

In some models, notably those incorporating quadratic forms, a set may be used more than once as an index in an equation. In these cases, it is necessary to have more than one name (identifier) for the set. A set can be given several different identifiers by means of the ALIAS statement.

The ALIAS statement is commonly placed in the SETS portion of the GAMS job file, after the set(s) to be renamed have been declared and defined. It takes the form:

```
ALIAS ((oldname),(newname1),(newname2),...,(newnameN));
```

After this statement is entered, the set in question may be referenced using any of its identifiers.

An example is found in the mean-variance (or E.V.) formulation of the farm problem (Appendix B, B.4). The objective function in this model is quadratic in "X(J)". Thus, it is necessary to have an identifier for the set "J", and another for the set "J-prime", even though these are the same set. This is achieved by writing

```
ALIAS (J,JP);
```

after the sets for the model have been declared.

The word ALIAS is, itself, a keyword in GAMS, and can be written

---

[20]Unfortunately, the identifier extension ".L" is used to refer both to starting values and optimal values of variables. The potential for confusion is eliminated, however, when it is recognized that, in both cases, it is the level of the variable which is denoted by "<identifier>.L". While the level can be assigned any value the modeller wishes before a SOLVE statement is executed, it becomes a definite (and, hopefully, unique) number after the model has been solved.

anywhere in the file. If more sets are to be entered following the ALIAS statement, it is necessary to to rewrite the keyword SET(S) prior to entering them.

"ORD" AND "CARD"

The symbol "ORD" is used to refer to the ordinal position of an element in a set list. The symbol "CARD" refers to the number of elements in the set.

"ORD" can be used in the creation of subsets, described below, and in assignments and equation specifications. One important example is that of dynamic programs in which discounted revenues are to be summed. In these cases, each period's revenue is multiplied by a discount factor raised to the power "t" (for "time"). The discount factor for each period may be constructed as:

```
SET
T TIME PERIODS /1*10/
 SCALAR DELTA DISCOUNT RATE /0.1/
 SCALAR DISC DISCOUNT;
 DISC = 1/(1+DELTA);
 DISPLAY DISC;
 PARAMETER RHO(T) DISCOUNT FACTOR;
 RHO(T) = DISC**(ORD(T));
 DISPLAY RHO;
```

The objective function may then be formulated as:

```
Z =E= SUM(T, RHO(T)*(expression);
```
[21]

Applications of the symbol "CARD" are found in computations of the mean and variance of a sample or population of data. For example, if we are interested in the mean and variance of a sample of five annual revenues of a product, the GAMS representation could appear as:

```
SETS
  T TIME PERIODS  /1*5/
PARAMETER R(T) REVENUE
  /1  10, 2  15, 3  40, 4  21, 5  34/
PARAMETER RBAR  AVERAGE REVENUE;
  RBAR = SUM(T, R(T))/CARD(T);
 DISPLAY RBAR;
```

---

[21] This specification would be used when the objective function involves summation over the periods (1,2,...,T). In the mine manager problem (Appendix B, B.7), the summation is from 0 to T. In this case, "RHO" is computed as

```
RHO(T) = DISC**(ORD(T)-1);
```

since the first position of the set "T" (where "ORD(T)" = 1) is held by time period "0".

```
PARAMETER D(T)  DEVIATIONS FROM AVERAGE REVENUE;
  D(T) = R(T)-RBAR;
 DISPLAY D;
PARAMETER V  SAMPLE VARIANCE;
  V = SUM(T, SQR(D(T))/CARD(T)-1;
 DISPLAY V;
```

(Note that CARD(T) is a scalar quantity which can be used in arithmetic expressions like any other number.)

Similar GAMS statements are used to compute the sample covariance matrix in the mean-variance formulation of the farm problem. (See Appendix B for the entire job file.)

Sets created using the ALIAS statement can appear in the domain of "ORD", but not in the domain of "CARD". This is an annoying detail, but not a serious problem, since aliased sets must have the same number of elements. An example is provided in the example file B.8 in Appendix B.

First, the aliased sets "J" and "K" are created from the original set "I". Later, in the creation of the parameter "AP", a condition is imposed:

```
AP(I,J)$(ORD(I)LT CARD(I) AND ORD(J)LT CARD(I))
```

Since "I" is the original set, it may appear in the domain of either "ORD" or "CARD". The set "J", however, cannot appear in the domain of "CARD", although it may appear in the domain of "ORD". Thus, the condition "ORD(J) LT CARD(J)" must be written as "ORD(J) LT CARD(I)".

ASSIGNMENTS[22]

Assignment statements are used to generate and/or modify sets and parameters. The structure of an assignment is:

Left-hand-side		Right-hand-side
Identifier(domain)$(condition)	=	expression

When an assignment is used to modify existing values, the procedure takes place in two steps. First, all values in the domain of the identifier on the left-hand-side are set to zero. Second, the right-hand-side expression, evaluated over the same domain is inserted.[23] Note that this expression may also contain a condition.

---

[22] This discussion is adapted from Meeraus, GAMS: Preliminary User's Guide (footnote 3).

[23] The relations used in "conditions" are less-than (LT), less-than-or-equal-to (LE), greater-than (GT), greater-than-or-equal-to (GE), equal-to (EQ), and not-equal-to (NE). For a complete list of relations, see Appendix A.

To illustrate, consider a parameter "R", which is defined over the sets "I" and "J", each of which contains three elements. We wish to construct a matrix "R(I,J)", in which the diagonal elements are equal to 2, and the off-diagonal elements are equal to 1. The statements used would be:

```
R(I,J) = 1;    , which yields:

        J1   J2   J3
I1       1    1    1
I2       1    1    1
I3       1    1    1   , followed by:

R(I,J)$(ORD(I) EQ ORD(J)) = 2; , resulting in the matrix:

        J1   J2   J3
I1       2    1    1
I2       1    2    1
I3       1    1    2  .
```

Here, only those values meeting the condition on the left-hand-side of the assignment are set to zero and subsequently replaced.

If, instead, the following assignment had been entered,

```
R(I,J) = 2$(ORD(I) EQ ORD(J)); ,
```

a different matrix would have been generated:

```
        J1   J2   J3
I1       2    0    0
I2       0    2    0
I3       0    0    2  .
```

In this case, values of "R(I,J)" would have been set to zero; only those meeting the condition on the right-hand-side would have been replaced.

An example of a complex assignment is found in the Mean-Gini formulation of the farm problem (see Appendix B.3).[24] There, we wish to compute, for each activity, the difference between the activity's revenue in one period and its revenue in every other period. However, once we have computed the difference between the revenue of an activity in period 1 and the corresponding revenue in period 3, we do not wish to recompute the difference between revenue in period 3 and revenue in period 1. In mathematical notation, we want to compute:

---

[24] The mean-Gini decision criteria was suggested by S. Yitzhaki,"Stochastic Dominance, Mean-Variance and Gini's Mean Difference", American Economic Review, 72(1982):178-185. The mean-Gini programming formulation is found in J. Okunev and J. Dillon, "A Linear Programming Algorithm for Determining Mean-Gini Efficient Farm Plans", Agricultural Economics, 2(1989).

$(R_{ij} - R_{kj})$, for all j, $j \in J$, $i,k \in T$, $k > i$.

The assignment statement for this can be written as:[25]

```
RIJK(T,TP,J) = (R(T,J)-R(TP,J))$(ORD(T) LT ORD(TP));
```

Note that the first expression on the right-hand-side is enclosed in parentheses. These are extremely important. If they are omitted, and the assignment statement is written

```
RIJK(T,TP,J) = R(T,J)-R(TP,J)$(ORD(T) LT ORD(TP));  ,
```

a different assignment is created.

In the correct version, there will be values generated for "RIJK" only when k > i, creating a total of ten values for each activity in this particular example. The entries will be:

```
R(1,J)-R(2,J)
R(1,J)-R(3,J)   R(2,J)-R(3,J)
R(1,J)-R(4,J)   R(2,J)-R(4,J)   R(3,J)-R(4,J)
R(1,J)-R(5,J)   R(2,J)-R(5,J)   R(3,J)-R(4,J)   R(4,J)-R(5,J)
```

With the incorrect assignment, omitting the parentheses around the expression "R(T,J)-R(TP,J)", a square matrix of order five will be generated for each activity. These matrices will contain the following entries:

```
R(1,J) - 0      R(2,J) - 0          . . .       R(5,J) - 0
R(1,J)-R(2,J)   R(2,J) - 0                            .
R(1,J)-R(3,J)   R(2,J)-R(3,J)                         .
R(1,J)-R(4,J)   R(2,J)-R(4,J)                         .
R(1,J)-R(5,J)   R(2,J)-R(5,J)       . . .       R(5,J) - 0
```

The incorrect assignment results in "RIJK" being set equal to R(T,J) when the condition is <u>not</u> met, and equal to R(T,J)-R(TP,J) when the condition is met.

This is an illustration of the role of the "$" symbol as an "if-then-else" statement. The incorrect assignment statement is effectively saying:

$$RIJK = R(T,J) - \begin{cases} R(TP,J) \text{ if the condition is met} \\ 0 \quad \text{otherwise.} \end{cases}$$

---

[25] Equivalently,

```
RIJK(T,TP,J)$(ORD(T) LT ORD(TP)) = R(T,J)-R(TP,J);  .
```

The two are equivalent, as there are no pre-existing values for "RIJK".

SET OPERATIONS

GAMS allows the user to perform several operations on sets. These include the creation of supersets and subsets, and the operations of union, intersection, and difference. Only the creation of subsets is described here, since these are the most commonly-used modified sets.

Subsets are created using the symbols "YES" and "$". A subset is a collection of elements of a larger set which meet some specified condition. An example from the Conrad-Clark mine-manager problem (Appendix B, B.7) is the subset of time during which extraction takes place. This is defined to be all periods except the terminal period as:

```
SETS
 T TIME PERIODS    /1*10/
 TE(T)  EXTRACTION PERIODS;
TE(T) = YES$(ORD(T) LT CARD(T));
```

The last line specifies that an element of "T" is also an element of "TE" if the condition "ORD(T) LT CARD(T)" is met."

Sets whose elements are entered in set lists are called "constant" sets. In contrast, sets which are created through modifying existing sets (creating a subset is an example of this modification) are called "non-constant" sets. All operations are permitted on constant sets. In contrast, some operations dealing with domains are not defined on non-constant sets. Specifically, the symbols "ORD" and "CARD", operations with leads and lags, and domain checking are all meaningless when dealing with non-constant sets.

MATRIX OPERATIONS

The GAMS language does not include built-in matrix operators. However, most of the operations on matrices which are used in mathematical programming applications can be undertaken using scalar operations on individual matrix elements.

Addition and subtraction of matrices can be performed using simple assignments. To illustrate, assume a user wishes to construct a matrix "C" which is equal to the sum of the matrices (strictly speaking, parameter tables) "A" and "B". This is done by initializing a two-dimensional parameter "C" and assigning to it values which equal, element by element, the elements of "A" plus the elements of "B":

```
PARAMETER C(I,J)  SUM OF A AND B;
C(I,J) = A(I,J) + B(I,J);
```

(Subtraction is accomplished by replacing the plus sign with a minus sign.)

The left-hand-side of the assignment states that the parameter "C" has "CARD(I)" rows and "CARD(J)" columns, as do the parameters "A" and

"B". Each element of "C" is then defined to be equal to the sum of corresponding elements of "A" and "B".

Matrix multiplication is carried out in similar fashion, by assigning values to a newly-created parameter table using the indexed operator "SUM". Since this is an indexed **scalar** operator, the usual rules concerning conformability of matrices for multiplication are modified somewhat.

In general, the matrix product "AB" is defined if and only if the number of columns of matrix "A" is equal to the number of rows of matrix "B". If "A" is (I x J) and "B" is (J x K), the product "AB" is defined, but the product "BA" is not.

In GAMS, two matrices can be multiplied provided that they share a common domain. This common domain can be thought of as the number of columns of the pre-multiplying matrix and the number of rows of the post-multiplying matrix. The statements creating the matrix product "AB(I,K)" as the product of the two matrices "A(I,J)" and B(J,K) are:

```
PARAMETER AB(I,K)  PRODUCT OF A AND B;
AB(I,K) = SUM(J, A(I,J)*B(J,K));
```

The assignment statement assigns a value for each element of "AB", to which we can give the generic name AB_{ik}. The right-hand-side of the assignment specifies that AB_{ik} is equal to the sum over the index "J" of the products of the elements of row "i" from matrix "A" and the elements of column "k" from the matrix "B".

The dimensions of the matrix "AB" are specified on the left-hand-side of the assignment, and are invariant to the order in which the elements of the multiplication are listed. Thus, it is possible to generate the matrix "AB" with this equivalent assignment statement:

```
AB(I,K) = SUM(J, B(J,K)*A(I,J));
```

since the right-hand-side contains an indexed sum of terms generated through **scalar** multiplication.

Note that the index of summation must be a domain common to both matrices.

The transpose of a matrix may be obtained using a single assignment as well. To obtain the transpose of "A(I,J)", a new parameter is initialized and set equal to the transpose of "A(I,J)", as follows:

```
PARAMETER AT(J,I)  TRANSPOSE OF A;
AT(J,I) = A(I,J);
```

Each element of "A" will now be assigned to its corresponding position in "AT".

In mathematical programming, the matrices are transposed in order to ensure conformability for multiplication. In GAMS, this step is not necessary. Two matrices of equal column (or row) dimension may be

multiplied directly, as if one of them had been transposed first. For example, in the mean-variance (E.V.) version of the farm problem (Appendix B, B.4), a covariance matrix is created which is equal to the parameter "D(T,J)" premultiplied by its transpose, and divided by "n-1". where "n" is the sample size. The parameter "D" in the example has three columns and five rows, so its transpose would have five columns and three rows. The matrix "D'D" will, therefore, be a square matrix of order three. The covariance matrix is initialized as a parameter "V(J,JP)", where "JP" is an alias of "J", and defined with the following statement:

```
PARAMETER V(J,JP) COVARIANCE MATRIX OF REVENUES;
V(J,JP) = SUM(T, D(T,J)*D(T,JP))/CARD(T)-1;
```

CONDITIONAL EQUATION SPECIFICATION

Structural equations may be specified subject to conditions, using the same type of expression found in conditional assignments. This can be illustrated using the input-output file (Appendix B, B.8), parts of which are reproduced here.

In this file, we wish to compute two $(I-A)^{-1}$ matrices, one of which applies to a model with the household sector endogenous, and the other to a model with exogenous households. Since the second model uses the same transactions matrix as the first, minus one row and column, we may use the same data, and simply specify a restricted set of equations.

```
SETS
 I SECTORS /AGR,MFG,TRNS,SVC,HH/
 ALIAS (I,J,K);
TABLE T(I,J) TRANSACTIONS TABLE HH ENDOGENOUS
     .
     .
     .
PARAMETER G(J) GROSS OUTPUT HOUSEHOLDS ENDOGENOUS
```

We begin by creating the "A" matrix for the endogenous model, called "A(I,J)". Next, we create an identity matrix of the same order, Following this, we create the smaller "A" and "I" matrices for the exogenous model, using conditional assignment statements.

```
PARAMETER A(I,J) DIRECT REQUIREMENTS HH ENDOGENOUS;
 A(I,J) = T(I,J)/G(J);
PARAMETER ID(I,K)  IDENTITY MATRIX ORDER CARD(I);
ID(I,K) = 1$(ORD(I) EQ ORD(K));
PARAMETER AP(I,J) DIRECT REQUIREMENTS HH EXOGENOUS;
 AP(I,J)$(ORD(I)LT CARD(I) AND ORD(J)LT CARD(I))
              = A(I,J);
PARAMETER IDP(I,K) IDENTITY MATRIX ORDER CARD(I)-1;
IDP(I,K)$(ORD(I)LT CARD(I)) =
         1$(ORD(I) EQ ORD(K));
```

The same logic is used in formulating the conditional statements defining the equation for the model with households exogenous.

```
 EQUATIONS...

  MATEND(I,K)    MATRIX CONSTRAINT HH ENDOGENOUS
  MATEX(I,K)     MATRIX CONSTRAINT HH EXOGENOUS;
      .
      .
      .
  MATEND(I,K)..   SUM(J,(ID(I,J)-A(I,J))*XEND(J,K)) =E= ID(I,K);
MATEX(I,K)$(ORD(I)LT CARD(I) AND ORD(K)LT CARD(I))..
     SUM(J,(IDP(I,J)-AP(I,J))*XEX(J,K)) =E= IDP(I,K);
```

INDEXED OPERATORS

GAMS has four indexed operators which can be used to write lengthy expressions in compact notation. These are "SUM", "PROD", "SMIN" and "SMAX". The general structure of statements using these is:

```
operator((indexl,...,indexN)$(condition), expression);
```

The indexed operator "SUM" was used in the linear programming formulation of the farm problem discussed in the tutorial in Section I. The symbol "SUM" is equivalent to the capital letter "Sigma" in conventional notation. "PROD" is the GAMS representation of the capital letter "Pi", used to denote the product of several terms. The operators "SMIN" and "SMAX" denote the minimum and maximum of an indexed collection of terms, respectively.[26]

Indexed operators may be defined over one or more one-dimensional sets. When only one set is used as the index, its identifier need not be enclosed within separate parentheses. For example, the constraint equations in the linear programming formulation of the farm problem are written:

```
SUPPLY(I)..  SUM(J, X(J)*A(I,J)) =L= B(I); .
```

When more than one set is used as an index, however, the collection of indices is enclosed within nested parentheses. An example comes from the transportation model (Appendix B, B.5), in which the objective function is written as:

```
COST..  SUM((I,J), C(I,J)*XT(I,J)) =E= TRCOST; .
```

Here, total cost is defined as the sum of the costs of all shipments from region "I" to region "J", hence the double summation index.

---

[26] Do not confuse the indexed operators "SMIN" and "SMAX" with the non-indexed operators "MIN" and "MAX". MIN(arg1,arg2,...,argN) returns the minimum of the group of arguments contained in parentheses. In contrast, "SMIN" returns the minimum of the several terms indexed by the controlling set identifier.

An operator's index may be unrestricted, as in the two examples above, or subject to conditions. Conditions are imposed, using the "$" operator, on the index itself. For example, in the objective function of the Mean-Gini formulation of the farm problem (Appendix B, B.3), we wish to sum the variables "YPLUS(T,TP)" and "YMINUS(T,TP)", but only those values for which "ORD(T)" is strictly less than "ORD(TP)" are to be included. This condition is appended to the indices of both sums. The objective function is written as:

```
   OBJMGINI..  SUM((T,TP)$(ORD(T) LT ORD(TP)), YPLUS(T,TP))
  + SUM((T,TP)$(ORD(T) LT ORD(TP)), YMINUS(T,TP)) =E= MG; .
```

The summation is defined over those element-pairs from the sets "T" and "TP" such that the condition holds.

SECTION IV

Non-Linear Programming

Although the solution of non-linear programming problems in GAMS involves many of the features used in linear programming, there are a number of aspects worthy of separate mention. These are discussed in this section.

As was the case with linear programming, the discussion of non-linear programming begins with an extended example. Following the discussion of static non-linear problems, some of the special features of dynamic programming problems, such as leads and lags, are discussed.

The problem to be solved is one which should be familiar to most students of microeconomic theory. Following Koopmans[27], it is cast in the form of a single agent general equilibrium problem, in which Robinson Crusoe (RC) must allocate his capital and labour among the production of fish and coconuts in order to attain maximum utility. Mathematically, this becomes a problem of maximizing a non-linear utility function subject to a non-linear opportunity set. Thus, both non-linearity of the objective function, and non-linearity of the constraint set are included.[28]

Specifically, RC's consumption of fish and coconuts is given by F_c and C_c, respectively, and welfare, w, is measured according to $w = F_c^{.25} C_c^{.75}$. Capital and labour are in fixed supply at 100 units each. Fish, (F_P) and coconuts, (C_P) are produced according to:

$$F_P = K_F^{.3333} L_F^{.6667}$$
$$C_P = K_c^{.6667} L_c^{.3333}$$

RC's problem is to maximize his welfare subject to the resource constraints and the constraints that production of fish and coconuts must be greater than or equal to consumption. Mathematically this becomes:

$$\max.\ w = F_c^{.25} C_c^{.75}$$
s.t.
$$F_P = K_F^{.3333} L_F^{.6667} \geq F_c$$
$$C_P = K_c^{.6667} L_c^{.3333} \geq C_c$$
$$K_F + K_c \leq 100$$
$$L_F + L_c \leq 100$$

---

27 T. Koopmans, Three Essays in the State of Economic Science, McGraw-Hill, New York, 1957.

28 For simplicity, the utility and production functions are all Cobb-Douglas. This guarantees the existence of a unique interior solution.

The GAMS job file for this problem is shown below. It is constructed in a fashion similar to that of a job file for a linear programming problem, but differs in a few significant aspects.

First, in the SOLVE statement, the solution process used is "NLP" rather than "LP". This is a trivial-looking difference in the input file itself. However, "NLP" is a very different solver than "LP". It involves the use of two FORTRAN subroutines, called FUNOBJ and FUNCON, for handling non-linear objective and constraint functions, respectively. In the case of a non-linear objective function, the variables are partitioned into basic, non-basic, and "superbasic". Whenever non-linear constraints

```
SETS
 G GOODS  /FISH,COCONUTS/
 I INPUTS  /CAPITAL,LABOUR/
TABLE ELAS(I,G)  OUTPUT ELASTICITIES
            FISH  COCONUTS
 CAPITAL  0.3333  0.6667
 LABOUR   0.6667  0.3333
PARAMETER END(I)  FACTOR ENDOWMENTS
 /CAPITAL  100
  LABOUR   100/
PARAMETER SHARE(G)  EXPENDITURE SHARES
 /FISH      0.25
  COCONUTS  0.75/
VARIABLES  C(G)  FINAL CONSUMPTIONS
           F(I,G)  FACTOR USE
           BIGW  WELFARE
POSITIVE VARIABLES C,F;
EQUATIONS
 UTILITY  UTILITY FUNCTION
 OUTPUT(G)  PRODUCTION FUNCTIONS
 FEASIBLE(I)  INPUT FEASIBILITY;
 UTILITY..  BIGW =E= PROD(G, C(G)**SHARE(G));
 OUTPUT(G)..  PROD(I, F(I,G)**ELAS(I,G)) =G= C(G);
 FEASIBLE(I).. SUM(G, F(I,G)) =L= END(I);
MODEL AUTARCHY /ALL/;
F.L(I,G) = END(I)/2;
F.LO(I,G) = 1;
C.L(G) = PROD(I, F.L(I,G)**ELAS(I,G));
SOLVE AUTARCHY USING NLP MAXIMIZING BIGW;
```

are included, the solver, MINOS 5.2, solves a set of linearly-constrained problems, recomputing linear approximations to the non-linear constraints at regular intervals.[29] Second, since MINOS solves the problem in this way, it is vital that starting values of

---

[29] Each approximation to the constraint set constitutes a "major iteration". Within each major iteration, a number of "minor iterations" are undertaken, in attempts to improve the value of the objective function.

variables be entered to provide points around which initial linear approximations can be computed.[30] Third, the equation listing in GAMS output displays the coefficients of these linear approximations, rather than the actual non-linear equations. Fourth, since non-linear equations may be undefined (or absurd) at certain values of the endogenous variables, it is frequently necessary to restrict the domains of variables through the imposition of upper and/or lower bounds.

STARTING VALUES

Both FUNOBJ and FUNCON can be thought of as routines which construct linear functions which approximate the true non-linear functions at some given point, using a procedure analagous to a first order Taylor series expansion. The approximations are constructed using the first partial derivatives of the functions at the given point. Therefore, it is imperative that the user tell GAMS the point at which to begin.

If no starting values of variables are specified, GAMS will set all variables equal to zero. Unfortunately, this frequently implies that some or all of the partial derivatives are also equal to zero. In that case, some or all of the constraints will be approximated either with non-existent functions (if both sides of a constraint equation are approximated with zeroes) or functions the solution set of which is empty (if one side of the constraint equation is equal to zero and the other side is constrained to be non-zero).

In the Robinson Crusoe problem, the starting values for factor allocation are entered with the line immediately following the declaration of the model to be solved:

```
F.L(I,G) = END(I)/2;
```

This tells MINOS to begin the search for an optimum from the initial allocation in which half the total endowment of each factor is allocated to the production of each good. It also tells MINOS to construct the first set of linear approximations of the non-linear functions using these values.

---

[30] The algorithms are described in B.A. Murtagh, and M. A. Saunders, "Large-Scale Linearly Constrained Optimization", Mathematical Programming 14(1978):41-72 and in B. A. Murtagh, and M. A. Saunders, "A Projected Lagrangian Algorithm and its Implementation for Sparse Nonlinear Constraints", Mathematical Programming Study 16(1982):84-117.

Once an initial factor allocation is established, the starting values of output quantities can be computed directly, using the production functions:

```
C.L(G) = PROD(I, F.L(I,G)**ELAS(I,G));
```

This statement is simply the GAMS representation of a set of Cobb-Douglas production functions. For each good, the starting level (.L) is equal to the product of the previously declared factor input levels (F.L) raised to their respective exponents.

There is a third assignment statement (F.LO(I,G,)=1) before the SOLVE statement, which places a lower bound on the factor allocations. Discussion of the purpose of this is deferred until after the equation listing is presented and interpreted.

Starting values, as defined above, should not be confused with "initial conditions" in dynamic programming problems. An initial condition is a value of a stock variable in the initial time period, and, as such, is a datum of the problem. The difference is illustrated in the mine manager's problem (Appendix B, B.7). There, an initial condition is imposed on "X" with the statement

```
X.FX("T1")=RSRV;
```

This value of "X" cannot be changed, although "X" values in subsequent periods can be.

In contrast, starting values for "Y(TE)" are entered as

```
Y.L(TE)=RSRV/(CARD(T)-1);
```

These values are only first approximations, and can be altered in the course of solving the problem.

EQUATION LISTING

As with linear programming problems, part of the GAMS output for a non-linear program consists of a detailed listing of the equations entered using compact notation. In this case, however, the numbers in parentheses are the partial derivatives of the equation with respect to each argument evaluated at the starting values. In addition, the value of each equation, evaluated at the starting values is displayed.

To illustrate, consider first the utility function listing from the "EQUATION LISTING" for the Robinson Crusoe problem, shown on the next page.

The utility function, written in standard notation is

$$U(F,C) = F^{a}C^{b}$$

```
EQUATION LISTING    SOLVE AUTARCHY USING NLP FROM LINE 30


---- UTILITY     =E=  UTILITY FUNCTION

UTILITY..  - (0.25)*C(FISH) - (0.75)*C(COCONUTS) + BIGW =E= 0 ;

      (LHS = -50 ***)

---- OUTPUT      =G=  PRODUCTION FUNCTIONS

OUTPUT(FISH)..  - C(FISH) + (0.3333)*F(CAPITAL,FISH) +
(0.6667)*F(LABOUR,FISH)

      =G= 0 ; (LHS = 0)

OUTPUT(COCONUTS)..  - C(COCONUTS) + (0.6667)*F(CAPITAL,COCONUTS)

      + (0.3333)*F(LABOUR,COCONUTS) =G= 0 ; (LHS = 0)

---- FEASIBLE    =L=  INPUT FEASIBILITY

FEASIBLE(CAPITAL)..  F(CAPITAL,FISH) + F(CAPITAL,COCONUTS)

       =L= 100 ;  (LHS = 100)

FEASIBLE(LABOUR)..  F(LABOUR,FISH) + F(LABOUR,COCONUTS)
       =L= 100 ;  (LHS = 100)
```

A first-order Taylor expansion of this function, at the point (F^*,C^*), can be written as

$$(W = U(F,C)) = U(F^*,C^*) + (F-F^*)U_F + (C-C^*)F_C$$

where the partials are evaluated at (F^*,C^*).

The starting values for F^* and C^* are both equal to 50.[31] When these are inserted into the above expression, we have

$$(W = U(F,C)) = (50^{1/4})(50^{3/4}) + (1/4)(F-F^*) + (3/4)(C-C^*)$$

Pulling all terms over to the left-hand-side, we obtain

$$(50^{1/4})(50^{3/4}) - (1/4)(F-F^*) - (3/4)(C-C^*) + W = 0 \quad .$$

This equation, less the first term, is found in the "EQUATION LISTING" as

```
UTILITY..  - (0.25)*C(FISH) - (0.75)*C(COCONUTS) + BIGW =E= 0 ;
```

The entry following this, "(LHS = -50 ***)" represents the value of the left-hand-side of the original function, when it is written in implicit form, with all variables taking on their specified starting values. This can be seen if we write out the utility function as

$$W - U(F,C) = 0 \quad .$$

Starting values for "F" and "C" have been specified, but no starting value for "W" has been given. Therefore, GAMS sets "W" equal to zero at the beginning of the job. The value of the equation above then becomes:

$$0 - (50^{1/4})(50^{3/4})$$

which is equal to -50, as specified. The three asterisks signify that this equation (the utility function) always holds with equality, in contrast with constraint equations, which may or may not be binding.

The fact that the signs on all the terms are negative is purely an artifact of the way the utility function was entered. Had the function "UTILITY" been entered as

```
PROD(G, C(G)**SHARE(G)) =E= BIGW; ,
```

all the signs would have been positive. Signs depend only on the ordering of the equation. GAMS always writes equations with the variables on the left, and parameters, if any, on the right. Any variables found on the right-hand-side in the original input file are

---

[31] This can be verified by inserting the starting values of the inputs, "END(I)/2"(all of which equal 50 as well), into the production functions and evaluating.

subtracted from both the left and right sides. The equation listing for the utility function entered as shown above would be:

```
UTILITY..  (.25)*C(FISH) + (.75)*C(COCONUTS) - BIGW =E= 0 ;
       (LHS = 50 ***)
```

In contrast to the utility function listing, the equation listing for the production functions shows mainly positive terms. This is because, in the GAMS job file, these equations are entered with the functional form on the left, and the output variable on the right.

The production function listings are similar to that of the utility function, with two exceptions. First, the "(LHS...)" term is equal to zero. This is because each function, evaluated at the specified starting levels for factor inputs, is exactly equal to the specified starting levels of the outputs. Thus, when the equations are rewritten with all endogenous variables on the left, the initial values of the left-hand-sides equal zero. Second, although the left-hand-sides are equal to zero at these levels, they are constrained only to be no less than zero globally. Hence, the three asterisks, connoting equality, are absent.

The feasibility constraints are linear, but are treated in the same manner as non-linear constraints. In the "EQUATION LISTING", there are no bracketed terms preceding the variables, but this is simply because the partial derivatives of linear functions always equal constants.

BOUNDS ON VARIABLES

In linear programming problems, the variables are normally constrained only to be non-negative. In non-linear programming problems, allowing a variable to become equal to zero can give rise to special problems.

In the Robinson Crusoe problem, if an input or an output should become zero at some point, at least one of the two production functions or the utility function will equal zero. As the utility function is to be maximized, a zero output will not be infeasible, but simply non-optimal. Presumeably MINOS would reject zero utility as suboptimal, and continue to search. However, if an input level should become zero, all the partial derivatives of the relevant production function, as well as the value of the function itself will equal zero. Since the amount of a good consumed is constrained to be no greater than the amount produced, this immediately implies zero utility.

To prevent this, bounds can be placed on variables. This is done with assignment statements, using the identifier extensions .LO, .UP, and/or .FX. The extension .LO is used to impose lower bounds on variables, as is done in the Robinson Crusoe problem:

```
F.LO(I,G) = 1; .
```

The extension .UP is used to impose upper bounds. These are less common, but no less important in the applications which require them. For example, in the Conrad-Clark mine manager problem (Appendix B, B.7), it is obviously impossible in practice to extract more than the entire reserve in any period. The MINOS algorithm does not know this, however, and cannot solve the problem without being given this upper bound.

When a variable is to be fixed, its upper and lower bounds coincide at the fixed value. The extension .FX is used to set upper and lower bounds simultaneously at the same value.

GENERATING REALISTIC STARTING VALUES

The starting values necessary for executing non-linear optimization problems often may be computed by GAMS from previously-entered data. In the Robinson Crusoe model, starting values were assigned as functions of parameters. In many other applications, starting values for one problem can be assigned using the optimal values from one or more previously-solved models.

For example, in the spatial allocation model (Appendix B, B.6), starting values for the quantities demanded and supplied in each region must be entered before the problem can be solved. Since it is reasonable to expect that the optimal post-trade quantities will lie in the neighborhoods of the autarchy equilibrium quantities, those values make good starting points.

These can be entered as starting values in one of two ways. One way is for the analyst to solve for the autarchy equilibria before executing the GAMS job, and to enter each number separately. This, however, requires the user to perform tasks which can be done more easily (and frequently more accurately) by the computer. The other way of obtaining autarchy equilibria is by making use of the linearity of the supply and demand equations, and incorporating a small linear programming problem into the GAMS job file. Although the LP problem is written in conjunction with other statements in the Job, the essence of the LP problem is as follows.[32]

---

[32] Here, the sum of autarchy equilibrium quantities is maximized, subject to the simultaneous satisfaction of each region's demand and supply equations. Since there is only one solution to the constraint set of the linear programming problem, the type and direction of the linear objective function is arbitrary.

```
VARIABLES   Q(I) AUTARCHY EQBM QUANTITIES
            LPOBJ LP OBJECTIVE
            PD(I) DEMAND PRICES
            PS(J) SUPPLY PRICES
POSITIVE VARIABLES Q, PD, PS
EQUATIONS
   OBJLP      OBJECTIVE LP
   SUPPLY(J)  SUPPLY CURVES
   DEMAND(I)  DEMAND CURVES
   OBJLP..    SUM(I, Q(I)) =E= LPOBJ;
   SUPPLY(J).. PS(J) + ETA(J)*Q(J) =E= V(J);
   DEMAND(I).. PD(I) - OMG(I)*Q(I) =E= LAM(I);
MODEL AUTARCHY  PRE TRADE EQBA  /OBJLP, SUPPLY, DEMAND/;
SOLVE AUTARCHY USING LP MAXIMIZING LPOBJ;
```

Once the linear programming problem has been solved, the starting values for "X" and "Y" (quantities supplied and demanded, respectively) are assigned to be equal to the autarchy equilibrium quantities:[33]

```
Y.L(I) = Q.L(I); X.L(J) = Q.L(J);
```

Assigning "X.L(J)" and "Y.L(I)" to be equal to "Q.L(I)" makes sense only if non-zero values for "Q.L(I)" exist. Therefore, this assignment statement must <u>follow</u> the "SOLVE" statement of the linear model.[34] These values must, however, be assigned <u>before</u> the "SOLVE" statement for the non-linear problem is encountered.

DYNAMIC PROBLEMS

In general, dynamic problems can be written as multiple-equation static problems in which the set of time periods is treated as any other set. The solution of such problems presents no additional difficulties in GAMS. However, there are a few special features of dynamic problems, specifically leads, lags, and loops, which merit special mention.

---

[33] Note that, although "Q" was originally declared over the set "I", it may be referenced as being defined over the set "J". since "I" is aliased with "J".

[34] This is the operational meaning of the line from the symbol list of the output

```
Q      VAR...      ...IMPL-ASN   64
```

This indicates that values for "Q" are assigned implicitly by the SOLVE statement on line 64. Until the SOLVE statement has been executed, no non-zero values for "Q" will exist.

Leads and Lags

Set elements may be referenced in assignment or equation definition statements through the use of the set identifier alone, or by using the set identifier in combination with the lead ("+") or lag ("-") operators. These can be illustrated using the simple GAMS job file shown below:

```
SET T TIME PERIODS  /T1,T2,T3,T4,T5/;
PARAMETER Y(T)  RIGHT HAND SIDE
 /T1  10
  T2  12
  T3  11
  T4   8
  T5  14/;
PARAMETER A(T);
PARAMETER B(T);
PARAMETER C(T);
PARAMETER D(T);
A(T) = Y(T-1);
B(T) = Y(T+1);
C(T+1) = Y(T);
D(T-1) = Y(T);
DISPLAY A,B,C,D;
```

When the set identifier "T" is used alone, an assignment or equation is generated, element by element, for each member of the set. For example, the assignment

```
X(T) = Y(T);
```

is taken by GAMS to mean:

Create a parameter "X(T)", values for which exist for each of the five elements of "T". Assign to "X("T1")" the (previously-defined) value of "Y("T1")". Assign to "X("T2")" the value of "Y("T2")", and so on up to "T5".

The set identifier, in conjunction with the lead operator "(T+n)" instructs GAMS to generate assignments or equations associated with elements "n" ordinal positions after each element of the set. When the lead operator occurs on the left-hand-side of the equation or assignment, it has the effect of creating a new subset of the original set. For example, the assignment

```
C(T+1) = Y(T):
```

is interpreted by GAMS to mean:

Create a parameter "C" which exists for all elements of the set "T+1". This set contains all elements of "T" for which there exists an element in the previous ordinal position. Thus, "T1" is not

a member of "T+1", as there is no element in "T" which precedes "T1". Initially, all values of "C" are zero. Next, assign to each element of "C(T+1)" the value of "Y(T)" corresponding to the **previous** element of "T". The parameter "C" is displayed in the output as:

```
----      16 PARAMETER C

     T2 10.000,     T3 12.000,     T4 11.000,     T5  8.000
```

Note that the number of elements of a parameter to be assigned through the assignment statement depends on the parameter's index on the **left-hand-side** of the statement. The statement above results in values for "C" being assigned for all elements of "T" except the first. By contrast, the assignment for the parameter "B",

```
B(T) = Y(T+1);
```

will result in values being assigned for all elements of the set "T", the last of which is zero. There is no need for a newly-created subset of "T". The last value, "B("T5")", is zero because GAMS begins the execution of an assignment by setting to zero **all** values in the domain of the identifier on the left-hand-side, and only then assigns the values defined by the expression on the right-hand-side. Since there is no element in the ordinal position "5+1" of the set "T", "B("T5")" will not be assigned a new value, and thus remains equal to zero.

The parameter "B" is displayed in the output as:[35]

```
----      16 PARAMETER B

     T1 12.000,     T2 11.000,     T3  8.000,     T4 14.000
```

The set identifier in combination with the lag operator, "(T-n)", instructs GAMS to generate assignments or equation specifications associated with elements "n" ordinal positions **before** each element of the set. The assignment defining the parameter "A",

```
A(T) = Y(T-1);
```

results in values of "A(T)" being assigned for all elements of the set "T" (again, no subset is created), with the first element, "A("T1")" being assigned a value of zero. Here, all values of "A(T)" are initially set equal to zero and replaced with the value of "Y" associated with the **previous** ordinal position in the set "T". Since there is no element with an ordinal position before "T1", "A("T1")" is not assigned a new value and remains zero. The parameter "A" is displayed in the output as:

---

[35] It appears that only four values for "B" have been generated. However, recall that GAMS does not print zero values, nor labels associated with them.

```
----      16 PARAMETER A

     T2 10.000,     T3 12.000,     T4 11.000,     T5  8.000
```

When the lag operator is included in the index on the left-hand-side of the assignment, again a subset of the indexed set will be created. Thus, the assignment for the parameter "D",

```
D(T-1) = Y(T);
```

results in the generation of a new set, "T-1", which contains all elements of "T" for which there exists an element in the next ordinal position. "T5" is **not** an element of "T-1", since there is no element in "T" which comes one position after "T5". Each of the (4) elements of "T-1" is initially set to zero, then assigned values of "Y" which correspond to the **next** element of "T". The parameter "D" is displayed as:

```
----      16 PARAMETER D

     T1 12.000,     T2 11.000,     T3  8.000,     T4 14.000
```

When using leads and lags in assignment statements, the fact that GAMS regards non-existence and zero values as equivalent permits a certain flexibility. In the example above, the assignments used to create "A" and "C" differ in terms of internal execution, but the results are, for practical purposes, the same.

More care is required when leads and lags are used in equation specifications. To illustrate, consider the example of a monotonicity constraint on consumption, discussed in Meeraus.[36]

The problem is one of maximizing intertemporal utility by allocating consumption of a fixed stock of wealth through time. In addition to feasibility, an additional constraint is imposed:

$$C_{t+1} \geq C_t \ , \quad t = 1,2,3.$$

There are six possible ways to write this constraint equation in the "EQUATIONS" portion of the GAMS job file, of which only two are completely correct. The following tables detail the results of using each.

---

[36] A. Meeraus, GAMS Preliminary User's Guide, The World Bank, 1982.

```
EQUATIONS
     .
     .
     .
   MC(T)  MONOTONICITY CONSTRAINT;
     .
     .
     .
```

(1) **MC(T).. C(T) =G= C(T-1);**

	C(1)	C(2)	C(3)		RHS	
MC(1)	1			=G=	0	(redundant)
MC(2)	-1	1		=G=	0	
MC(3)		-1	1	=G=	0	

(2) **MC(T).. C(T+1) =G= C(T);**

	C(1)	C(2)	C(3)		RHS	
MC(1)	-1	1		=G=	0	
MC(2)		-1	1	=G=	0	
MC(3)			-1	=G=	0	(incorrect)

(3) **MC(T+1).. C(T) =G= C(T-1);**

	C(1)	C(2)	C(3)		RHS	
MC(1)						
MC(2)	1			=G=	0	(redundant)
MC(3)	-1	1		=G=	0	

(4) **MC(T+1).. C(T+1) =G= C(T);**

	C(1)	C(2)	C(3)		RHS	
MC(1)						
MC(2)	-1	1		=G=	0	
MC(3)		-1	1	=G=	0	

(5) **MC(T-1).. C(T) =G= C(T-1);**

	C(1)	C(2)	C(3)		RHS	
MC(1)	-1	1		=G=	0	
MC(2)		-1	1	=G=	0	
MC(3)						

(6) **MC(T-1).. C(T+1) =G= C(T);**

	C(1)	C(2)	C(3)		RHS	
MC(1)		-1	1	=G=	0	
MC(2)			-1	=G=	0	(incorrect)
MC(3)						

The "LOOP" Statement

In the subsection above, the sequence of actions initiated by an assignment statement is discussed. There, it is noted that GAMS sets to zero all terms on the left-hand-side of the assignment, then replaces these zeroes with the values of the expression on the right-hand-side. It is important to recognize that these are parallel operations. For example, the assignment

```
D(T) = X(T)*Y(T);
```

results in all elements of "D(T)" being set equal to zero before any of them are set equal to the corresponding values of "X(T)*Y(T)".

This implies that it is impossible, using only simple assignment statements, to instruct GAMS to perform sequential, or dynamic, assignments. In order to do so, it is necessary to use the "LOOP" statement.

The "LOOP" statement is similar in construction to expressions involving indexed operators. It has the form:

```
LOOP((index($condition)), assignment);
```

To illustrate, consider the example from Kendrick and Meeraus (footnote 3) in which quantities demanded for future years are projected from the base year using differing annual growth rates. The equation of motion describing quantities demanded is:

$$d_{t+1} = (1 + g_t)d_t \ , \ d_1 \text{ given}$$

where d_t = quantity demanded in year "t"
g_t = growth rate in year "t".

To generate the sequence of quantities demanded, it might seem logical to use the assignments:

```
D("1") = (initial period quantity);
D(T+1) = D(T)*(1+G(T));
```

Unfortunately, this intuitive statement, which looks very similar to the assignment statement in the section on leads and lags, will not generate demand quantities beyond year "2". This is due to the fact that both the left and right-hand side of the assignment involve the same parameter D. The reason for the problem in this area can be understood by examining the way in which GAMS executes assignment statements.

First, since the left-hand-side of the assignment is indexed by "(T+1)", the initial value, "D("1")", will not be changed. All other values ("D("2")", "D("3")", etc.) will be set equal to zero. Thus, at the completion of the first phase of the assignment, there will exist temporary values of "D(T)", all but the first of which are zero.

The assignment will then be executed, generating new values for "D(T)" by evaluating the expression on the right-hand-side with reference to the existing temporary values of "D(T)". Following the evaluation of the expression with respect to all temporary values, all of the values of "D(T)" will be replaced. Thus, the value of "D("2")", say, will not be replaced with "D("1")*(1 + G("1")" until the right-hand-side expression has been evaluated with respect to **all** temporary values.

The result of using the single assignment, then, will be:

T	D(T) temporary	D(T-1)temporary times (1 + G(T-1)
1	(initial)	(initial)
2	0	(initial)*G("1")
3	0	0
4	0	0
.	.	.
.	.	.
.	.	.

What is required is, in effect, a sequence of assignments. In each, only **one** value of "D(T)" is set equal to zero, then immediately replaced with "D(T-1)*(1+G(T-1))". Following the assignment of each "D(T)", the subsequent assignment takes "D(T)" to be a non-zero value and repeats the instructions for the next element in the set "T". This is done by entering:

```
D("1") = (initial value);
LOOP(T, D(T+1) = (1 + G(T))*D(T));
```

The "LOOP" statement above carries out the following instructions:

(a) Take the first element of "T" (= "1"). Assign the value zero to "D(T+1)" = "D("2")". Next, reset "D("2")" equal to "D("1")*(1+G("1"))". **Note that "D("1")" is not set to zero by this assignment.**

We now have D("1") = (initial)
D("2") = (initial)*(1+G("1")) .

(b) Take the second element of "T" (= "2"). Assign the value zero to "D(T+1)" = "D("3")". Reset "D("3")" equal to "D("2")*(1 + G("2")). As before, "D(T)" = "D("2")" is **not** set to zero at any point in executing this second assignment.

c) Perform this operation again for each remaining element of "T". When the last element is encountered, the expression "D(T+1)" will not be defined. Stop.

Loops may contain more than one assignment statement, and may in fact contain nested "LOOP" statements. For example, suppose that the demand projection discussed above involved quarterly, rather than annual, growth rates. Suppose further that these growth rates are entered in tabular format, as shown in the following GAMS job file:

```
SETS
 Q QUARTERS  /1*4/
 Y YEARS /85*88/
PARAMETER D(Q,Y) DEMAND BY QUARTER;
TABLE G(Q,Y) GROWTH RATE BY QUARTER
       85     86     87     88
1     .005   .004   .001   .006
2     .004   .006   .002   .002
3     .006   .005   .003   .004
4     .002   .005   .004   .009 ;
```

Demand in the first quarter of "85" is assigned to be equal to "100". Demand in each of the subsequent quarters is projected to be equal to the demand in the previous quarter times (1 + previous quarter growth rate). What is required here is some way of replacing the following lengthy list of assignments:

```
D("1","85") = 100;
D("2","85") = D("1","85")*(1+G("1","85");
D("3","85") = D("2","85")*(1+G("2","85");
D("4","85") = D("3","85")*(1+G("3","85");
D("1","86") = D("4","85")*(1+G("4","85");
     .                    .
     .                    .
     .                    .
D("4","88") = D("3","88")*(1+G("1","88");
```

with something much shorter.

For the first year, we may use

```
D("1","85") = 100;
LOOP(Q, D(Q+1,Y) = D(Q,Y)*(1+G(Q,Y));
```

The result of this will be as follows: Before any values are assigned to it, the two-dimensional parameter "D(Q,Y)" is composed of a table of zeroes. The first assignment sets "D("1","85")" equal to 100. The "LOOP" statement tells GAMS to create a loop over the set "Q" (which contains the elements {1,2,3,4}). Thus, we will have:

```
D("2","85") = (100)*(1+0.005)          ( = 100.5)
D("3","85") = (100.5)*(1+0.004)        ( = 100.902)
D("4","85") = (100.902)*(1+0.006)      ( = 101.507)
```

This is what we want. However, there is nothing in this "LOOP" statement which tells GAMS to set "D("1","86") equal to

"D("4","85")*(1+G("4","85")) and run through the loop again. As it stands, the loop is constructed only over the set "Q".

What we need, for each four-quarter period, is:

```
LOOP(Q, D(Q+1,Y) = D(Q,Y)*(1+G(Q,Y)));
D("1",Y+1) = D("4",Y)*(1+G("4",Y)));
```

The "LOOP" statement generates values for "D("2",Y)", "D("3",Y)" and "D("4",Y)", given a value for "D("1",Y)". The assignment statement below it generates a value for "D("1",Y+1)" given the value of "D("4",Y)". Now, we need to report this for the next four-quarter period.

Fortunately, it is possible to embed both lines in another "LOOP" statement. This will be a loop over the set "Y" (years):

```
LOOP(Y, LOOP(Q, D(Q+1,Y) = D(Q,Y)*(1+G(Q,Y)));
D("1",Y+1) = D("4",Y)*(1+G("4",Y)));
```

Note the number of parentheses. This two-line "LOOP" statement is interpreted by GAMS as:

(a) Take the first element of "Y" (= "85"). Take the first element of "Q" (= "1"). Set "D("2","85")" equal to "D("1","85")*(1+G("1","85").

(b) Still using the first element of "Y" take the second element of "Q" (= "2"). Set "D("3","85")" equal to "D("2","85")*(1+G("2","85")).

(c) Set "D("4","85")" equal to "D("3","85")*(1+G("3","85")).

(d) Take the fourth element of "Q". Since "Q+1" is not defined, exit from the loop over "Q" and go to the next line.

(e) Set "D("1","86") equal to "D("4","85")*(1+G("4","85")).

(f) Take the second element of "Y" (= "86"). Repeat steps (a)-(e), incrementing "Y" by one unit. Continue until reaching the last element of "Y". Stop, as the loop over "Y" is now complete.

Loops of assignment statements should not be used unless absolutely necessary. Assignments which do not involve dynamic equations are more efficiently carried out using single statements. These instruct GAMS to assign values for all elements of a parameter in parallel.

APPENDIX A

Functions, and Relational Operators

A.1. GAMS BUILT-IN FUNCTIONS

A partial list of built-in functions, adapted from Kendrick and Meeraus (1985), is presented below.

NAME AND ARGUMENTS	INTERPRETATION
ABS(X)	absolute value of X
CEIL(X)	the integer just larger than X
EXP(X)	"e" raised to the power X
FLOOR(X)	the integer just smaller than X
LOG(X)	natural (Neperian) logarithm of X
LOG10(X)	logarithm to base 10 of X
MAX(X1,X2,...,Xn)	the largest element of (X1,X2,...,Xn)
MIN(X1,X2,...,Xn)	the smallest element of (X1,X2,...,Xn)
POWER(X,Y)	X raised to the power Y, where Y must be an integer
ROUND(X,Y)	X rounded off to the digit Y. If Y is negative, this means the Y'th significant digit to the left of the decimal point. If Y is positive, this means the Y'th significant digit to the right of the decimal point. If Y is absent, X is rounded to the nearest integer.
SIGN(X)	The sign of X (+1, 0, or -1)
SQR(X)	X raised to the power 2
SQRT(X)	The square root of X
TRUNC(X)	The truncated value of X

A.2. RELATIONAL OPERATORS

GAMS contains most of the standard operators found in programming packages. These are listed below, in the order in which they are evaluated in a GAMS statement.

SYMBOL	INTERPRETATION
$	Conditional Operator, discussed in Section IV, "Special Features"
(X)**(Y)	X raised to the exponent Y. Note If "Y" is 2, it is better to use the special function SQR(X).
(X)*(Y)	X multiplied by Y
(X)/(Y)	X divided by Y
(X)+(Y)	The sum of X and Y
(X)-(Y)	X minus Y
LT	X less than Y
LE	X less than or equal to Y
(X) EQ (Y)	X equal to Y
NE	X not equal to Y
GE	X greater than or equal to Y
GT	X greater than Y

The order of priority can be altered by the use of parentheses, which can be nested up to 50 levels deep.

In structural equations, the relations "greater-than-or-equal-to","equal-to", and "less-than-or-equal-to" are written "=G=", "=E=", and "=L=", respectively. The reason for the difference is that the relations "=X=" are not operative until a "SOLVE" statement is encountered, while the other relations are operative without it.

APPENDIX B

GAMS Job File Examples

The purpose of this appendix is to illustrate how a number of applied problems can be formulated in GAMS. These applications are used frequently in agricultural and resource economics. Most sections contain an algebraic statement of the problem, followed by the Job File. If the algebraic statement is not given, it is found elsewhere in the document.

B.1. FARM PROBLEM, LP FORM, INCLUDING DUAL

This is an extended job file for the farm problem used in the tutorial in Section I, equation (5). Here, a solution to the dual is generated as a separate problem.

```
*FARM LP PROBLEM, INCORPORATING THE DUAL
*
 SETS
   J ACTIVITIES
     /STD-WHT, OATS, NEW-WHT/
   I INPUTS
     /RES1*RES4/
   PARAMETER R(J) REVENUE OF ACTIVITIES
     /STD-WHT    108.3
      OATS        66.36
      NEW-WHT    127.58/ ;
   PARAMETER B(I)  AVAILABILITY OF INPUTS
     /RES1   12
      RES2    8
      RES3  400
      RES4   80/ ;
   TABLE A(I,J)  USE OF INPUTS PER ACTIVITY
          STD-WHT     OATS     NEW-WHT
   RES1       1          1          1
   RES2       1          0          1
   RES3      30         20         40
   RES4       5          5          8 ;
 VARIABLES
  N   NET REVENUE
  X(J)  ACTIVITY LEVELS
  M   DUAL OBJECTIVE
  W(I)  DUAL ACTIVITY LEVELS
  POSITIVE VARIABLES X,W;
 EQUATIONS
   REVENUE         OBJECTIVE FUNCTION
   SUPPLY(I)      RESOURCE CONSTRAINTS
   OBJDUAL         DUAL OBJECTIVE FUNCTION
   CONDUAL(J)      DUAL CONSTRAINT FUNCTIONS;
   REVENUE..   SUM(J, X(J)*R(J)) =E= N;
   SUPPLY(I)..   SUM(J, X(J)*A(I,J)) =L= B(I);
   OBJDUAL..   SUM(I, W(I)*B(I)) =E= M;
```

```
   CONDUAL(J)..  SUM(I, W(I)*A(I,J)) =G= R(J);
   MODEL FARMPRIM  /REVENUE, SUPPLY/ ;
   MODEL FARMDUAL  /OBJDUAL, CONDUAL/ ;
   SOLVE FARMPRIM USING LP MAXIMIZING N;
   SOLVE FARMDUAL USING LP MINIMIZING M;
```

B.2. MOTAD VERSION OF THE FARM PROBLEM

This Job will set up and solve the MOTAD formulation of the problem given in Section II, equation (6). As stated there, the data input begins with five years of observations on gross revenue and the mean and deviations are calculated using GAMS.

```
*FARM PROBLEM, MOTAD FORMULATION
*
 SETS
     J ACTIVITIES
     /STD-WHT, OATS, NEW-WHT/
     I INPUTS
     /RES1*RES4/
     T TIME PERIODS
     /1*5/ ;
  PARAMETER B(I) AVAILABILITY OF INPUTS
     /RES1    12
      RES2     8
      RES3   400
      RES4    80/ ;
  TABLE A(I,J)  USE OF INPUTS PER ACTIVITY
          STD-WHT    OATS    NEW-WHT
  RES1        1         1        1
  RES2        1                  1
  RES3       30        20       40
  RES4        5         5        8 ;
   TABLE R(T,J)   REVENUES PER ACTIVITY PER TIME PERIOD
           STD-WHT    OATS    NEW-WHT
    1        99.8     68.3     112.7
    2       133.3    130.4     238.4
    3       142.7     33.3      93.9
    4       154.3     74.4      83.2
    5        11.4     25.4     109.7 ;
  PARAMETER RBAR(J)  AVERAGE REVENUE PER ACTIVITY;
     RBAR(J) = SUM(T, R(T,J)) / (CARD(T)) ;
   DISPLAY RBAR;
  PARAMETER D(T,J)  DEVIATIONS FROM MEAN REVENUES;
     D(T,J) = R(T,J)-RBAR(J);
   DISPLAY D;
 SCALAR XPINC  EXPECTED INCOME  /945/;
  VARIABLES   SAD     SUM OF INCOME DEVIATIONS
              X(J)    ACTIVITY LEVELS
              Y(T)  TIME COEFFICIENTS
              POSITIVE VARIABLES X,Y;
```

```
EQUATIONS
   OBJMOTAD    OBJECTIVE FUNCTION
   SUPPLY(I)   RESOURCE CONSTRAINTS
   ANNUAL(T)   YEARLY NONNEGATIVITY CONSTRAINTS
   EXPREV      EXPECTED REVENUE;
 OBJMOTAD..    SUM(T, Y(T))  =E= SAD;
 SUPPLY(I)..   SUM(J, X(J)*A(I,J)) =L= B(I);
 ANNUAL(T)..   (SUM(J, X(J)*D(T,J))) + Y(T) =G= 0;
 EXPREV..      SUM(J, X(J)*RBAR(J)) =E= XPINC;
  MODEL MOTAD /ALL/ ;
 SOLVE MOTAD USING LP MINIMIZING SAD;
```

As part of the output from this job, the following parameter lists will be printed:

```
----      30 PARAMETER RBAR        AVERAGE REVENUE PER ACTIVITY

    STD-WHT 108.300,     OATS      66.360,     NEW-WHT 127.580


----      33 PARAMETER D           DEVIATIONS FROM MEAN REVENUES

      STD-WHT         OATS      NEW-WHT

1      -8.500         1.940     -14.880
2      25.000        64.040     110.820
3      34.400       -33.060     -33.680
4      46.000         8.040     -44.380
5     -96.900       -40.960     -17.880
```

B.3. FARM PROBLEM, MINIMIZING HALF OF GINI'S MEAN DIFFERENCE

In 1982, Yitzhaki (see footnote 24) suggested using the mean and Gini's mean difference as an alternative approach to identifying efficient plans for risk averse decision makers. This mean-Gini (MG) efficiency approach uses the same data required for the MOTAD problem and is defined in the following way. Alternative F is MG efficient relative to alternative G if

(1B) $\mu_F \geq \mu_G$ and $\mu_F - \Gamma \geq \mu_G - \Gamma_G$,

where, for c_t^i = returns for alternative j (j=F,G) in year t

$$(2B) \quad \mu_j = \frac{1}{T} \sum_{t=1}^{T} c_{tj} \; ; \text{ and}$$

$$(3B) \quad \Gamma = \frac{1}{T^2} \sum_{t=1}^{T} \sum_{k>t}^{T} |c_{tj} - c_{kj}| = \text{half of Gini's mean difference.}$$

To generate the MG efficient locus from a set of alternatives, one can begin by first generating the MΓ efficient set. In a programming context, this can be done by finding all solutions which minimize Γ subject to mean income equal to a constant and parameterizing the mean income from zero to infinity.

Letting X_j be production activity j and c_{tj} is the return from activity j in year t, then following similar logic to that in the MOTAD formulation, both Yitzhaki (1982) and Okunev and Dillon (1989, see footnote 24), define:

$$(4B) \quad T_t = \sum_{j=1}^{n} c_{tj} X_j$$

Then, they maximize

$$(5B) \quad \Gamma = 1/n \sum_{t=1}^{T} \sum_{k>t}^{T} |T_t - T_k| \; ;$$

subject to

$$(6B) \quad \sum_{j=1}^{n} \mu_j X_j = \mu;$$

and the resource constraints

$$(7B) \quad \sum_{j=1}^{n} a_{rj} X_j \leq b_r \quad (r=1,\ldots,s)$$

(8B) $X_j \geq 0$ (all j).

In this form the problem is not linear but can be transformed into an LP problem by remembering that for $y_{tk}^+, y_{tk}^- \geq 0$

(9B) $|T_t - T_k| = (y_{tk}^+ + y_{tk}^-)$.

The problem then becomes

$$\text{(10B)} \quad \text{minimize} \sum_{t=1}^{n} \sum_{k>t}^{n} (y_{tk}^+ + y_{tk}^-)$$

s.t.

$$\text{(11B)} \quad \sum_{j=1}^{n} (c_{tj} - c_{kj})X_j - y_{ik}^+ + y_{ik}^- = 0 \quad \text{(all } t=1,\ldots,T \text{ and } k>T)$$

$$\text{(12B)} \quad \sum_{j=1}^{n} \mu_j X_j = \mu$$

$$\text{(13B)} \quad \sum_{j=1}^{n} a_{rj} X_j \leq b_r \quad (r=1,\ldots,5)$$

$$\text{(14B)} \quad X_j \geq 0; \; y_{ik}^+, y_{ik}^- \geq 0$$

Once the $\mu\Gamma$ efficient set is determined, the MG efficient set can be determined ex post by applying equation (1B). Using this algebraic formulation, the $\mu\Gamma$ efficient set for farm example (using the five years of gross revenue given above) can be generated using the following GAMS job. For large problems, calculating the differences used in constructing Gini mean difference could be exceedingly time consuming. An important feature of GAMS is that this set of differences can be calculated using only a few assignment statements.

```
* FARM PROBLEM, MEAN-GINI REPRESENTATION
*
  SETS
   J ACTIVITIES
     /STD-WHT, OATS, NEW-WHT/ ;
   I INPUTS
     /RES1*RES4/ ;
   T TIME PERIODS
     /1*5/ ;
   ALIAS (T,TP);
  TABLE  A(I,J)  INPUT USE PER ACTIVITY
          STD-WHT     OATS     NEW-WHT
  RES1        1          1          1
  RES2        1                     1
  RES3       30         20         40
  RES4        5          5          8 ;
  PARAMETER B(I)  AVAILABILITY OF INPUTS
     /RES1   12
      RES2    8
      RES3  400
      RES4   80 ;
  TABLE R(T,J)  REVENUES PER ACTIVITY PER TIME PERIOD
        STD-WHT   OATS   NEW-WHT
   1     99.8     68.3    112.7
   2    133.3    130.4    238.4
   3    142.7     33.3     93.9
   4    154.3     74.4     83.2
   5     11.4     25.4    109.7 ;
  PARAMETER RBAR(J)  AVERAGE REVENUE PER ACTIVITY;
    RBAR(J) = SUM(T, R(T,J))/(CARD(T));
    DISPLAY RBAR;
  PARAMETER RIJK(T,TP,J)  DIFFERENCES IN ANNUAL REVENUES;
   RIJK(T,TP,J) =(R(T,J)-R(TP,J))$(ORD(T)LT ORD(TP));
    DISPLAY RIJK;
  SCALAR XPINC  EXPECTED INCOME  /945/ ;
 VARIABLES
  MG MEANGINI COEFFICIENT
  X(J) ACTIVITY LEVELS
  YPLUS(T,TP)  POSITIVE Y VALUES
  YMINUS(T,TP)  NEGATIVE Y VALUES
  POSITIVE VARIABLES X, YPLUS, YMINUS;
 EQUATIONS
  OBJMGINI      OBJECTIVE FUNCTION
  SUPPLY(I)     INPUT AVAILABILITY
  ANNUAL(T,TP)  YEARLY NONNEGATIVITY CONSTRAINTS
  EXPREV        EXPECTED REVENUE ;
  OBJMGINI..  SUM((T,TP)$(ORD(T)LT ORD(TP)), YPLUS(T,TP))
              +SUM((T,TP)$(ORD(T)LT ORD(TP)),YMINUS(T,TP))
              =E= MG;
  SUPPLY(I)..    SUM(J, X(J)*A(I,J))  =L= B(I) ;
  ANNUAL(T,TP).. ((SUM(J, X(J)*RIJK(T,TP,J)))-YPLUS(T,TP)
                 +YMINUS(T,TP))$(ORD(T)LT ORD(TP)) =E= 0;
  EXPREV..  SUM(J, X(J)*RBAR(J)) =E= XPINC ;
```

```
 MODEL MEANGINI  /ALL/ ;
SOLVE MEANGINI USING LP MINIMIZING MG;
```

As part of the output from this job, the following parameter lists will be printed:

```
----      31 PARAMETER RBAR       AVERAGE REVENUE PER ACTIVITY

    STD-WHT 108.300,    OATS      66.360,    NEW-WHT 127.580

----      34 PARAMETER RIJK       DIFFERENCES IN ANNUAL REVENUES

        STD-WHT        OATS     NEW-WHT

1.2     -33.500      -62.100   -125.700
1.3     -42.900       35.000     18.800
1.4     -54.500       -6.100     29.500
1.5      88.400       42.900      3.000
2.3      -9.400       97.100    144.500
2.4     -21.000       56.000    155.200
2.5     121.900      105.000    128.700
3.4     -11.600      -41.100     10.700
3.5     131.300        7.900    -15.800
4.5     142.900       49.000    -26.500
```

B.4. FARM PROBLEM, MINIMIZING THE VARIANCE OF ANNUAL REVENUE

This Job illustrates the input for the mean-variance (E-V) efficiency locus for the farm problem using the same five years of data on gross margins (H. Markowitz, "Portfolio Selection", The Journal of Finance, 7(1952):76-90.) The problem is to find $X_j \geq 0$:

$$\min. \sum_{i=1}^{n} \sum_{j=1}^{n} \sigma_{ij} X_i X_j$$

s.t.

$$\sum_{j=1}^{n} a_{ij} X_j \leq b_i \quad (i=1,\ldots,m)$$

$$\sum_{j=1}^{n} \bar{c}_j X_j = \mu$$

where σ_{ij} is the covariance of revenue between i and j and is the variance when i = j, and $\bar{c}_j$ is average revenue.

Notice that because this is a quadratic programming problem, the solve statement involves NLP (the non-linear solver).

```
*FARM PROBLEM SET UP AS MEAN-VARIANCE (EV) QUADRATIC
* PROGRAMMING PROBLEM
  SETS
   J ACTIVITIES
     /STD-WHT, OATS, NEW-WHT/
   I INPUTS
     /RES1*RES4/
   T TIME PERIODS
     /1*5/
  ALIAS (J,JP);
  PARAMETER B(I)  AVAILABILITY OF INPUTS
     /RES1   12
      RES2    8
      RES3  400
      RES4   80/ ;
  TABLE R(T,J)  REVENUES PER ACTIVITY PER TIME PERIOD
         STD-WHT   OATS   NEW-WHT
     1    99.8     68.3    112.7
     2   133.3    130.4    238.4
     3   142.7     33.3     93.9
     4   154.3     74.4     83.2
     5    11.4     25.4    109.7
  TABLE A(I,J)  INPUT USE PER ACTIVITY
            STD-WHT    OATS    NEW-WHT
   RES1        1         1        1
   RES2        1         0        1
   RES3       30        20       40
```

```
 RES4        5         5        8  ;
PARAMETER RBAR(J)  AVERAGE REVENUE PER ACTIVITY;
  RBAR(J) = SUM(T, R(T,J)) / (CARD(T));
  DISPLAY RBAR;
PARAMETER D(T,J)  DEVIATIONS FROM MEAN REVENUES;
  D(T,J) = R(T,J)-RBAR(J);
  DISPLAY D;
PARAMETER V(J,JP)  COVARIANCE MATRIX OF REVENUES;
  V(J,JP) = SUM(T, D(T,J)*D(T,JP)) / (CARD(T)-1);
  DISPLAY V;
 SCALAR  XPINC  EXPECTED INCOME  /945/;
VARIABLES
   RISK  OBJECTIVE VARIABLE
   X(J) ACTIVITY LEVELS
   POSITIVE VARIABLE X;
EQUATIONS
   OBJ         OBJECTIVE FUNCTION
   SUPPLY(I)   INPUT AVAILABILITY
   EXPREV      EXPECTED REVENUE;
OBJ..         SUM(JP, X(JP)*(SUM(J, X(J)*V(J,JP))))
   =E= RISK;
SUPPLY(I)..  SUM(J, X(J)*A(I,J))  =L= B(I);
EXPREV..     SUM(J, X(J)*RBAR(J)) =E= XPINC;
MODEL EV  /OBJ, SUPPLY, EXPREV/ ;
SOLVE EV USING NLP MINIMIZING RISK;
```

As part of the output, the following parameter lists will be printed:

```
----      32 PARAMETER RBAR        AVERAGE REVENUE PER ACTIVITY

    STD-WHT 108.300,    OATS      66.360,    NEW-WHT 127.580


----      35 PARAMETER D           DEVIATIONS FROM MEAN REVENUES

      STD-WHT        OATS     NEW-WHT

1      -8.500       1.940     -14.880
2      25.000      64.040     110.820
3      34.400     -33.060     -33.680
4      46.000       8.040     -44.380
5     -96.900     -40.960     -17.880


----      38 PARAMETER V           COVARIANCE MATRIX OF REVENUES

            STD-WHT        OATS     NEW-WHT

STD-WHT    3346.555    1196.527     357.370
OATS       1196.527    1735.053    2139.264
NEW-WHT     357.370    2139.264    3981.527
```

B.5. TRANSPORTATION PROBLEM

This job is to minimize the cost of transporting goods (at fixed per unit costs) in fixed supply at three sources to meet the fixed demand in three other regions. (S. Gass, Linear Programming Methods and Applications, 5th ed. McGraw Hill Book Co., New York, 1985.) The problem is (x_{ij} = amount shipped from origin i to destination j):

$$\max.\ 0x_{11} + 2x_{12} + 3x_{13} + 2x_{21} + 0x_{22} + 1x_{23} + 3x_{31} + 1x_{32} + 0x_{33}$$

s.t.

$$\begin{aligned}
x_{11} + x_{21} + x_{31} &\geq 20\\
x_{12} + x_{22} + x_{32} &\geq 25\\
x_{13} + x_{23} + x_{33} &\geq 61.538\\
x_{11} + x_{12} + x_{13} &\leq 30\\
x_{21} + x_{22} + x_{23} &\leq 15\\
x_{31} + x_{32} + x_{33} &\leq 61.538
\end{aligned}$$

$$x_{ij} \geq 0 \text{ all } i,j$$

```
*SPATIAL ALLOCATION DATA APPLIED TO SIMPLE TRANSPORTATION
* MODEL
*
 SETS
   I REGIONS
   /R1*R3/;
   ALIAS (I,J);
 PARAMETER X(I)  QUANTITIES SUPPLIED IN REGIONS
   /R1  30
    R2  15
    R3  61.538/;
 PARAMETER Y(J) QUANTITIES DEMANDED IN REGIONS
   /R1   20
    R2   25
    R3   61.538/;
 TABLE C(I,J)  UNIT TRANSPORT COSTS BETWEEN REGIONS
         R1   R2   R3
   R1     0    2    3
   R2     2    0    1
   R3     3    1    0 ;
 VARIABLES   XT(I,J) SHIPMENTS BETWEEN REGIONS
             TRCOST  TOTAL TRANSPORTATION COST
             POSITIVE VARIABLE XT;
 EQUATIONS
    COST      OBJECTIVE FUNCTION
    SUPPLY(I)  QUANTITIES SUPPLIED IN REGIONS
    DEMAND(J)  QUANTITIES DEMANDED IN REGIONS;
  COST..     SUM((I,J), C(I,J)*XT(I,J)) =E= TRCOST;
  SUPPLY(I).. SUM(J, XT(I,J)) =L= X(I);
  DEMAND(J).. SUM(I, XT(I,J)) =G= Y(J);
  MODEL TRANSPORT /ALL/ ;
  SOLVE TRANSPORT MINIMIZING TRCOST USING LP;
  DISPLAY XT.L;
```

B.6. SPATIAL ALLOCATION PROBLEM

In this job, a spatial allocation problem, in which the demand and supply curves for the commodity are linear in each region, is solved for the competitive trade optimum. The competitive trade solution is obtained. Letting y_i, x_i and p_i, be demand, supply and price in region i, respectively, the supply and demand functions and transportation costs between regions are given by:

$p_1 = 10 - 0.1y_1$ $\quad p_2 = 15 - .2y_2$ $\quad p_3 = 20 - .15y_3$

$p_1 = 5 + .1x_1$ $\quad p_2 = 2.5 + .5x_2$ $\quad p_3 = 4 + .1x_3$

$t_{11} = 0$ $\quad t_{12} = 2$ $\quad t_{13} = 3$

$t_{21} = 2$ $\quad t_{22} = 0$ $\quad t_{23} = 1$

$t_{31} = 3$ $\quad t_{32} = 1$ $\quad t_{33} = 0$

The problem formulated in quantity terms is:

$$\max. \begin{bmatrix}20 & 15 & 20\end{bmatrix}\begin{bmatrix}y_1\\y_2\\y_3\end{bmatrix} - \begin{bmatrix}5 & 2.5 & 4\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix}$$

$$-1/2\begin{bmatrix}y_1 & y_2 & y_3\end{bmatrix}\begin{bmatrix}.1 & 0 & 0\\0 & .2 & 0\\0 & 0 & .15\end{bmatrix}\begin{bmatrix}y_1\\y_2\\y_3\end{bmatrix} -1/2\begin{bmatrix}x_1 & x_2 & x_3\end{bmatrix}\begin{bmatrix}.1 & 0 & 0\\0 & .5 & 0\\0 & 0 & .11\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix}$$

$$-\begin{bmatrix}0 & 2 & 3 & 2 & 0 & 1 & 3 & 1 & 0\end{bmatrix}\begin{bmatrix}X_{11}\\X_{12}\\X_{13}\\X_{21}\\X_{22}\\X_{23}\\X_{31}\\X_{32}\\X_{33}\end{bmatrix}$$

subject to

$$\begin{bmatrix}1 & & & 1 & & & 1 & & \\ & 1 & & & 1 & & & 1 & \\ & & 1 & & & 1 & & & 1\\ -1 & -1 & -1 & & & & & & \\ & & & -1 & -1 & -1 & & & \\ & & & & & & -1 & -1 & -1\end{bmatrix}\begin{bmatrix}X_{11}\\X_{12}\\X_{13}\\X_{21}\\X_{22}\\X_{23}\\X_{31}\\X_{32}\\X_{33}\end{bmatrix} \geq \begin{bmatrix}y_1\\y_2\\y_3\\-x_1\\-x_2\\-x_3\end{bmatrix}$$

y_i, x_i and $x_{ij} \geq 0$ all i and j.

This problem is a quadratic programming problem. Thus, it is solved using NLP; the starting values for the NLP are given by the solution to a small LP problem in which output from the three regions is maximized

subject to supply being equal to demand in each region. This is essentially the autarchy solution. After this problem is solved (also using NLP), GAMS statements are constructed to solve for equilibrium prices, given the equilibrium quantities from the solution.

```
*SPATIAL ALLOCATION MODEL WITH LINEAR DEMAND AND SUPPLY
* CURVES,  SOLVED USING QUADRATIC PROGRAMMING

* STARTING VALUES ARE COMPUTED WITH A SMALL LP FORMULATION

 SETS
   I REGIONS
   /R1*R3/;
   ALIAS (I,IP,J,JP);
 PARAMETER LAM(I)  DEMAND FUNCTION INTERCEPTS
   /R1  10
    R2  15
    R3  20/ ;
 PARAMETER OMG(I)  DEMAND FUNCTION SLOPES (ABSOLUTE VALUE)
   /R1  0.1
    R2  0.2
    R3  0.15/ ;
 PARAMETER V(J)  SUPPLY FUNCTION INTERCEPTS
   /R1  5
    R2  2.5
    R3  4  / ;
 PARAMETER ETA(J)  SUPPLY FUNCTION SLOPES
   /R1  0.1
    R2  0.5
    R3  0.11/ ;
 TABLE C(I,J)  UNIT TRANSPORT COSTS BETWEEN REGIONS
         R1   R2   R3
   R1     0    2    3
   R2     2    0    1
   R3     3    1    0 ;
 PARAMETER OMEGA(I,IP)  DEMAND SLOPE MATRIX;
    OMEGA(I,IP) = OMG(I)$(ORD(I)EQ ORD(IP));
   DISPLAY OMEGA;
 PARAMETER H(J,JP)  SUPPLY SLOPE MATRIX;
    H(J,JP) = ETA(J)$(ORD(J)EQ ORD(JP));
   DISPLAY H;
 VARIABLES   Y(I) QUANTITIES DEMANDED
             X(J) QUANTITIES SUPPLIED
             Q(I) AUTARCHY EQBM QUANTITIES
             XT(I,J) SHIPMENTS BETWEEN REGIONS
             LPOBJ LP OBJECTIVE
             PD(I) DEMAND PRICES
             PS(J) SUPPLY PRICES
             WELF   TOTAL WELFARE
             POSITIVE VARIABLES X,XT,Y,PD,PS;
```

```
EQUATIONS
   OBJLP       OBJECTIVE LP
   OBJSPAT     OBJECTIVE SPATIAL
   SUPPLY(J)   SUPPLY CURVES
   DEMAND(I)   DEMAND CURVES
   PRDN(J)     QUANTITIES PRODUCED
   CONS(I)     QUANTITIES CONSUMED;
 OBJLP..     SUM(I, Q(I)) =E= LPOBJ;

 OBJSPAT..  SUM(I, LAM(I)*Y(I)) -SUM(J, V(J)*X(J))
            - (SUM(IP, Y(IP)*(SUM(I, Y(I)*OMEGA(I,IP)))) / 2)
     -(SUM(JP, X(JP)*(SUM(J, X(J)*H(J,JP)))) / 2)
            -SUM((I,J), XT(I,J)*C(I,J))  =E=  WELF;
 SUPPLY(J).. PS(J) - ETA(J)*Q(J) =E= V(J);
 DEMAND(I).. PD(I) + OMG(I)*Q(I) =E= LAM(I);
 PRDN(I)..  -SUM(J, XT(I,J)) =G= -X(I);
 CONS(J)..  SUM(I, XT(I,J))  =G=  Y(J);
MODEL AUTARCHY  PRE TRADE EQBA  /OBJLP, SUPPLY, DEMAND/;
SOLVE AUTARCHY USING LP MAXIMIZING LPOBJ;
Y.L(I) = Q.L(I); X.L(J) = Q.L(J);
MODEL SPATIAL /OBJSPAT, PRDN, CONS/ ;
 SOLVE SPATIAL MAXIMIZING WELF USING NLP;
 PARAMETER NOTRADE  NO TRADE PRICES AND QUANTITIES;
 NOTRADE(I,"PRICE") = LAM(I)-OMG(I)*Q.L(I);
 NOTRADE(I,"QUANTITY") = Q.L(I);
 PARAMETER CLEAR  MARKET CLEARING PRICES AND QUANTITIES;
 CLEAR(I, "DEMANDED") = Y.L(I);
 CLEAR(J, "SUPPLIED") = X.L(J);
 CLEAR(I, "DEMPRICE") = LAM(I)-OMG(I)*Y.L(I);
 CLEAR(J, "SUPPRICE") = V(J)+ETA(J)*X.L(J);
 PARAMETER TRADE SHIPMENTS FROM ROW TO COLUMN;
 TRADE(I,J) = XT.L(I,J);
  DISPLAY NOTRADE, CLEAR, TRADE;
```

The "DISPLAY" statements referring to the matrices in the objective function will result in the following being printed:

```
----      31 PARAMETER OMEGA      DEMAND SLOPE MATRIX

             R1          R2          R3

R1        0.100
R2                    0.200
R3                                0.150

----      34 PARAMETER H          SUPPLY SLOPE MATRIX

             R1          R2          R3

R1        0.100
R2                    0.500
R3                                0.110
```

The report writer (which begins with the declaration of the parameter "NOTRADE") uses the optimal quantities from the solution process to compute the equilibrium prices in each region. These, together with the optimal quantities themselves, and the shipments between regions, are reported on the final page of output as:

---- 61 PARAMETER CLEAR MARKET CLEARING PRICES AND QUANTITIES

	DEMANDED	SUPPLIED	DEMPRICE	SUPPRICE
R1	20.000	30.000	8.000	8.000
R2	25.000	15.000	10.000	10.000
R3	61.538	61.538	10.769	10.769

---- 61 PARAMETER TRADE SHIPMENTS FROM ROW TO COLUMN

	R1	R2	R3
R1	20.000	10.000	
R2		15.000	
R3			61.538

B.7. OPTIMAL EXTRACTION PROBLEM, WITH REPORT WRITER

An example of a dynamic problem, in which the objective is to determine the optimal program of extraction of ore from a mine, over a ten year period, is found in J. Conrad and C. Clark, Natural Resource Economics, Cambridge University Press, Cambridge, 1987. The problem involves a non-linear objective, since extraction costs are stock-dependent, but incorporates only linear constraints. The problem can be described and written as follows.

A mine is to be shut down in year $t = 10$, but before then, the manager must determine the optimal production schedule, y_t^+, for periods $t=0,\ldots,9$. The price of the ore being mined is given by $P = 1$ and the cost of extracting y_t is $C_t = y_t^2/x_t$, where x_t is the remaining ore reserves at the start of period t.

Net revenue in any period t is $py_t - y_t^2/x_t = (1-y_t/x_t)y_t$. The change in reserves can be described by $x_{t+1} - x_t = -y_t$, and initial reserves are $x_0 = 1000$.

To maximize the discounted stream of net revenue for a discount rate of 0.1 the following problem can be solved:

$$\max. \quad \sum_{t=0}^{9} \rho^t \, [1 - y_t/x_t]y_t$$

s.t.

$$x_{t+1} = x_t - y_t$$

$$x_0 = 1000$$

$$y_t, \; x_t \geq 0;$$

where $\rho = 1/1+\delta$.

```
SETS
 T TIME PERIODS   /0*10/
 TI(T) FIRST PERIOD
 TE(T) EXTRACTION PERIODS;
TI(T) = YES$(ORD(T) EQ 1);
TE(T) = YES$(ORD(T) LT CARD(T));
DISPLAY TI, TE;
SCALAR DELTA DISCOUNT RATE /0.1/
SCALAR DISC DISCOUNT;
DISC = 1/(1+DELTA);
DISPLAY DISC;
PARAMETER RHO(T) DISCOUNT FACTOR;
RHO(T) = DISC**(ORD(T)-1);
DISPLAY RHO;
SCALAR RSRV INITIAL RESERVE /1000/;
VARIABLES  Y(T) PRODUCTION
           X(T) REMAINING RESERVES
           PI  TOTAL PROFIT
  POSITIVE VARIABLES X, Y;
```

```
EQUATIONS
 NETREV  NET REVENUE
 STOCK(T)  STOCK ADJUSTMENT ;
 NETREV.. PI =E= SUM(TE, RHO(TE)*Y(TE)*(1-Y(TE)/X(TE)));
 STOCK(T+1)..  X(T+1) =E= X(T)-Y(T) ;
X.FX(TI) = RSRV;
X.LO(T) = .001;
X.UP(T) = RSRV;
Y.L(TE) = RSRV/(CARD(T)-1);
MODEL MINE  /ALL/ ;
SOLVE MINE USING NLP MAXIMIZING PI;
PARAMETER ANSWERS  FINAL SOLUTION VALUES;
 ANSWERS(T, "STOCK") = X.L(T);
 ANSWERS(T, "PRODUCTION") = Y.L(T);
 ANSWERS(T, "LAMBDA") = STOCK.M(T);
DISPLAY ANSWERS;
```

The output from this job will include a display of created subsets and created parameters. This display is shown here:

```
----      7 SET       TI          FIRST PERIOD

    0


----      7 SET       TE          EXTRACTION PERIODS

    0,    1,    2,    3,    4,    5,    6,    7,    8,    9


----     11 PARAMETER DISC        =        0.909 DISCOUNT


----     14 PARAMETER RHO         DISCOUNT FACTOR

   0  1.000,    1  0.909,    2  0.826,    3  0.751,
   4  0.683,    5  0.621     6  0.564,    7  0.513,
   8  0.467,    9  0.424,    10 0.386
```

The last portion of this file creates a report of the results, grouping in a single table production, remaining reserves, and the dynamic Lagrange multiplier. The table is shown here:[1]

---- 35 PARAMETER ANSWERS FINAL SOLUTION VALUES

	STOCK	PRODUCTION	LAMBDA
0	1000.000	238.229	
1	761.771	183.679	0.524
2	578.092	141.821	0.471
3	436.271	109.713	0.421
4	326.558	85.094	0.373
5	241.464	66.219	0.327
6	175.246	51.741	0.280
7	123.505	40.613	0.231
8	82.891	32.026	0.176
9	50.865	25.433	0.106
10	25.433		EPS

[1]The table is not identical to that shown on page 35 of Conrad and and Clark. The difference is due to their reporting current value multipliers (lambdas), whereas GAMS computes, as marginal values, present value multipliers.

B.8 AN INPUT-OUTPUT ANALYSIS

Many agricultural and resource economists have used input-output (I-O) methods in their research, although I-O analysis is normally not thought of as an application of mathematical programming methods. It can be demonstrated that for an I-O system with only one primary factor of production, output and price determination in an I-O system can be viewed as the unique solutions to a primal and a dual programming problem, (C. Yan, Introduction to Input-Output Economics, Holt, Rhinehart and Winston, Inc., New York, 1969). The I-O model is well known and can be summarized algebraically by

$$X = AX + Y$$

where

X is a (Mx1) vector of output from the n sectors of an economy;

A is an (nxn) matrix of input-output coefficients (the input from industry i to produce a unit of output j);

Y is an (nx1) vector of exogenous final demands for the n sectors.

Rearranging this set of equations, we have

$$X - AX = (I-A)X = Y$$

or

$$X = (I-A)^{-1} Y.$$

That is, this set of equations provides estimates of the output from each sector needed to deliver Y to final demand.[2]

The solution to this problem requires the calculation of $(I-A)^{-1}$ and GAMS does not have the capacity to invert a matrix directly. However, for a matrix B, one solves the problem

$$\begin{bmatrix} B & & & 0 \\ & B & & \\ & & B & \\ & & & \ddots \\ 0 & & & B \end{bmatrix} \begin{bmatrix} Q_1 \\ Q_2 \\ \cdot \\ \cdot \\ \cdot \\ Q_n \end{bmatrix} = \begin{bmatrix} e_1 \\ e_2 \\ \cdot \\ \cdot \\ \cdot \\ e_n \end{bmatrix}$$

---

[2]Under quite general conditions, one can show that $(I-A)^{-1}$ does exist. (F. V. Waugh, "Inversion of the Leontief Matrix by Power Series", Econometrica, 18(1950):142-154.)

where e_i are nx1 column vectors with a 1 in the i^{th} row and zeros elsewhere. The (nx1) vectors, Q_i, are the n columns of B^{-1}. By substituting (I-A) for B, we obtain the Leontief matrix.

The GAMS routine that follows calculates this Leontief matrix and illustrates how it can then be used to derive forecasts of output needed to meet new final demand levels and how to calculate type I and type II income multipliers. It should be noted that one could maximize any linear function and still obtain the appropriate inverse matrices. By maximizing the sum of the Q's, the sales or output multipliers are the dual variables to the problem. The transactions matrix used in the example is in figure B.1.

Figure B.1. Transactions Table for Empirical Example (in thousands of $)

Inputs	Output						
	Ag.	Man.	Trans.	Serv.	House.	Other Final Demand	Total Sales
Agriculture	34	290	0	0	7	138	469
Manufacturing	25	1,134	5	201	607	12,340	14,312
Transportation	6	304	54	105	22	119	610
Services	48	962	71	877	2,558	2,381	6,897
Households w	208	3,242	252	2,697	869	1,447	8,715
Other Final Payments v	148	8,380	228	3,017	4,652		
Total Purchases	469	14,312	610	6,897	8,715		

```
*INPUT OUTPUT COMPUTATIONS
 SETS
  I SECTORS /AGR,MFG,TRNS,SVC,HH/
  ALIAS (I,J,K);
 TABLE T(I,J) TRANSACTIONS TABLE HH ENDOGENOUS
         AGR    MFG    TRNS   SVC    HH
 AGR      34    290                    7
 MFG      25   1134       5   201   607
 TRNS      6    304      54   105    22
 SVC      48    962      71   877  2558
 HH      208   3242     252  2697   869;
 PARAMETER G(J) GROSS OUTPUT HOUSEHOLDS ENDOGENOUS
  /AGR 469,MFG 14312,TRNS 610,SVC 6897,HH 8715/
 PARAMETER A(I,J) DIRECT REQUIREMENTS HH ENDOGENOUS;
  A(I,J) = T(I,J)/G(J);
 PARAMETER ID(I,K)  IDENTITY MATRIX ORDER CARD(I);

  ID(I,K) = 1$(ORD(I) EQ ORD(K));
 PARAMETER AP(I,J) DIRECT REQUIREMENTS HH EXOGENOUS;
  AP(I,J)$(ORD(I)LT CARD(I) AND ORD(J)LT CARD(I))
              =  A(I,J);
 PARAMETER IDP(I,K) IDENTITY MATRIX ORDER CARD(I)-1;
 IDP(I,K)$(ORD(I)LT CARD(I)) =
         1$(ORD(I) EQ ORD(K));
 DISPLAY A,AP,ID,IDP;
 VARIABLES
  EXOBJ         OBJECTIVE HOUSEHOLDS EXOGENOUS
  ENDOBJ        OBJECTIVE HOUSEHOLDS ENDOGENOUS
  XEX(J,K)      INVERSE MATRIX HH EXOGENOUS
  XEND(J,K)     INVERSE MATRIX HH ENDOGENOUS;
 EQUATIONS
  OBJEND         OBJECTIVE FUNCTION HH ENDOGENOUS
  OBJEX          OBJECTIVE FUNCTION HH EXOGENOUS
  MATEND(I,K)    MATRIX CONSTRAINT HH ENDOGENOUS
  MATEX(I,K)     MATRIX CONSTRAINT HH EXOGENOUS;
  OBJEND..       SUM(J,SUM(K,XEND(J,K))) =E= ENDOBJ;
  OBJEX..        SUM(J$(ORD(J)LT CARD(I)),
       SUM(K$(ORD(K)LT CARD(I)),XEX(J,K))) =E= EXOBJ;
  MATEND(I,K)..   SUM(J,(ID(I,J)-A(I,J))*XEND(J,K)) =E= ID(I,K);

  MATEX(I,K)$(ORD(I)LT CARD(I) AND ORD(K)LT CARD(I))..
     SUM(J,(IDP(I,J)-AP(I,J))*XEX(J,K)) =E= IDP(I,K);
 MODEL ENDO  INVERSE HH ENDOGENOUS  /OBJEND,MATEND/;
 MODEL EXOG  INVERSE HH EXOGENOUS  /OBJEX,MATEX/;
 OPTION LIMROW = 0; OPTION LIMCOL = 0;
 SOLVE ENDO USING LP MAXIMIZING ENDOBJ;
 PARAMETER ENDINV(J,K) I-A INVERSE HH ENDOGENOUS;
  ENDINV(J,K) = XEND.L(J,K);
 SOLVE EXOG USING LP MAXIMIZING EXOBJ;
 PARAMETER EXINV(J,K) I-A INVERSE HH EXOGENOUS;
  EXINV(J,K) = XEX.L(J,K);
 DISPLAY ENDINV,EXINV;
```

```
PARAMETER HHINC(J) HOUSEHOLD INCOME BY SECTOR
 /AGR 208,MFG 3242,TRNS 252,SVC 2697,HH 869/
PARAMETER DIR(J) DIRECT INCOME REQUIREMENTS;
 DIR(J) = HHINC(J)/G(J);
PARAMETER ETA(K);
 ETA(K)$(ORD(K)LT CARD(I)) = SUM(J,EXINV(J,K)*DIR(J));
PARAMETER TYPEI(K)  TYPE I INCOME MULTIPLIERS;
 TYPEI(K)$(ORD(K)LT CARD(I)) = ETA(K)/DIR(K);
PARAMETER EMPL(J) TOTAL EMPLOYMENT BY SECTOR
 /AGR 14,MFG 286,TRNS 30,SVC 483/
PARAMETER EPSI(J) EMPLOYMENT PER DOLLAR PURCHASES;
 EPSI(J)$(ORD(J)LT CARD(I)) = EMPL(J)/G(J);
PARAMETER GAM(K);
 GAM(K)$(ORD(K)LT CARD(I)) = SUM(J,EXINV(J,K)*EPSI(J));
PARAMETER TYPIMP(K) TYPE I EMPLOYMENT MULTIPLIERS;
 TYPIMP(K)$(ORD(K)LT CARD(I)) = GAM(K)/EPSI(K);
DISPLAY
 HHINC,DIR,ETA,TYPEI,EMPL,EPSI,GAM,TYPIMP;
```

Other Agricultural Economics Research Publications

No. 89-5	Farmer Investment Behavior: A Review of Literature	Brian T. Brase Eddy L. LaDue
No. 89-6	Eighty Years of Change in Dairy Farming Dryden Township, Tompkins County, New York	B. F. Stanton
No. 89-7	1988 Budget Guide, Estimated Prices for Crop Operating Inputs and Capital Investment Items	Darwin Snyder
No. 89-8	A Microcomputer Program for Projecting Annual Cash Flows, Debt Repayment Ability and Proforma Financial Statements	Eddy L. LaDue David B. Cook
No. 89-9	Diversification of the Cheddar Cheese Industry Through Specialty Cheese Production: An Economic Assessment	John C. Martin David M. Barbano Richard D. Aplin
No. 89-10	Management Strategies to Improve Profitability on Limited Resource Dairy Farms: A Linear Programming Analysis	B. F. Stanton
No. 89-11	A Risk Evaluation of Groundnut Genotypes in Drought Prone Areas of India	Elizabeth Bailey Richard N. Boisvert
No. 89-12	Dairy Farm Management Business Summary, New York 1988	Stuart F. Smith Wayne A. Knoblauch Linda D. Putnam
No. 89-13	Economic Losses to New York's Dairy Sector Due to Mastitis	Heiko Frick William Lesser
No. 89-14	Bovine Somatotrapin: Its Impact on the Spatial Distribution of the U.S. Dairy Industry	Jork Sellschopp Robert J. Kalter
No. 89-15	Strategic Alternatives For the New York Apple Industry	Bruce Anderson
No. 89-16	Farming Alternatives: Experience in New York State	Lynn H. Miller Wayne A. Knoblauch Judy J. Green John R. Brake