CALPAN

A microcomputer model for calculating accounting prices with input-output techniques

User's manual

by

Elio Londero
Roberto Soto

Papers on Project Analysis No. 30, Revision 1

Inter-American Development Bank
Washington, D.C., 1998

Disclaimer

The Inter-American Development Bank has used its best efforts in preparing this manual and the models described in it. The Bank makes no warranties of any kind, either expressed or implied warranties of merchantability and fitness for a particular purpose with regard to this model or its documentation, and shall not be responsible in any way for the use and/or results obtained thereof.

Trademarks

LOTUS, 1-2-3 and SYMPHONY are registered trademarks of Lotus Inc. IBM-PC and PC-DOS and OS/2 are registered trademarks of International Business Machines Corporation. MS-DOS and Windows are registered trademarks of Microsoft Development Corporation.

Comments or suggestions

Please send your comments or suggestions on the program and/or its documentation to:

Elio Londero
Inter-American Development Bank
1300 New York Ave. N.W.
Washington D.C. 20577
U.S.A.

Requests

Copies of this manual may be requested from:

Publications Office
Inter-American Development Bank
1300 New York Ave. N.W.
Washington, D.C. 20577
U.S.A.

First Edition: 1991
Second revised edition: 1998

Este manual está disponible en castellano en el
Banco Interamericano de Desarrollo, Washington, D.C., 20577.

PREFACE TO THE REVISED EDITION

This revised edition only incorporates editorial changes. The user that prefers to continue using his/her old version of the manual will not be incurring in any significant cost and may benefit of his own notes, since no modification has been introduced to the program. Technical progress has helped the use of the program under DOS, especially as regards the inversion of large matrices, since price reductions for RAM memory allows for CALPAN to be used on a virtual disk with significant time savings. At the time, only limited experience is available under WindowsNT and OS/2, but no problem has been identified so far. Any information on that regard provided by the users would be greatly appreciated.

PREFACE TO THE FIRST EDITION

CALPAN is a set of programs for IBM-PC microcomputers or compatibles, designed to calculate accounting price ratios using input-output techniques. It also serves to calculate domestic resource costs of foreign exchange and to perform other useful preprogrammed operations. Finally, and in order to allow those calculations that have not been preprogrammed, it includes a program to perform arithmetic operations with matrices.

Two previous versions of CALPAN existed: Version 1.2 and Version 2.0. Considering that the first one used a different file format, this one includes a program to translate files from one version to the other (Chapter 16). Version 2.0 uses the same file format than this one, allowing for the direct use of files created with either version.

The reader who is interested only in the accounting price ratios will gain an understanding of the possibilities of this program by first reading Chapters 1 through 6, 10 and 11 and leaving the remainder for a second sitting. We recommend that the reader do a numerical example while reading the manual. For that purpose, the one presented in Powers (1981, Chap. 2, figures 2.18 and 2.19) or one created by the reader can be used.

The first version of the programs DEFINE, MATRANS, CALCULAG, FUNCION and

RPC was prepared initially by Terry Powers and Roberto Soto. The development of these programs, along with the design and development of the remainder of the system was done by Elio Londero and Roberto Soto. A preliminary version of the CALPAN system was used to calculate accounting prices for Mexico (BID-NAFINSA, 1986) and Uruguay (Flament, 1987).

The authors would like to thank Terry Powers, Miguel Flament, Roberto Fernández, Eduardo Barbieri and Rafael Cubillos for their help in improving this program. Thanks are also due to María Susana Arano, Andrés Escalante, Marie Maberg, Fernando Quevedo, and especially to Rafaela Varela, for their skills and patience in finding program errors. In addition we would like to thank Joel Rothstein for his translation of this manual. It goes without saying that the authors are responsible for the remaining errors and would appreciate hearing from the users concerning those they may encounter.

TABLE OF CONTENTS

CHAPTER 1
ACCOUNTING PRICE RATIOS

This chapter will present, in synthetic form, the logic underlying the principal objective of the CALPAN system: the calculation of accounting price ratios using input-output techniques. Therefore, we will not analyze the concept of accounting price ratios nor the preparation and use of intersectoral relations matrices for this purpose. If you are interested in these subjects, consult Scott et al. (1976), Powers (1981) or Londero (1987, 1989, 1992).

The accounting price ratio of a good or service i, that we represent by apr_i, is the ratio of its accounting price to its market price. The principal objective of using input-output techniques for this calculation is to capture the "backward" linkages originating in an excess demand for goods that are nontraded and produced at the margin, i.e. an additional demand for that good is satisfied by additional domestic production. Let

$$\Sigma_i A_{ij} + \Sigma_h F_{hj} = Q_j p_j \qquad [1.1]$$

be the composition of the marginal cost of producing the quantity Q_j of good j, where A_{ij} is the intermediate demand of marginally produced inputs i, and F_{hj} is the intermediate demand of nonproduced inputs and transfer payments h, both needed to produce Q_j. The system of equations describing the relations between products (j) and inputs (i,h) can be expressed as

$$\left\{ \begin{array}{l} \Sigma_i A_{i1} + \Sigma_h F_{h1} = Q_1 p_1 \\ \quad . \\ \quad . \\ \quad . \\ \Sigma_i A_{in} + \Sigma_h F_{hn} = Q_n p_n \end{array} \right. \qquad [1.2]$$

and presented as in Table 1.1. This type of matrix, which contains the basic information of the intersectoral relations, will be called a "data matrix", and will be called "type D" by the program.

The same group of intersectoral relations can be expressed in coefficients per unit of gross value of production by dividing each equation by $Q_j p_j$. By so doing, we obtain

Table 1.1. Representation of the intersectoral relationships

A_{11}	A_{12}	A_{13}	...	A_{1n}
.	.	.		.
.	.	.		.
.	.	.		.
A_{n1}	A_{n2}	A_{n3}	...	A_{nn}
F_{11}	F_{12}	F_{13}	...	F_{1n}
.	.	.		.
.	.	.		.
.	.	.		.
F_{k1}	F_{k2}	F_{k3}	...	F_{kn}
Q_1 p_1	Q_2 p_2	Q_3 p_3		Q_n p_n

$$\left\{ \begin{array}{l} \Sigma_i\, a_{i1} + \Sigma_h\, f_{h1} = 1 \\ \quad . \\ \quad . \\ \quad . \\ \Sigma_i\, a_{in} + \Sigma_h\, f_{hn} = 1 \end{array} \right. \qquad [1.3]$$

where a_{ij} is the value of input i needed per unit of additional gross value of production of sector j, and f_{hj} is the value of the nonproduced input or transfer payment h per unit of additional gross value of production of sector j. This system may be presented as in Table 1.2. The corresponding matrices will be called coefficient matrices, and the program will call them "type C" matrices.

Using the relations in coefficient form, we can now write the system of equations to obtain the accounting price ratios as

$$\left\{ \begin{array}{l} \Sigma_i\, a_{i1}\, apr_i + \Sigma_h\, f_{h1}\, apr_h^f = apr_1 \\ \quad . \\ \quad . \\ \quad . \\ \Sigma_i\, a_{in}\, apr_i + \Sigma_h\, f_{hn}\, apr_h^f = apr_n \end{array} \right. \qquad [1.4]$$

Table 1.2. Representation of intersectoral relationships in coefficient form

a_{11}	a_{12}	a_{13}	...	a_{1n}
.	.	.		.
.	.	.		.
.	.	.		.
a_{n1}	a_{n2}	a_{n3}	...	a_{nn}
f_{11}	f_{12}	f_{13}	...	f_{1n}
.	.	.		.
.	.	.		.
.	.	.		.
f_{k1}	f_{k2}	f_{k3}	...	f_{kn}
---	---	---		---
1	1	1		1

and present it in matrix form as

$$[apr_i]\,[a_{ij}] + [apr_h^f]\,[f_{hj}] = [apr_j] \qquad [1.5]$$

or as

$$\mathbf{apr\ A} + \mathbf{apr}^f\ \mathbf{F} = \mathbf{apr} \qquad [1.6]$$

where

apr = $[apr_i]$ = row vector of the accounting price ratios of the produced goods

A = $[a_{ij}]$ = transaction matrix

$\mathbf{apr}^f = [apr_h^f]$ = row vector of the accounting price ratios of the nonproduced inputs and transfer payments

F = $[f_{hj}]$ = matrix of the direct requirements of nonproduced inputs and transfer payments

From [1.6] the **apr** vector can be written as

$$\mathbf{apr} = \mathbf{apr}^f\ \mathbf{F}\ (\mathbf{I} - \mathbf{A})^{-1} \qquad [1.7]$$

This algebraic expression can be interpreted in the following manner. The matrix

$$\mathbf{G} = \mathbf{F}\,(\mathbf{I} - \mathbf{A})^{-1} \qquad [1.8]$$

contains the total (direct and indirect) requirements of nonproduced inputs and transfer payments per unit of gross value of production. The program will call this type of matrix "type G". Finally, the product

$$\mathbf{apr} = \mathbf{apr}^{f}\,\mathbf{G} = \mathbf{apr}^{f}\,\mathbf{F}\,(\mathbf{I} - \mathbf{A})^{-1} \qquad [1.9]$$

corrects the total requirements of nonproduced inputs and transfer payments by the ratios of the accounting to the market prices, and sums them for each product *j*, obtaining the corresponding marginal costs at accounting prices. As we will see later, CALPAN has specific programs to create the data matrix, calculate the coefficient matrix [**A** | **F**], calculate the matrix $\mathbf{G} = \mathbf{F}\,(\mathbf{I} - \mathbf{A})^{-1}$, and calculate the vector $\mathbf{apr} = \mathbf{apr}^{f}\,\mathbf{G}$.

There are cases in which the apr_h^f of some nonproduced input depends in turn on the *apr* of some produced good. In these cases, CALPAN calculates the *apr* by iteration. For that purpose we have to provide the program with the functional relations between each apr_h^f and the corresponding apr_j.

$$apr_h^f = apr_h^f(\mathbf{apr}) \qquad [1.10]$$

CALPAN will start by assigning an initial or "seed" value to each apr_h^f. If $\mathbf{apr}_0^{f}$ represents the vector containing the values of the apr_h^f that are exogenous to the system and the "seed" values of the apr_h^f functions, we will obtain the first approximation of the vector **apr**, which we will call $\mathbf{apr}_1$, as

$$\mathbf{apr}_1 = \mathbf{apr}_0^{f}\,\mathbf{G} \qquad [1.11]$$

The program calculates the vector $\mathbf{apr}_1$, and then uses it to calculate a new vector $\mathbf{apr}_1^{f}$ according to [1.10]. With the latter, it returns to calculate the vector **apr** as

$$\mathbf{apr}_2 = \mathbf{apr}_1^{f}\,\mathbf{G} \qquad [1.12]$$

and repeats the procedure until the differences

$$\mathbf{apr}_n - \mathbf{apr}_{n-1}$$

$$\mathbf{apr}^f_n - \mathbf{apr}^f_{n-1}$$

are smaller than a predetermined level. In other words, this procedure continues until the solution converges (for a predetermined tolerance). CALPAN provides a program to define the functional relations $apr^f_h = apr^f_h(\mathbf{apr})$, and the program that calculates the vector **apr** will do the number of iterations needed for the result to converge.[1]

[1] Lucking (1993) proposes an alternative method.

CHAPTER 2
THE ORGANIZATION OF THE CALPAN SYSTEM

2.1 Principal Characteristics

Programs

CALPAN was designed to facilitate the construction of intersectoral relations matrices (IRM) and the calculation of *apr* according to the methodology described in the previous chapter. Its structure shows the sequence of operations and calculations needed and is organized in such a manner that the user is able to return to previous points in the process in order to review or update his/her data as necessary. CALPAN can operate with matrices up to 300 × 300.

The system consists of a group of programs, described in the ensuing chapters, whose names and functions are the following:

CALPAN The system program selector or main menu. It is the program that runs the remaining programs.

DEFINE Serves to define for the system an IRM for the first time by naming its rows and columns. It also serves to edit these names.

MATRANS Permits the entry of the numerical values or elements of an IRM, the reordering of its columns or rows, performs calculations between these columns and rows, creates new ones, erases existing ones, calculates the coefficients matrix, and others.

CALCULAG Calculates the matrix $(\mathbf{I} - \mathbf{A})^{-1}$ and then premultiplies it by matrix **F**, obtaining the matrix of direct and indirect requirements of nonproduced inputs and transfer payments at market prices (type G matrix).

PRECIOS Converts matrices of total requirements of nonproduced inputs and transfers from producer's prices to user's prices and vice versa.

AJUSTEG Serves to update the type G matrix for price changes of the nonproduced inputs.

BIENES Calculates matrices of total requirements of nonproduced inputs and transfer payments (type G) for products or sectors that were not included in the original IRM.

FUNCION Allows the specification of functional relations between the apr^{f}_{h} and the $apr_{,j}$ equation [1.10], in preparation for the calculation of vector **apr**.

RPC Calculates vectors **apr** and $\mathbf{apr}^{f}$, according to equation [1.7] or following the "iterative" process if necessary.

CALCID Calculates the domestic costs of foreign exchange (*drc*) starting from a vector $\mathbf{apr}^{f}$ and from two matrices of total requirements of nonproduced inputs and transfer payments (type G); one containing the total requirements corresponding to domestic production costs and the other containing those corresponding to the equivalent export or import price.

ARIMAT Can be used to perform arithmetic operations with matrices, to extract partitions from a matrix, and to concatenate matrices.

IMPRIME Permits the user to see the matrix on the screen or to print it in whole or in portions.

ARCHIVOS Permits the user to manage the files recorded on magnetic disc that contain the IRM without having to exit to the operating system.

TRADUCE Converts the files recorded on magnetic disk in the CALPAN formats to the DIF and ASCII formats, and vice versa. It also converts CALPAN 1.2 files to CALPAN 3.0, and vice versa.

Function keys

On the last line of the screen, all programs show a group of functions that can be executed when the user presses the corresponding "function key." These function keys are those designated as F1, F2, ..., F10 on the keyboard. When a function is available in more than one program, it is always assigned the same function key.

When a program has more functions than can be displayed on one line, the "F10 others" function appears. This permits the user to switch from one to another set of functions, whose names always appear on the last line of the screen.

Operational modes

The programs DEFINE, MATRANS and IMPRIME can be operated in two mutually exclusive modes called COLUMNS and ROWS, respectively. Any operation performed on the IRM while using one of these programs will affect the columns if the program is in the COLUMNS mode, or the rows if the program is in the ROWS mode. The function "F5 Mode" permits the user to quickly switch from one mode to the other by pressing the F5 key.

Online help

CALPAN provides online help. By pressing the F2 key, it displays a brief text explaining the functions that can be executed, as well as the page number in this manual where these functions are described in detail. The help texts are recorded on disk files under the names AYUDAxxx.HLP. If CALPAN does not find the help file, then it will display a message indicating that the file in question does not exist.

Types of matrices

CALPAN identifies the matrices it uses by types corresponding to the function that they have in the calculation of the apr. Therefore, by knowing the matrix type, we can interpret its elements. These matrix types are identified with a letter with the following meaning.

D These are the matrices that contain (or will contain if it is being defined or constructed) the data that the economist will use in order to calculate the coefficients matrix.

C These are normalized or coefficients matrices that CALPAN calculates from data matrices using the function "F7 Norm" of the MATRANS program.

T This type of matrices are those obtained as a result of the operation $(\mathbf{I} - \mathbf{A})^{-1}$, also known as the "Leontief inverse", which is calculated by the CALCULAG program. The AJUSTEG program also generates type T matrices containing the elements of

$$\text{type T matrix} = (\mathbf{I} - \mathbf{A} - \mathbf{T}_1 - \ldots - \mathbf{T}_n)^{-1}$$

where $\mathbf{T}_1$, $\mathbf{T}_2$, ..., $\mathbf{T}_n$, are diagonal matrices (see Chapter 8).

G These are the matrices of total (direct and indirect) requirements of nonproduced inputs and transfer payments calculated by the CALCULAG program. They correspond to the matrix $\mathbf{F}(\mathbf{I} - \mathbf{A})^{-1}$. In order for the CALCULAG program to calculate a type G matrix, a type C matrix must exist. The programs PRECIOS, BIENES and AJUSTEG also generate type G matrices starting from an original.

I These are not really IRM, but $\mathbf{apr}^f$ vectors calculated by the RPC program or created by the user using the DEFINE and MATRANS programs. They are stored internally as matrices of dimensions $k \times 1$ to facilitate printing them. The RPC program names the only column of these matrices "Values of the APR."

S Similar to type I, these matrices are not IRM, but vectors of accounting price ratios (*apr*). They are also calculated by the RPC program and stored internally as matrices of dimensions $n \times 1$. The program names the only column of these matrices as "Values of the APR." The name of the rows corresponds to the sector whose *apr* they represent. The AJUSTEG program generates type S matrices containing the values of the endogenous correctors or ratios between the prices of the produced goods (see Chapter 8). The CALCID program also generates type S matrices that contain the domestic resource costs of foreign exchange corresponding to the sectors included in the calculation (see Chapter 12).

Although CALPAN allows the user to create matrices of any type (A, B, ...,Z), those programs performing preprogrammed operations assume that D, C, T and G type matrices correspond to the preceding definitions. Consequently, if the user creates a type C matrix that is not a coefficient one and uses it as source in CALCULAG, he will obtain a type G matrix that will not be one of total requirements of non-produced inputs and transfers.

Precision Level

Internally, CALPAN operates with an 11 digits precision level for matrix elements. Those eleven digits are stored in memory using 6 bytes, and each numerical entry is represented by a pair of numbers called mantissa and exponent. Mantissa is the name for the eleven significant digits, while the exponent is the power of ten that multiplies the mantissa. For example, the number

15689.345 will be represented in memory as 1.5689345000×10^4. That allows CALPAN to operate with numbers X such that

$$1.0 \times 10^{-37} < X < 1.0 \times 10^{38}$$

In other words, the greatest number CALPAN will be able to represent is one followed by 38 zeroes, while the smallest would be the decimal point followed by 37 zeroes and a number one. But, only the first eleven digits of these numbers, those in the mantissa, will be significant. The user may enter a number with more than 11 significant digits, but CALPAN will represent only the first eleven, implicitly taking the rest as zeroes.

2.2 How to Start the CALPAN System

Before using the system for the first time, it is recommended that you i) writeprotect the original CALPAN diskettes; and ii) make a copy of the originals, store them in a safe place and work only with the copies. Hence, it will always be possible to get the programs from the original diskettes if there is some problem with the copies. If you do not know how to copy diskettes, consult your system analyst.

There are many ways to start the CALPAN system depending on whether the programs are stored on a diskette or on a hard disk (see Appendix A for instructions on system installation). If you are working with two disk drives, at least one of them must have a 1.2Mb capacity. In such a case, start by initializing the system ("warm boot"), i.e. place the working copy of the operating system start up diskette (containing a CONFIG.SYS file with the specifications indicated in Appendix A) in drive A, and simultaneously press the Ctrl-Alt-Del keys. When the A> prompt appears, place the diskette that contains the CALPAN system in drive A and type the following:

CALPAN<ENTER>

It is not necessary to type the characters "<ENTER>". Simply press the Enter key (or "Return" on some keyboards). At this point, the CALPAN MAIN MENU will appear on the screen and

you may begin working. If you are working on the hard disk, the manner in which the CALPAN system is called depends on how it was installed. In this regard, consult Appendix A or your system analyst.

WARNING: It is recommended to always starts by initializing the system ("warm boot") by simultaneously pressing the Ctrl, Alt and Del keys. This avoids problems that can be created by "residues" left in memory by a previously used program.

CHAPTER 3

THE MAIN MENU

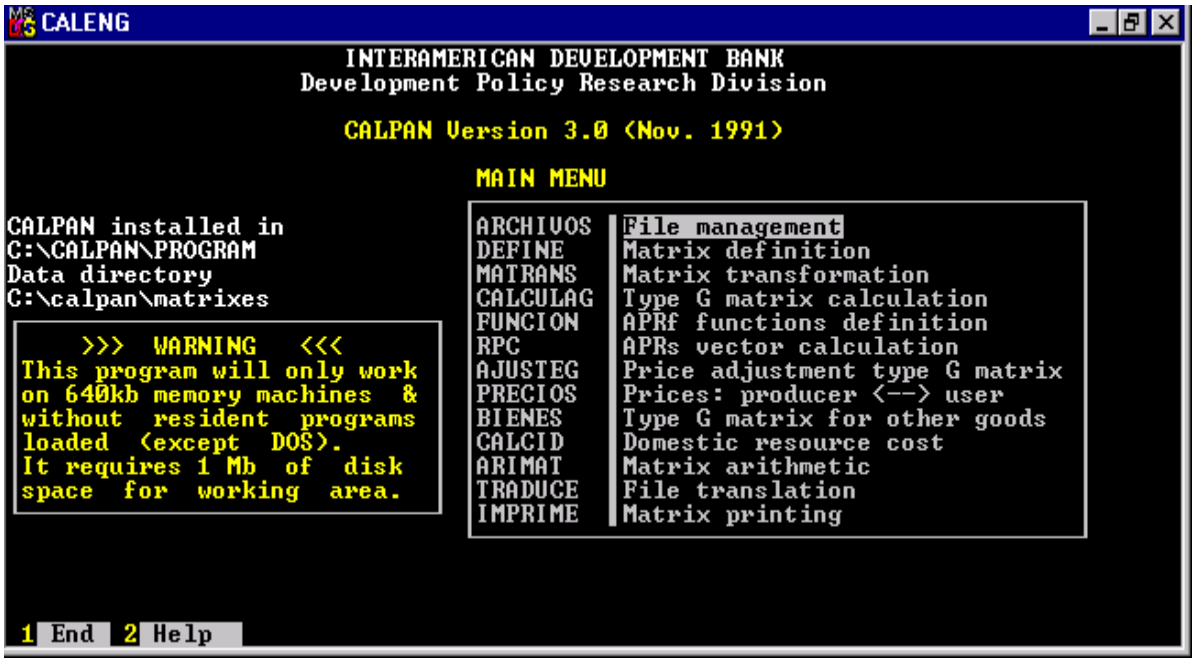
The main menu is a program selector that allows the user to call any of the programs available in the system. The names of those programs are displayed in a box on the screen, followed by a brief description of the operations they perform.

The disk drive and directory names that appear to the left of the box indicate the location of the system and that of the data (for more information about these directories, see Appendix A, "Installation instructions". The names appear in the following manner:

```
CALPAN installed in
disc:\directory name

It will search for data in:
disc:\directory name
```

Upon starting your work, CALPAN will always assume that the data is in the directory from which it was called. For example, if CALPAN were installed in the directory C:\CAL



Screen 1. Main menu

and was called from there, it will assume that the data is also stored in this directory. If the data is in a directory different from which CALPAN was called, then the user must indicate the drive and subdirectory where it is located using the Change_Dir function from the ARCHIVOS program (see Chapter 14 and Appendix A). In order to save the data in a subdirectory (which is recommended), this subdirectory must be created by using the command MD (Make Directory) of the operating system.

The screen displays a warning message concerning memory and disk space requirements. If these conditions are not met, the program will not function adequately (see Appendices A and B).

The user can select the program found at the cursor (name that appears in reverse video on the screen) by pressing the <ENTER> key. To move the cursor from one program to another, use the vertical arrows.

While CALPAN is the main menu, the F1 key is used to exit the system and return to the operating system. Upon exiting to the operating system, it will point to the last disk drive and subdirectory that CALPAN used (the data subdirectory). In the other programs, upon pressing the F1 key while the "Command-->" message is present, the program in question will return to the main menu. If F1 is pressed while writing text, data, or names of files, the program will return to "Command-->."

By pressing the F2 Help key, CALPAN will display a help text containing a description of the keys to operate the main menu as well as a page number from this manual where the respective explanation is found. After reading the help text, the user can return to the main menu by pressing the F1 key. If the file AYUDA100.HLP does not exist, the system will display a message indicating this and will not display any such text.

CHAPTER 4
THE DEFINITION OF MATRICES

The DEFINE program is used to create matrices. This program only permits the user to define the type of matrix to be created and enter the *names* of its columns and rows. The program operates in four different "states", each one with a specific purpose. These "states" are:

1. START
2. ENTRY
3. MODIFY
4. COMMAND

The "states" ENTRY and MODIFY operate in two mutually exclusive modes:

1. COLUMNS mode
2. ROWS mode.

The mode and the current state of the DEFINE program is always displayed on the screen. Also displayed is a window that shows the disk drive and directory where DEFINE will search for and save the matrices.

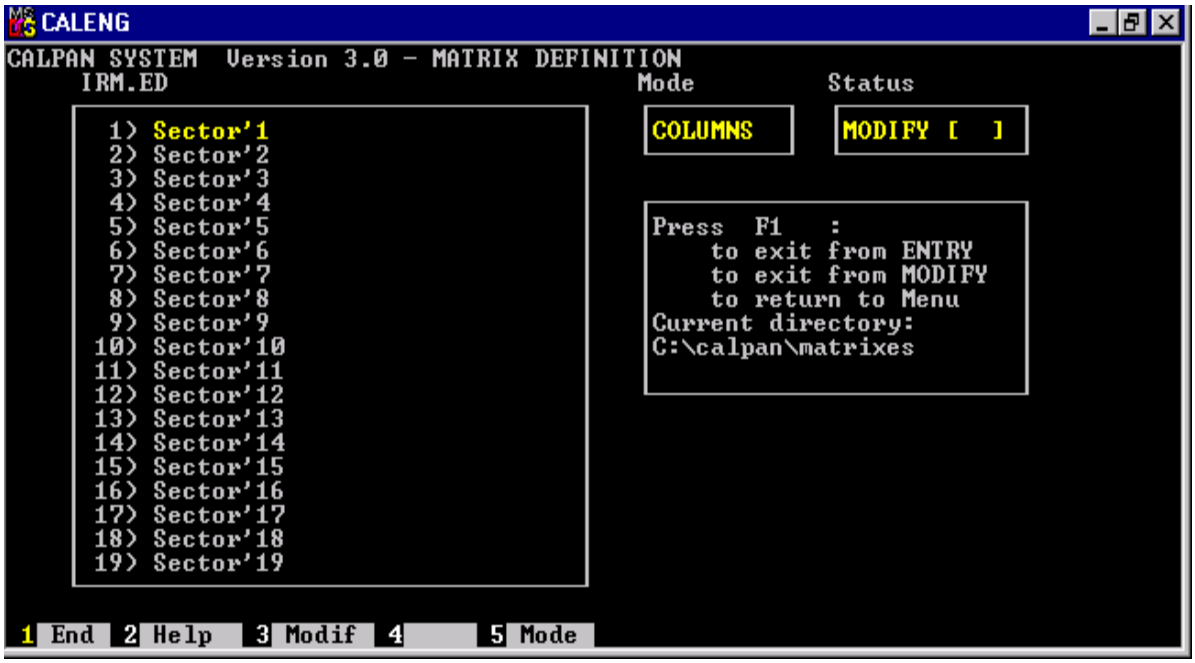
4.1 The START State

Upon selecting the DEFINE program, the program will appear in the START state, which is the one used to determine the name and the type of matrix to be defined. In this state, the program displays the following message:

```
[Press F2 for Help]  Type of Matrix (A, B, C, ..., Z) -->_
```

If you press the F1 key while in the start state, DEFINE returns to the main menu; if F2 is pressed, the program displays a help text with references to page numbers in this manual.

In order to define a matrix, it is necessary to start by defining its type, which is done by pressing one of the letter keys (A, B, C, ..., Z). CALPAN permits the creation of matrices of any type, but all the preprogrammed operations assume that type D, C, T, G, and I matrices correspond to data, coefficients, total production requirements, total non-produced inputs



Screen 2. Matrices definition program (DEFINE)

requirements, and *apr* for non-produced inputs, respectively (see Chapter 2). Next, DEFINE will ask the user to name the matrix that he/she wishes to define. This name may not contain more than eight numbers and/or letters. If the matrix already exists on the disk and data directory designated upon starting, (see Chapter 1) DEFINE will display a message while it reads the matrix from the disk and will proceed immediately to the MODIFY state in the COLUMNS mode. If the matrix is not there (the matrix was not defined or calculated before), DEFINE will ask if you wish to create it, with the message:

```
Does not exist <name of matrix>.XX  Do you want to create it?  (Y/N)
```

If the N option is selected, the matrix will not be defined and DEFINE will return to the main menu. If the matrix does not exist on the disk and the Y option is selected, the matrix will be defined and DEFINE will immediately pass to the ENTRY state in the COLUMNS mode.

4.2 The ENTRY State

DEFINE, in the START state, automatically activates the ENTRY state when a matrix

does not exist on the disk or the designated data subdirectory. Thus, you can start by registering (entering) the *names* of the matrix's rows and columns. This state is always presented in the COLUMNS mode.

The ENTRY state positions the program in the COLUMNS mode, presents "NAMES OF COLUMNS" in reverse video, places the column counter at 1 (showing it on the last line of the data entry window) and waits for the user to enter the name of the first column. In order to name a column or row, type the row name and press <Enter>. Next, ENTRY will ask the user the following row name, automatically increasing the counter of columns and rows. To finish entering row and column names, press the F1 key.

The row and column names should always begin with a letter, although it may contain numbers and apostrophes (') in other places, and must be no longer than 27 characters. If the space bar is used, CALPAN will show an apostrophe in its place (this rule applies to the entire system where names of rows and columns are introduced). By doing so, the names of columns and rows are defined with just one "word." When printed (on hard copy), blank spaces will appear in place of the apostrophes.

In order to edit these names, use the BACKSPACE key. This also applies in any other situation when entering numbers or letters and is valid for all the programs in this system.

Upon finishing the entry of the column names, DEFINE will ask the user if the names of the columns are to be copied (repeated) in the corresponding row positions. If the user chooses the Y option, the names of the columns will be copied in the corresponding row positions and ENTRY will continue requesting row names starting from the next available position ($name_{n+1}$ in Table 4.1). This permits the construction of a rectangular matrix composed of a transaction matrix, that *should* have the same row and column names, and by the matrix of nonproduced inputs and transfer payments thus creating an IRM like the one shown in Table 4.1.

If the user chooses not to repeat the names of the columns in the corresponding rows, the entry of row names begins at the first position ($name_1$, in Table 4.1). To finish entering row names, press the F1 key and DEFINE will go to the COMMAND state, in the ROW mode.

If the user presses the F1 key to finish the entry of column or row names and *has not defined at least one name for each row and column*, the program will display an error message

Table 4.1. Definition of intersectoral relations matrices

<table>
<tr><td></td><td>name1</td><td>name2</td><td>...</td><td>namen</td></tr>
<tr><td>name1</td><td colspan="4" rowspan="4">matrix A</td></tr>
<tr><td>name2</td></tr>
<tr><td>.
.
.</td></tr>
<tr><td>namen</td></tr>
<tr><td>namen+1</td><td colspan="4" rowspan="4">matrix F</td></tr>
<tr><td>namen+2</td></tr>
<tr><td>.
.
.</td></tr>
<tr><td>namen+k</td></tr>
</table>

and will continue asking for column and row names. It is necessary to define at least one name per row and column.

4.3 The COMMAND State

While in the COMMAND state, DEFINE permits the user to select any of the function keys that are shown on the bottom line of the screen (F1, F2, F3, F5). The use of the first two has already been described and the use of the third will be explained in the next section. By pressing F5 Mode, DEFINE switches from one mode to another (from COLUMNS to ROWS and vice versa).

In order to return to the main menu, press the F1 key. By doing so, DEFINE saves the most recently defined (or modified) matrix on the disk, in a file which has the same name as the matrix with an extension .E<type>, where <type> is A, B, C, ..., Z.

4.4 The MODIFY State

This state permits the editing of the matrix row and column names, depending on the mode DEFINE is in at that moment. To switch to the MODIFY state press the F3 key while

DEFINE is the COMMAND state. The program will then show the row and column names in the window, placing the cursor at the first name. The name that can then be edited (modified) is the one at cursor. To place the cursor at another name, use the vertical arrows. If the number of columns and rows exceeds the space on the data entry window, MODIFY will scroll the list of matrix names on the window when the user reaches the upper or lower limits, such that the user can always see 19 names on the screen.

DEFINE does not permit the elimination or addition of names to a matrix. It only permits the creation of a matrix (by registering the names of the columns and rows) or the editing (modifying) of the names of an existing one. The creation or elimination of rows and columns of a matrix can be done with the MATRANS program. To change a name, position the cursor at the existing name and type the new name completely, followed by <ENTER>. MODIFY will display the new name in the same position. If the <ENTER> key is not pressed after changing the name, but the user presses an arrow or F1 instead, the name will not be changed, regardless of what is indicated on the screen. To actually change the name of a matrix, the <ENTER> key must be pressed.

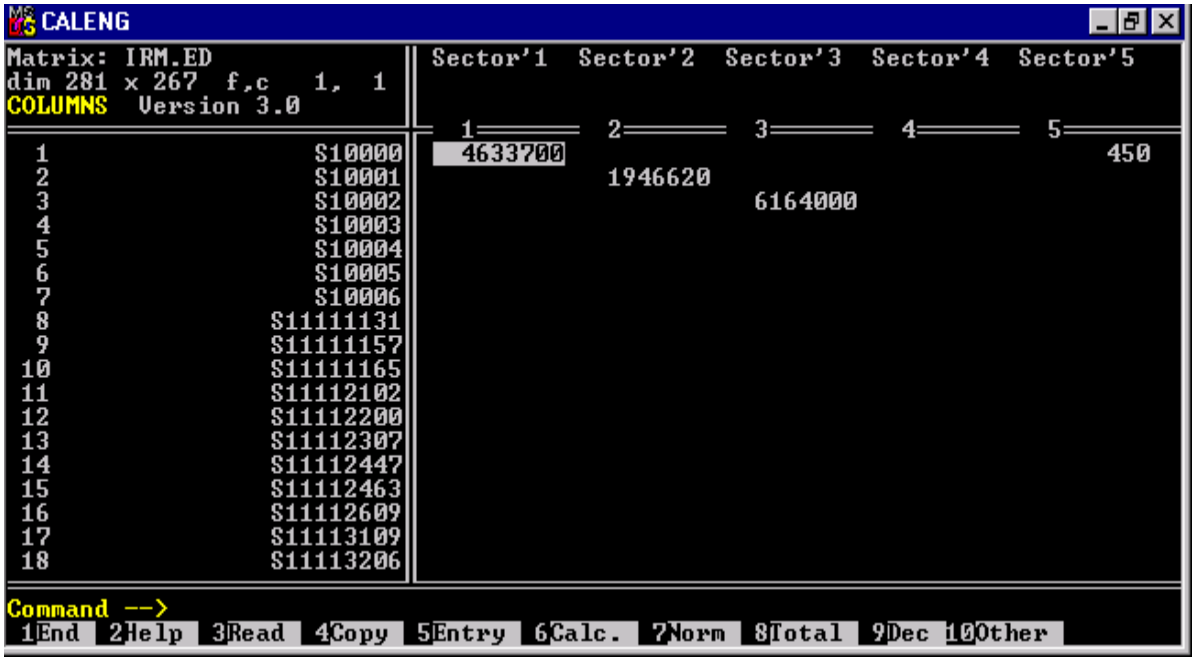
To exit the MODIFY state press the F1 key and DEFINE will return to the COMMAND state. Once the matrix is defined by entering the names of its rows and columns, you need to enter numerical values or elements of the matrix. This task can be completed by using the MATRANS program.

CHAPTER 5

THE CONSTRUCTION AND TRANSFORMATION OF MATRICES

The principal objective of the MATRANS program is to permit the entry of the elements of a previously defined matrix (see Chapter 4). MATRANS also allows the calculation of coefficient matrices, and includes a large number of functions to calculate, create, erase, and move columns and rows.

Upon starting it, MATRANS displays a window where the user can see part of the matrix on the screen and move around it by using the keyboard arrows or the "F9" function key. MATRANS also displays a fictitious matrix of one column and one row (both with the name "undefined"). The numerical cursor is the rectangular light that appears on the upper left corner of the space reserved for the numeric entries. The penultimate line of the screen is used to ask the user for commands (operations to be completed), as well as to display messages or ask the user to enter numbers or other information. The functions executed by pressing the function keys are shown on the last line of the screen. Before executing any function, the user must load a



Screen 3. Program for the construction and transformation of matrices (MATRANS)

Table 5.1. Possible MATRANS operations

GROUP 1	GROUP 2
F1 Return to the main menu	F1 Return to the main menu
F2 Provide Help	F2 Provide Help
F3 Load a matrix to memory	F3 Read a block from a file
F4 Save a matrix on disk	F4 Copy a block to a file
F5 Change the mode (Rows<>Cols)	F5 Enter numerical data
F6 Move rows or columns	F6 Calculate rows or columns
F7 Erase rows or columns	F7 Normalize matrix in memory
F8 Create rows or columns	F8 Add columns (totals)
F9 Move the numerical cursor	F9 Specify number of decimals
F10 Change to set 2	F10 Change to set 1

matrix to memory using the F3 Load function, which is described later.

MATRANS provides two sets of function keys; the one in use appears on the last line of the screen. To switch between the two sets, press the "F10" function key. Table 5.1 shows every type of operation that the program can perform along with the corresponding function key.

5.1 First Set of MATRANS Function Keys

F1 Exit (Sets 1 and 2)

By pressing F1 while in the Command state, MATRANS returns to the menu. Before doing so, the program will ask if you want to save the matrix with which you have been working. If you have not previously saved the matrix by pressing F4 Save and want to save the changes just made, press the "Y" key. If F1 is pressed while writing text, data, or names

F2 Help (Sets 1 and 2)

The F2 key permits the user to see the help texts and then return to normal MATRANS operations (see Chapter 2).

F3 Load

The F3 Load function allows the user to load a matrix saved on magnetic disk. Upon executing this function, the program will ask which type of matrix is to be loaded (A, B, ...,Z) as

well as its name. If the matrix does not exist, an error message will appear on the screen. MATRANS provides information about the matrix currently in use on the upper left side of the screen. There you can see the name of the matrix, (file name) its dimensions (dim *n,m*), the position of the numerical cursor (*r,c*) the operation mode (ROWS or COLUMNS), and the version of the program you are using.

A matrix can be loaded to memory whenever MATRANS asks for a command, even if there is a matrix currently in memory. Therefore, the user should always be certain that the matrix shown on the screen has been saved (if so desired) before loading another one. Thus, only one matrix is in use at a time.

F4 Save

This function permits the user to save the matrix currently in memory on magnetic disk. This matrix can be saved on magnetic disk with a different name than the original, but the program always records the same type of matrix (A, B, ..., Z) that is in memory. By using the F4 Save function, MATRANS will ask for the name of the matrix to be saved. If the user does not specify a matrix name (presses only <ENTER>), the matrix in memory will be written over the matrix of the same name originally loaded (except if it has been normalized, see "F7 Norm"). If the user specifies a name, the matrix will be written to disk in a file with that name. In both cases, it is possible that a matrix with the same name may already exist in the data directory. If that is the case, MATRANS will display the message:

```
XXXX.XX already exists (R)eplace or (C)ancel? (R/C) -->_
```

and the user then has the opportunity of canceling the operation (by pressing the C key) or replacing the data in the original file with that in memory (by pressing the R key).

WARNING: If there is not enough disk space to save the new matrix, the program will abort the operation and the data in memory will be lost. The user should verify the availability of disk space *before entering the CALPAN system.*

F5 Mode

The majority of the MATRANS function keys operate in two mutually exclusive modes: ROWS and COLUMNS. The mode in use is always shown on the upper left corner of the screen. By pressing the F5 key, MATRANS switches from one mode to the other. Before completing any operation that depends on the mode, the user must verify that MATRANS is in the desired mode. The following functions, which will be described later, depend on the mode.

GROUP 1	GROUP 2
F6 Move rows or columns	F3 Read row block
F7 Erase row or column	F4 Copy block on to file
F8 Create rows or columns	F6 Calculate rows or columns

F6 Move

This function allows the user to move rows and columns. Upon pressing the F6 key, MATRANS will ask for the name of the row or column to be moved. You should then type the name of the desired row or column. If MATRANS does not find one with this name, it will display an error message. In order for MATRANS to find a name on the matrix, it must be typed exactly as the one in memory, including capital letters, small letters, commas and any other characters used.

In order to preserve the matrix structure, each time a row or column is moved, MATRANS will also move the row or column with the same name to the corresponding position, provided that they are in the same position before the change is requested. In other words, if upon moving the row "sector 7" from the seventh to the ninth row, MATRANS finds the column "sector 7" in the seventh column, both will be moved from the seventh to the ninth column and row, respectively. If the names and positions are not identical, only the column or row will be moved.

F7 Erase

This function erases rows and columns from memory. The row or column erased is the one at the cursor. By pressing the F7 key, MATRANS gives the following message:

```
Erase Column (Row) <name> (Y/N) -->_
```

If the user answers no (N key), the program will not erase it and the row or column will remain on the screen. If the user answers yes (Y key) and a row or column with that name exists, the program will ask:

```
Erase Column (Row) <name> also? (Y/N) -->_
```

to which the user may respond yes or no. This is done to preserve the structure of the IRM. In any case, MATRANS will only erase from the matrix the row or column specified by the user.

F8 Create

The F8 function incorporates rows or columns to the matrix in memory. The newly created row or column will have zeroes in all positions and will appear in the position immediately following the cursor, displacing the existing rows and columns to create space for the new one. Therefore, if the cursor is in column 9, the new column will be created in position 10. By pressing the F8 key, MATRANS will ask you to name the row or column to be created. If the name is one that already exists, MATRANS will give the message:

```
<name> already exists, do you wish to create it? (Y/N) -->_
```

giving the user the opportunity to verify the spelling of the name. It is recommended not to create rows or columns with previously used names. Upon creating the new row or column, in order to maintain the structure of the IRM, MATRANS will ask:

```
Also create column(row) <name> also? (Y/N) -->_
```

Then the user can decide whether or not to create the corresponding row/column.

F9 Cursor

This function allows the user to place the cursor at any position on the matrix. This is especially useful when working with large matrices. By pressing the F9 key while the first set of

functions is in use, MATRANS will display the following message:

```
row,column -->_
```

The user must then specify the new position of the cursor. This is done by typing the number or name of the row and that of the column separated by a comma ",". The name or number can be omitted, but *not* the comma. Blank spaces are not allowed. If a name is not specified or a zero is entered, MATRANS will not change the position of the cursor. Examples of this function are given in Table 5.2.

5.2 Second Set of MATRANS Function Keys

F3 Read

During the construction of an IRM, it may be necessary to transfer row or column data from a disk file to the matrix in memory. To facilitate such a transfer, MATRANS provides two complementary functions, "F3 Read" and "F4 Copy". The F3 function permits the transfer of an adjoining block of rows or columns (depending on which mode MATRANS is in) from disk to memory. This block must be contained in a matrix that is found in the current directory and is of the same type (A, B, ..., Z) as the one in memory. Since a block may be an entire matrix, two or more smaller matrices may be combined to form a larger one.

By pressing the F3 key, MATRANS will ask for the name of the matrix from which to read with the following message:

```
READ from matrix? -->_
```

If the user presses F1 or <ENTER>, MATRANS will ask for another command. If the user specifies the name of a file followed by <ENTER>, MATRANS will verify that this file exists. If it does not exist in the data directory, an error message will be displayed. If the file exists, MATRANS will ask the user to enter the name of the first row(column) block to be read. Once the user has entered the first name, MATRANS will verify that such a row (column) exists; if it does not exist, an error message will be displayed and it will again ask for the first name

Table 5.2. Examples of the use of the F9 function

If the user specifies:	The cursor will be moved to:
f,c -->foreign'exchange,3	row "foreign'exchange", col 3
foreign'exchange,ccf	row "foreign'exchange", col "ccf"
0,5	same row, col 5
,5	same row, col 5
10,ccf	row 10, col "ccf"
ccf,0	row "ccf", same column
ccf,	row "ccf", same column
,	same row, same column
0,0	same row, same column
4,20	row 4, col 20

the first name. If at this moment the user presses the F1 key, the operation will be canceled. If MATRANS finds the first name of the block in the file, it will ask for the last name. If the user presses <ENTER> at this moment, MATRANS will assume that the block consists of just one row(column), will read it from the file, and will incorporate it to the matrix in memory. If F1 is pressed, MATRANS will return to COMMAND. When the user enters the last name, the program will verify that this row or column exists. If it does not exist, an error message will be displayed and the program will again request the final name until:

a) the entered name exists;

b) <ENTER> is pressed, in which case it will read the one column or row block and will incorporate it to the matrix in memory; or

c) F1 is pressed to return to COMMAND.

The blocks of columns and rows are loaded on memory beginning in the space after the last column/row. If the columns/rows are used only for interim calculations during the design of the matrix, they can be left in those positions. If, however, the user wishes to place these in other positions, the "F6 MOVE" key must be used. If upon reading a block of columns/rows, the number of columns/rows does not coincide with the dimensions of the

matrix in memory, MATRANS a) will replace the number of missing columns/rows with zeroes if there are too few columns/rows; b) will cut off the extra columns/rows that are not on the matrix in memory.

F4 Copy

This function is complementary to "F3 Read". It is used to copy to a file (write to disk) an adjoining block of columns(rows) found on memory. By pressing F4, MATRANS will ask for the name of the first column or row of the block to be copied. If F1 is pressed the function will be canceled. If the user specifies a name of a column or row, MATRANS will verify that it exists in memory and will then ask for the last column or row name of the block to be copied. If at this moment the <ENTER> key is pressed, a block of one column or row will be copied. If the user enters another name, MATRANS will again verify that it exists in memory and the block will be copied. If the final name does not exist, an error message will be displayed and the program will continue requesting the final name until:

a) the entered name exists;

b) <ENTER> is pressed in which case the one column/row block is copied; or

c) F1 is pressed to return to COMMAND.

After requesting the first and last names of the block, MATRANS will ask the name of the file where it is to be copied. If the user does not enter the name of a file, the program will cancel the copying operation. If the file already exists, MATRANS will ask the user whether or not to replace it with the new block or to cancel the operation. A block of columns or rows written from memory to disk is copied as if it were an entire matrix even though it may consist of just one column/row.

F5 Entry

This function permits the numerical values of the cells (elements of the matrix) to be incorporated into the matrix in memory. Upon executing this function (by pressing the F5 key), MATRANS shows the symbol ">" on the penultimate line of the screen and the desired numerical values can then be entered. This is done by typing the desired numbers followed by

<ENTER> (they can be edited by using the <BACKSPACE> key before pressing <ENTER>). The number can be introduced in any of the forms accepted by MATRANS: exponents, whole, or decimal. If the user presses only the <ENTER> key, the value at the cursor will not change. After entering a value, MATRANS automatically moves the numerical cursor to the next position (down in the columns mode and to the right in the rows mode); when completing a row or column, MATRANS will switch to the beginning of the next one.

If the user wishes to enter data starting from a specific matrix cell, it is first necessary to put the cursor in this cell by using the arrows or the F9 Cursor function before executing the F5 Entry function. To finish entering numerical data, press F1, and MATRANS will ask for a new command.

F6 Calc

This function permits the calculation of a column/row as a linear combination of others. Therefore, before executing this function, the user must be sure that MATRANS is in the correct mode (ROWS or COLUMNS). By pressing F6, MATRANS will give the following message.

```
Write arithmetic expression
```

All of the matrix' numerical values will be erased from the screen and the cursor will be placed at the first position of the first cell, waiting for the user to enter an arithmetic expression. If the F1 key is pressed anytime during the entry of an arithmetic expression, MATRANS will cancel the command and request a new one.

This function calculates the same arithmetic expression for each element of the row or column, i.e. the same arithmetic expression is used with the numerical values corresponding to each cell or position. In other words, given the vectors $\mathbf{b} = [b_i]$, $\mathbf{c} = [c_i]$ and $\mathbf{d} = [d_i]$, and the arithmetic expression

$$\mathbf{b} = \mathbf{c} + 0.5 \times \mathbf{d}$$

the F6 function will calculate the vector

$$\mathbf{b} = [c_i + 0.5 \times d_i]$$

An arithmetic expression or function can be defined by a text of up to 128 characters. This text must follow the format:

<name>=<formula>

where <name> is the name of the vector (column or row) to be calculated and <formula> defines how it is to be calculated; the "=" sign is essential to the execution of this function. A formula consists of constants, names, arithmetic operations and parenthesis. Exponential notation is not accepted with constants, i.e. it is NOT possible to enter 1.035E-1 since the letter "E" is considered a name. Table 5.3 gives some examples of arithmetic expressions that can be used with F6 Calc. If there is some error with the arithmetic expression or in the calculation process, MATRANS will give an error message and the calculation will not be completed.

F7 Norm

This function is used to calculate the coefficient matrix (type C) that CALCULAG will

Table 5.3. Examples of F6 Calc arithmetic expressions

Expression	Meaning
Col'A=1.0	Gives the value 1.0 to all the elements of the column Col'A
Taxes=Imports*0.15	Calculates each element of the "Taxes" row as 15% of each corresponding element of the "Imports" row
Taxes=0.20*(Petroleum+ machinery and equipment)	Each element of the row "Taxes" will be calculated as 20% of the sum of the corresponding elements of the rows between parentheses.

use later to calculate type T and type G (total requirements) matrices.

If changes have been made to the matrix in memory, it is convenient to save it by using the F4 Save function before normalizing. Then, the F7 Norm function deletes the matrix in memory and replaces it with the normalized one. Therefore, by pressing the F7 key, the user may choose whether to normalize the matrix in memory or to save it first.

The normalization is done by columns regardless of the mode MATRANS is in at that moment. The program divides the value in each cell by the sum of that column, and it does it for each column. Upon completing the normalization, a type C matrix will remain in memory with the same name as the matrix originally loaded. Therefore, by using the functions "F4 Save", "F4 Copy" and "F3 Lee", MATRANS will assume that it should save, copy, or read a type C matrix, respectively. Also, the function F7 Norm automatically expresses the numbers with 5 decimal places.

F8 Total

This function verifies the matrix numerical data. To do this, all of the columns are summed and the total is shown on the penultimate line of the screen. These totals can be used to determine whether the numerical information has been entered correctly. It should be noted that when using this function with normalized matrices (type C), the column total must equal 1.0 or there is an error.

F9 Dec

This function allows the user to specify the number of decimal places for the numerical data shown on the screen. MATRANS uses this function to change the numerical format, thus simplifying what appears on the screen. By pressing the F9 key, MATRANS will ask for a number of decimal digits to be shown (0 to 5) or the letter "E" for exponential notation. The formats that MATRANS permits are:

FORMAT	EXAMPLE
Exponential	1.234E+2
Integer	123

Decimal 123.45

MATRANS always works internally with a precision of 11 significant digits and only changes what is on the screen rounding off to the number of digits desired. When the number positions occupied by a number exceeds seven, MATRANS will display the value in exponential notation.

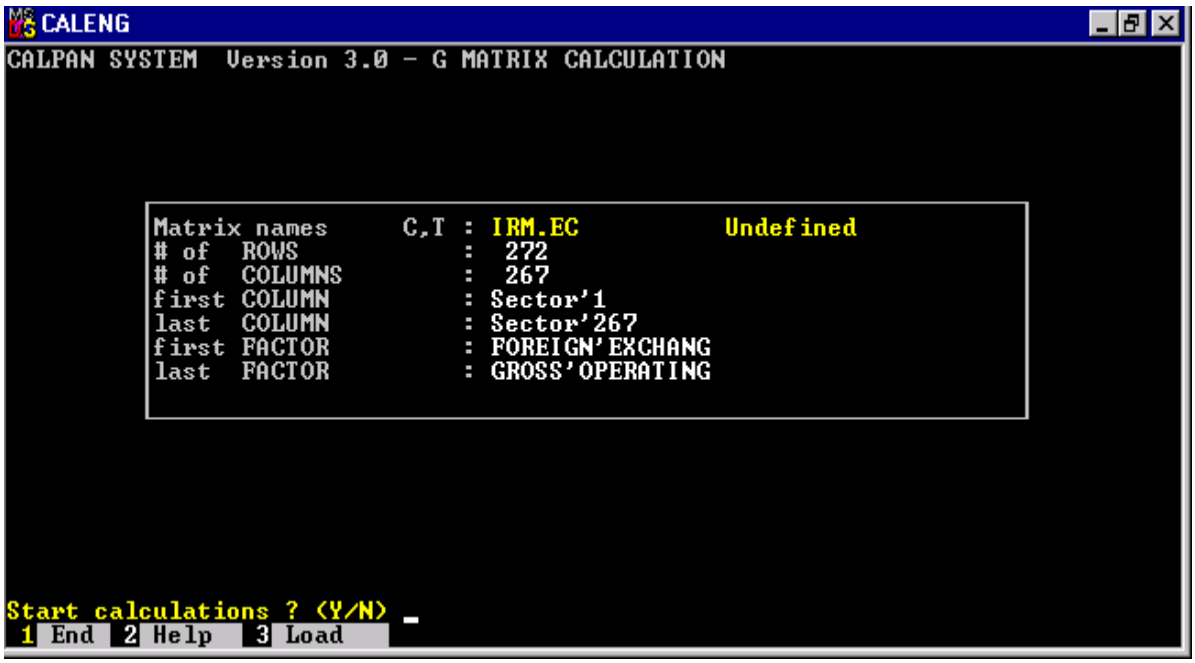
CHAPTER 6

THE CALCULATION OF MATRICES FOR TOTAL REQUIREMENTS OF NONPRODUCED INPUTS

The CALCULAG program, as its name indicates, calculates the matrix $\mathbf{G} = \mathbf{F}\,(\mathbf{I} - \mathbf{A})^{-1}$ of equation [1.8]. Upon starting this program, a window appears on the screen containing the name of the coefficients matrix, its dimensions, the names of the first and last column of matrix **A**, as well as the names of the first and last rows of matrix **F** (first nonproduced input, last nonproduced input). CALCULAG provides the following 3 function keys: "F1 Exit", "F2 Help", "F3 Load". The first two have been described (see Chapter 2) and the remaining one operates as follows.

F3 Load

This function loads to memory the type C matrix that will later be used to calculate the respective type G matrix. An intermediate step to calculate a type G matrix is to calculate matrix $(\mathbf{I} - \mathbf{A})^{-1}$, also called "Leontief inverse", identified by MATRANS as type T.



Screen 4. Program for the calculation of total requirements of nonproduced inputs and transfers CALCULAG

CALCULAG permits the calculation of a type G matrix starting from a previously existing type T matrix. Upon pressing the F3 key, the program sends the following message:

```
Use an existing type T matrix? (Y/N)_
```

If the answer is affirmative, the program will request the name of the coefficient (type C) matrix, the name of the type T matrix, and the name of the file where the type G matrix is to be saved. If the latter question is answered by simply pressing <ENTER>, the program will save the type G matrix with the same name of the type C matrix. When requesting a matrix name, CALCULAG expects the user to type it, followed by <ENTER>. If the matrix does not exist in the data directory, the program will send an error message. If the matrix exists, it will load it to memory and will show its characteristics on the screen.

If the answer is that the user does not want to use an existing type T matrix, the program will request the name of the type C matrix and the precision with which to calculate the type T matrix (from 3 to 8 significant digits, with a default value of 5). Before starting the calculations, the program will send the following message:

```
Save the (I-A)^(-1) matrix? (Y/N)_
```

If the answer is negative, the program will prompt the user to start the calculations. If the answer is affirmative, it will request the name under which you want to save the $(\mathbf{I} - \mathbf{A})^{-1}$ matrix (as type T) sending the message

```
Name of the (I-A)^(-1) matrix (type T), or ENTER -->_
```

If only <ENTER> is pressed, CALCULAG will save the matrix with the name of the type C previously loaded. If another name is given, it will save the type T matrix with such a name. When deciding to save a type T matrix the user must keep in mind that, in general, these matrices will not have any null element, and consequently will occupy considerable disk space (see Appendix C). Then the program will ask the user whether he wants to start the calculation

```
Start calculations? (Y/N)_
```

and if the answer is affirmative, it will start the calculation after sending the following message

```
Calculating matrix T = (I-A)^(-1)
```

That message will remain on the screen until the program finishes that calculation (from a few seconds to several hours, depending on the dimensions of the type C matrix).

CALCULAG will verify the existence of the inverse of (**I** – **A**). If that is not the case, the program will warn the user and will stop the calculation. In such a situation, the user should exit the program to verify the type C matrix previously loaded (the values and structure of the type C matrix in memory have already been altered). It is also possible to load another type C matrix without exiting CALCULAG.

After finishing the calculation of the type T matrix, the program will send the following message:

```
Calculating matrix G = F*(I-A)^(-1)
```

This message will remain on the screen till the calculation of the type G matrix is finished. At that time, the type G matrix just calculated will replace in memory the type C matrix originally loaded, and the program will save that type G matrix.

Before starting the calculation, and in order to extract the F matrix, the program verifies that the number of rows of the type C matrix is greater than the number of its columns. If that were not the case, it sends the following message:

```
Unable to calculate G, A is not square, press ENTER
```

Additionally, if an existing type T matrix is used, the program verifies that matrices $(\mathbf{I} - \mathbf{A})^{-1}$ and **F** are conformable for the multiplication. If that is not the case, it send the following message:

```
Matrices (I-A)^-1 and F are not conformable, press ENTER
```

CHAPTER 7
THE CONVERSION FROM USER'S TO PRODUCER'S PRICES

Given a type G matrix that corresponds to a price level (user's, for example), CALPAN can convert it to another price level, (producer's) using the PRECIOS program. To do this, the program uses two matrices: (a) an original type G matrix, and (b) a type C matrix prepared by the user. The latter must contain the transport, commerce, and other costs coefficients corresponding to the final product, calculated with respect to its user's price. This second matrix should be prepared using DEFINE and MATRANS.

The calculations completed by this program can be illustrated by an example in which a type G matrix is converted from user's to producer's prices. Thus, we define the column corresponding to any sector *j* of a type C matrix as

$$\Sigma_i a_{ij} + \Sigma_h f_{hj} = 1 \qquad [7.1]$$

where the coefficients correspond to the user's price. At the same time, a_{cj} and a_{tj} will be the direct requirements of commerce and transport. Since they correspond to user's prices, each of these coefficients will include the commerce or transport margins of the traded inputs (a^m_{cj}, a^m_{tj}) and those of the final product (a^j_{cj}, a^j_{tj}), such that

$$\begin{aligned} a_{cj} &= a^m_{cj} + a^j_{cj} \\ a_{tj} &= a^m_{tj} + a^j_{tj} \end{aligned} \qquad [7.2]$$

Finally, let

$$g^u_{hj}, \text{ such that } \Sigma_h g^u_{hj} = 1 \qquad [7.3]$$

be the total requirements of nonproduced inputs and transfer payments of sector *j* at user's prices, and let

$$g_{hc}, \text{ such that } \Sigma_h\, g_{hc} = 1$$
$$g_{ht}, \text{ such that } \Sigma_h\, g_{ht} = 1 \qquad [7.4]$$

be those of commerce and transportation, respectively. The total requirements at producer's prices g^p_{hj} will be calculated as

$$g^p_{hj} = \frac{g^u_{hj} - g_{hc}\, a^j_{cj} - g_{ht}\, a^j_{tj}}{1 - (a^j_{cj} + a^j_{tj})} \qquad [7.5]$$

As any other matrix in the CALPAN system, the matrix containing the distribution margins (commerce, transportation, and other costs), those explaining the difference between user's prices and producer's prices, can be created with the DEFINE program. Using DEFINE, the user will define a type D matrix whose columns have the same names as the type G matrix to be converted. The rows will be: (a) an auxiliary row that can be called anything accepted by the system (it is recommended that this row be called something that identifies it as the one corresponding to the producer's prices); (b) a row for each one of the distribution costs that explain the difference between the user's and the producer's prices, with the same names as they had in the original type G matrix.

Next, this matrix should be loaded in MATRANS to enter the numerical values. These can be entered independently of the scale, i.e. in percentages, total values, coefficients, etc., such that the value representing producer's prices plus the distribution costs always results in the value at user's prices. Thus, for example, the matrix containing the structures of the user's prices will always be similar to that shown in Table 7.1, in which "PP" is the producer's price, and "name 74" and "name 78" are the transport and trade margins of the final product, respectively. After saving the data matrix, the corresponding coefficients can be calculated, generating a type C matrix like the one shown in Table 7.2. You should then erase the "PP" row (using the F7 function) and save the matrix that contains the distribution costs coefficients calculated with respect to the user's price.

Table 7.1. Hypothetical example of a matrix of user's price structures

	name1	name2	...	name80
PP	100	1.00000		457
name74	1	0.02000		7
name78	10	0.17000		55

In the matrix containing the distribution margins, the columns corresponding to those services that explain the difference between producer's and user's prices ("name 74" and "name 78" in the example) should be omitted; if these columns exist, the corresponding numeric entries should all be zero. Otherwise, the PRECIOS program will not load such and will send an error message.

It is important to emphasize:

i) the need to erase the corresponding producer's price row. Otherwise, PRECIOS will interpret this row as a cost that explains part of the difference between user's and producer's prices and will try to find the corresponding column on the type G matrix; upon not finding one, it will give an error message.

ii) that no matter the direction of the conversion, (producer's <---> user's), the distribution costs coefficients should always be calculated with respect to the user's prices.

Once the matrix of the distribution costs coefficients has been saved, you can return to the main menu and start the PRECIOS program. Upon doing so, a window will appear showing the characteristics of the type G matrix to be converted as well as the name of the matrix

Table 7.2. Hypothetical example of a matrix of user's price structures, in coefficients.

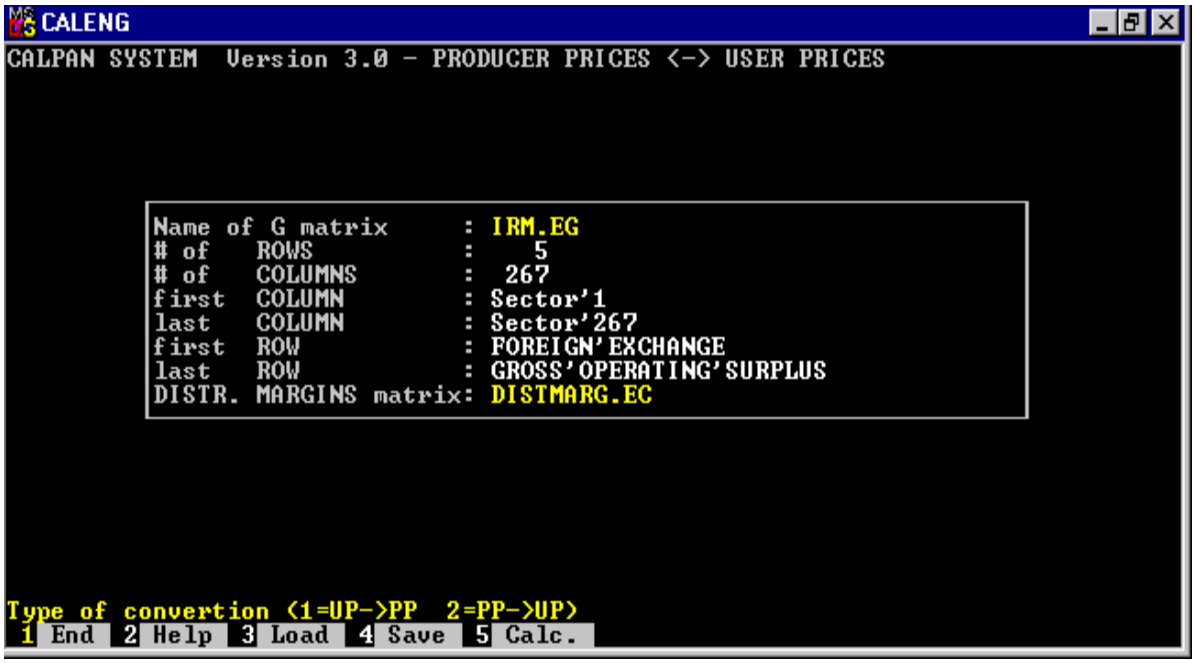
	name1	name2	...	name80
PP	0.90090	0.84034		0.88054
name74	0.00901	0.01681		0.01349
name78	0.09009	0.14286		0.10597

Source: Calculated from Table 7.1.

containing the distribution costs coefficients corresponding to user's prices. The program has five function keys: the first two functions work exactly as in all of the programs previously described; the others are described in this chapter.

F3 Load

Using this function, you can load both the type G matrix to be converted as well as the type C matrix containing the distribution costs coefficients. By pressing the F3 key, the program will ask for the name of the type G matrix to be converted and then the name of the distribution costs coefficients matrix (type C). If you try to load a type C matrix that does not exist in the data directory, PRECIOS will give an error message and will again ask for the matrix name. If the names of the rows of the C matrix differ from those of the columns of the type G matrix, the program will give an error message for each one and assume that no matrix has been loaded. If, in the type C matrix of distribution costs coefficients, the program finds nonzero elements in the columns whose names are the same as one or more rows (for example, transportation and commerce) the program will give an error message indicating that these columns should contain only zeroes and will not load the type C matrix.



Screen 5. Program for converting type G matrices from user's to producer's prices and vice versa (PRECIOS)

F4 Save

This function is used to save the matrix calculated with the F5 function (see next paragraph). The matrix to be saved is a type G matrix corresponding to user's or producer's prices, depending on the option chosen. As a precaution, it is not possible to save the new matrix with the same name as the one originally loaded. Thus, it is necessary to specify a different name for the newly calculated matrix.

F5 Calc

By pressing the F5 key, the previously described calculating process is started. This function calculates the type G matrix that will replace the matrix originally loaded to memory. Before starting the calculation, the program will ask the direction of the conversion. You should answer with option 1 if you wish to convert from user's to producer's prices and option 2 if you wish to convert producer's prices to user's prices.

WARNING: Since the work space is limited, the use of the PRECIOS program is subject to the following constraints:

$$2 \times NRG + NRC < RMAX + 1$$

where *NRG* is the number of rows in the original type G matrix, *NRC* is the number of rows in the type C matrix containing the distribution margins and *RMAX* = 300. If you exceed these limits, the program will give an error message and will not load the matrices.

CHAPTER 8
UPDATING FOR RELATIVE PRICE CHANGES

Frequently, the existing matrix (or the data to prepare it) corresponds to relative prices of produced and nonproduced goods that have since changed. In this case, it will be preferable to have a matrix that corresponds to the new relative prices. CALPAN allows the calculation of a new type G matrix corresponding to the new relative prices under the assumption that the prices of the produced goods are determined by those of the nonproduced inputs and transfer payments according to the input-output price model. In other words, the program assumes fixed, physical coefficients and forward transfer of the price increases of nonproduced inputs and transfer payments. The program that carries out these calculations is called AJUSTEG.

AJUSTEG distinguishes between two types of coefficients: exogenous and endogenous. The first are the ratios of the new to the old prices of the nonproduced inputs (example, wage or exchange rate indexes) which are considered exogenous to the price determination system. In the case of those nonproduced inputs or transfer payments (elements of **F**) that have not been assigned an exogenous coefficient, the program assumes that they are a constant proportion of prices (example, sales tax and gross operating surplus). The second, or endogenous coefficients, are the price indexes resulting from the transfer of cost increases to prices, and therefore are the ones that correspond to those elements of the **F** matrix whose value coefficients are a constant proportion of the gross value of production; indirect taxes, for example. In other words, AJUSTEG uses a price model of the type

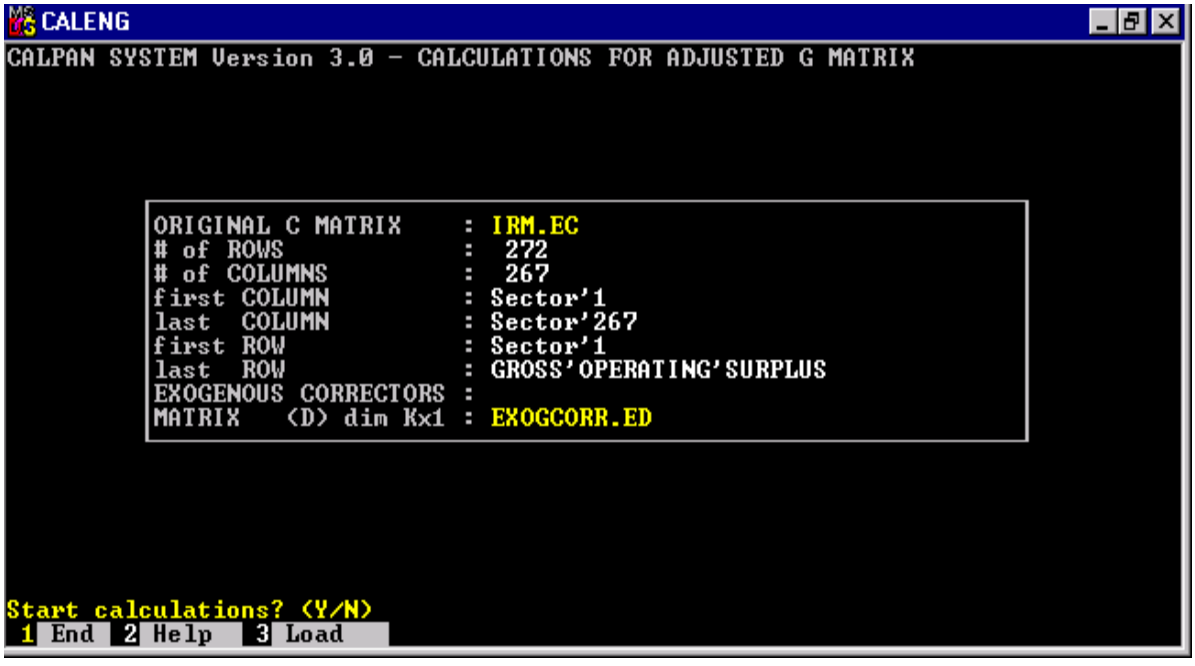
$$\mathbf{z} = \mathbf{z}_h \mathbf{V} (\mathbf{I} - \mathbf{A} - \mathbf{T}_1 - \ldots - \mathbf{T}_n)^{-1} \qquad [8.1]$$

in which **z** is the vector of the price indices of the produced goods, $\mathbf{z}_h$ is the vector of the price indices of those nonproduced inputs or transfer payments corrected exogenously, **A** is the transaction matrix and the $\mathbf{T}_i$ are diagonal matrices whose diagonal elements are those of the vectors (rows) of the **F** matrix corresponding to items that are a fixed proportion of output price and, therefore, are corrected endogenously. AJUSTEG calculates the **z** vector and then uses it in conjunction with the $\mathbf{z}_h$ to calculate a new type G matrix corresponding to the old physical

coefficients and the new prices. Appendix D provides a more detailed explanation of the procedures and formulas, and Londero (1992) provides examples following a similar, although not identical approach.

To complete these calculations, AJUSTEG requires the type C matrix corresponding to the old prices (or matrices **A** and **F**) and the vector of exogenous coefficients $\mathbf{z}_h$. The latter is created with DEFINE as a type D matrix with one column and as many rows as rows of the **F** matrix that will be exogenously corrected. The column may have any valid name, but AJUSTEG wants the names of the rows to be identical to the nonproduced inputs and transfer payments of the type C matrix and *assumes that those omitted should be corrected endogenously*, i.e., those omitted correspond to the diagonal matrices $\mathbf{T}_i$ of equation [8.1]. Then, the vector created in DEFINE has to be loaded into MATRANS, the values of the ratios of the new to the old prices (or z_h price indices) have to be entered, and the vector has to be saved for later use by AJUSTEG.

Upon starting AJUSTEG, the system displays a window showing the name of the type C matrix to be used in the calculation, its characteristics and the name of the matrix (vector) of exogenous coefficients. It also displays the function key described in the following pages.



Screen 6. Program for adjusting a type G matrix to changes in the relative prices of the nonproduced inputs (AJUSTEG)

F3 Load

The "F3 Load" function key permits the user to load the type C matrix and the vector of exogenous coefficients to the work space. Upon pressing the F3 key, the program will ask for the name of the original type C matrix from which it will calculate the adjusted type G matrix. After the user has entered this name and AJUSTEG has loaded this matrix to memory, the program will ask for the name of the type D matrix of EXOGENOUS coefficients (vector $\mathbf{z}_h$ of equation [8.1]).

An intermediate step to calculate the adjusted type G matrix is to calculate matrix $(\mathbf{I} - \mathbf{A} - \mathbf{T}_1 - \mathbf{T}_2 ... - T_n)^{-1}$. Before starting the calculations, the program will send the following message:

```
Save the (I-A-T1-T2-...-Tn)^(-1) matrix? (Y/N)
```

If you answer yes, the program will ask for the name under which it will save that matrix. Then the program will ask for the name under which you want to save the adjusted type G matrix. If these questions are answered by simply pressing <ENTER>, the program will save the type G matrix and the type T matrix under the same name of the type C matrix. The program will continue by asking if you wish to save the vector of endogenous coefficients. If you answer yes, it will ask the name of the file where the vector is to be saved, as a type S matrix. Then the program will ask the user whether he wants to start the calculation by sending the message

```
Start the calculations? (Y/N)
```

If the answer is affirmative, it will start the calculation and send messages indicating the different calculations that are being made. Once the program completes the calculation of the adjusted type G matrix, it will save it under the name previously provided.

CHAPTER 9
THE CALCULATION OF TOTAL REQUIREMENTS OF NONPRODUCED INPUTS FOR SECTORS NOT INCLUDED IN THE ORIGINAL MATRIX

CALPAN includes a program called BIENES to calculate the total requirements of nonproduced inputs and transfer payments corresponding to sectors not included in the original IRM. For that purpose, each one of the inputs (produced and nonproduced) and transfer payments that make up the cost structure of the sector or product in question should be assigned to a row on the original type C matrix so that BIENES knows how to decompose the cost structure backwards, using the respective type G matrix. BIENES follows the procedure described below.

Given the costs structure of sector *k* not included in the matrix

$$\Sigma_i \, a_{ik} + \Sigma_h \, f_{hk} = 1 \qquad [9.1]$$

in which each produced input *i* or nonproduced input *h* corresponds to one in the original matrix, BIENES will calculate the total requirements of nonproduced inputs g_{hk} as

$$g_{hk} = f_{hk} + \Sigma_i \, g_{hi} \, a_{ik} \qquad [9.2]$$

in which g_{hi} are the total requirements of nonproduced inputs from product *i*, and will present them as a vector (in the case of one product) or as a type G matrix (in the case of various products).

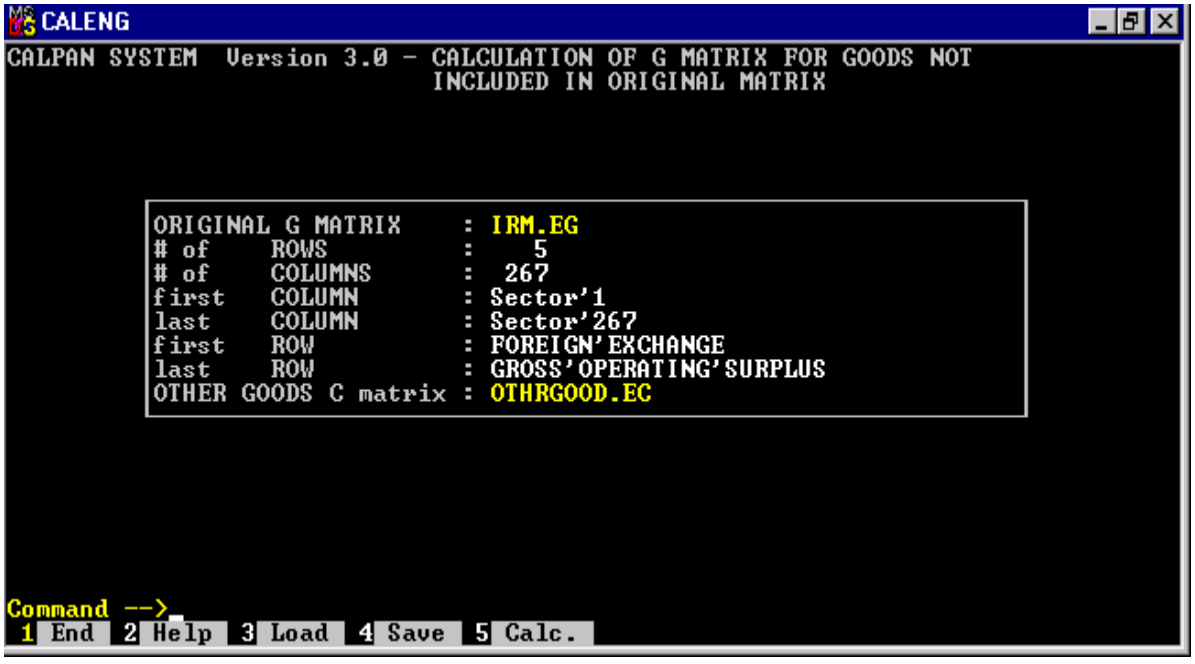
$$\mathbf{G}^k = [g_{hk}] = [f_{hk} + \Sigma_i \, g_{hi} \, a_{ik}] \qquad [9.3]$$

Therefore, it is necessary that the user, using DEFINE and MATRANS, creates and constructs the vector (matrix) that contains the cost(s) structure(s) that is (are) to be decomposed. This matrix can be created as type D, but should be converted to coefficients (type C) so it may be used by the BIENES program.

Since BIENES generates a type G matrix, the resulting matrix may be used by the program that calculates accounting price ratios (RPC) or domestic resource cost of foreign exchange (CALCID). In this manner, the analyst may, for example, calculate *apr* for products not originally included in the matrix.

While preparing the vector or matrix, you should follow some simple and obvious rules. First, the inputs from sector *k* should be ordered such that all of the produced inputs are followed by the nonproduced inputs; it is recommended that they be ordered as in the original matrix. However, it is not necessary to specify those inputs whose coefficients are zero. Since BIENES decomposes the vector (matrix) looking for the vectors [g_{hi}] that correspond to each coefficient a_{ik}, it is necessary that the names of the rows are identical to those of the columns and rows of the type G matrix.

Once the type C matrix containing the cost structures has been created and saved, you can return to the main menu and call the BIENES program. The program will display a window to show the name and characteristics of the type G matrix to be used for the decomposition and the



Screen 7. Program for calculating the total requirements of nonproduced inputs for sectors not included in the original matrix (BIENES)

name of the type C matrix in question. Also displayed are the function keys described in this chapter

F3 Load

The matrices should be loaded to memory using the F3 Load function. Upon pressing this key, the BIENES program will ask for the name of the original type G matrix. Once this matrix is loaded to memory, the program will ask for the name of the type C matrix of goods not included in the original matrix, or OTHER GOODS matrix.

F4 Save

Once the type G matrix for other goods has been calculated, it can be written to disk using the F4 function key. By pressing it, the program will ask for the name of the file where the matrix is to be copied and will proceed on to save the matrix.

F5 Calc

Once the matrices are loaded to memory, the process of calculating the type G matrix for other goods not included in the original matrix is started by pressing the F5 function key. Upon finishing the calculation, the type G matrix for the other goods just calculated replaces the original type G and C matrices in memory.

WARNING: Since the amount of work space is limited, the BIENES program is subject to the constraint

$$2 \times NRG + NRC < RMAX + 1$$

in which *NRG* is the number of rows of the original type G matrix, *NRC* is the number of rows of the type C "other goods" matrix and *RMAX* = 300. If you exceed these limits, the program will give an error message and will not load the matrices requested.

CHAPTER 10
THE SPECIFICATION OF apr_h^f FUNCTIONS

CALPAN includes a program called FUNCION that permits the user to specify exogenous apr_h^f to the system as well as the functional relations that may exist between the apr_h^f and the accounting price ratios of production sectors (apr_j). These are the apr_h^f functions that we identified in equation [1.10], i.e.

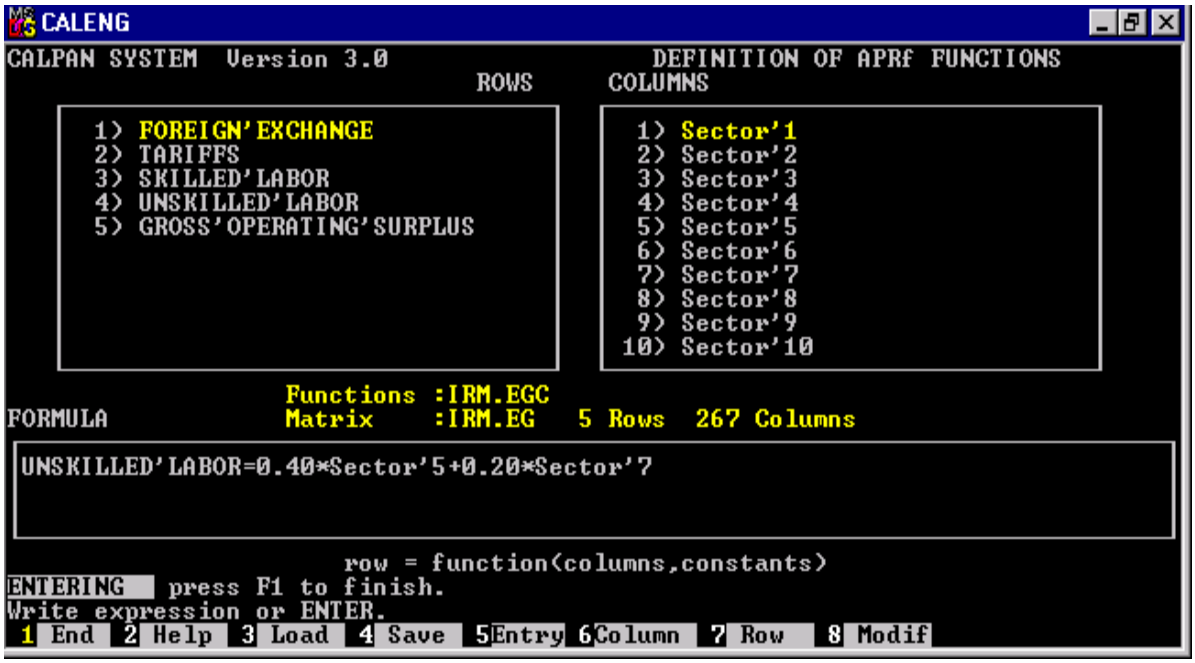
$$apr_h^f = apr_h^f(\mathbf{apr})$$

Each apr_h^f function corresponds to a row of the type G matrix and each apr_j corresponds to a column of the same. Therefore, for FUNCION to adequately register the functions (apr_h^f), it should have a type G matrix in memory, as well as a file where the definition of the $apr_h^f(\mathbf{apr})$ functions can be copied.

This program includes eight function keys: "F1 Exit", "F2 Help", "F3 Load", "F4 Save", "F5 Enter", "F6 Column", "F7 Row" and "F8 Modif". The F1 and F2 function keys operate as explained in Chapter 2. However, the functions "F3 Load" and "F4 Save" are somewhat different than previously described.

F3 Load

This function serves to load to memory the description of the rows and columns of a type G matrix as well as the definition of the apr_h^f functions of equation [1.10]. It operates in the manner described in Chapter 2 of this manual, except that the values of the matrix are not loaded. Once the type G matrix is loaded, FUNCION will ask for the name of the file that will contain the definition of the apr_h^f functions; this file will be given the extension '.EGC'. If upon this request the user presses <ENTER>, the program will assume that the functions file has the same name as the type G matrix. If a file of this name does not exist, or if the dimensions of the matrix do not coincide with those in the functions file, the program will give a warning message indicating that the user should enter the definition of all the apr_h^f functions.



Screen 8. Program for the specification of apr_h^f functions (FUNCION)

Therefore, the user will receive this message the first time he/she tries to define these relations for a specific type G matrix. If the functions file already exists, and the dimensions coincide, the program will load it to memory and the user will be able to modify or enter the definitions in the manner described later in this chapter. Upon completing the loading process, two windows are displayed on the screen with the names of the columns and rows of the type G matrix.

F4 Save

By pressing the F4 key, FUNCION will save the definitions of the apr_h^f functions but *not* the type G matrix, since it is only used as a reference. The program will ask for the name of the functions file where these definitions are to be saved. If you press only the ENTER key, the program will assume that this file will have the same name as the type G reference matrix with an extension '.EGC.' If the functions file already exists on disk, FUNCION will tell you so and ask whether to replace it with the new one or cancel the copying operation.

Along with defining the functions, FUNCION also records in the file the dimensions of the type G matrix, as well as an internal code that represents the manner of calculating the values

of the function from the values of the apr_j for the sectors that appear in the arithmetic expression, or from numerical constants. The RPC program will read this internal code later in order to calculate the **apr** vector.

F5 Entry

When the apr_h^f functions file to be used does not exist, the corresponding definitions must be entered. By pressing the F5 key, the name of the first row will appear in the functions window in the following manner:

<name of row>= 1.0

The cursor remains immediately after the equal sign (below the default value 1) waiting for the user to write the numerical value of the exogenous apr_h^f or, the arithmetic expression or function corresponding to the row whose name is <name of row>. These functions are arithmetic expressions like the ones defined in MATRANS (see Chapter 5), except that the user only needs to specify the right side of the expression; FUNCION always provides the left side. To enter the numerical value of the apr_h^f function, you must type the number or the arithmetic expression followed by <ENTER>.

When the numerical values of the apr_h^f functions are entered, it is possible to leave some rows undefined by pressing only <ENTER>. The user should be sure that this is desired since the RPC calculation program initializes the vector with "seed" values equal to 1.0. If the user does not modify them by entering a number or a function, the values of these apr_h^f will be 1.0.

Upon entering an element, FUNCION always gives the name of the next row until all of the elements have been entered. If you wish to finish entering elements before arriving at the last row of the type G matrix, press F1, in which case the program will assign a value of one to the rows for which no arithmetic expression has been entered.

Upon writing functions, the user should verify that the spelling structure of each one of the names of the columns is exactly the same as that appearing in the COLUMNS window. FUNCION will give an error message for any error found in the arithmetic expression and will permit the user to try again to enter the appropriate expression. Here are some examples of apr_h^f

functions:

Foreign Exchange=1

Labor=0.50*ACF+0.20*CCF

Taxes=0

Fixed Supply=SCF

In the example, it is assumed that "ACF", "CCF", and "SCF" are column names appearing in the COLUMNS window. The cases of "Foreign Exchange" and "Taxes" correspond to what we have referred as exogenous apr_h^f.

F6 Column

The F6 key is used in order to see the names of the columns when writing the definition of a function (using F5 Enter). By pressing it, the cursor will be located at the name of the first column in the right window of the screen. To move the cursor and locate the window in an appropriate spot on the matrix, use the vertical arrows. When the upper or lower limits of the window have been reached, FUNCION will scroll the names along the window if necessary. To return to the previous state, press the F1 key. The F6 key may be used when FUNCION asks for a command, or when the user is entering an arithmetic expression (see "F5 Enter" or "F8 Modif"). In this case, the cursor will return to the position where it was before the F6 key was pressed.

F7 Rows

The F7 key is similar to the F6 key, except that it operates over the rows of the type G matrix and uses the window on the left of the screen. It cannot be used when entering an arithmetic expression, since upon entering or modifying definitions, FUNCION automatically assigns the name of the corresponding row.

F8 Modif

The F8 key is used to edit (modify) the definition of a value or an apr_h^f. function corresponding to a specific row. By pressing it, FUNCION will ask for the name of the row to be modified. If the user specifies a row that does not exist, FUNCION will give an error message

and ask for another command. If the row exists, it will be displayed in the functions window along with its definition. At this time, you may modify the arithmetic expression by completely rewriting it. If F1 is pressed at any moment during the modification, it will not be completed and FUNCION will ask for a new command. In addition, if the user presses only <ENTER> before pressing any other character, no modification will be made.

If you wish to modify a large number of definitions without specifying the row name for each one, use the "F5 Enter" key. By pressing F5, the definition of the first row will appear. If this is the row to be modified, type the new expression followed by <ENTER>. If this row is not to be modified, press only <ENTER> and the original expression will remain intact. Next, FUNCION will present the following row along with its definition and you may continue in the same manner until all rows have been completed. If F1 is pressed at any moment during this process, FUNCION will ask for a new command.

WARNING: If after having defined the corresponding apr_h^f functions (or exogenous values) for a specific type G matrix, the user modifies and saves this matrix using MATRANS or DEFINE (changing names of rows or columns, or erasing, moving or combining them), these programs will give a warning message to remind the user to modify the function(s) file(s) corresponding to that matrix. The RPC program only verifies that the dimensions of the type G matrix and the ones recorded in the functions file are the same. *If the user has modified the structure of the type G matrix in the manner indicated, without modifying the corresponding functions file, and even if the the mentioned dimensions are still the same, the numerical values of the resulting apr_j will be wrong*.

CHAPTER 11
THE CALCULATION OF ACCOUNTING PRICE RATIOS

The RPC program simultaneously calculates the accounting price ratios of the production sector as well as those of the nonproduced inputs and transfer payments using the iterative process described at the end of Chapter 1. Therefore, RPC requires a type G matrix and the file corresponding to the **apr**^{f} vector, created by the FUNCION program, containing the exogenous or the apr_h^f functions. This file also contains an internal code from the RPC program. The program cannot start the calculation if this code does not exist, and the only means of creating it is by using the FUNCION program.

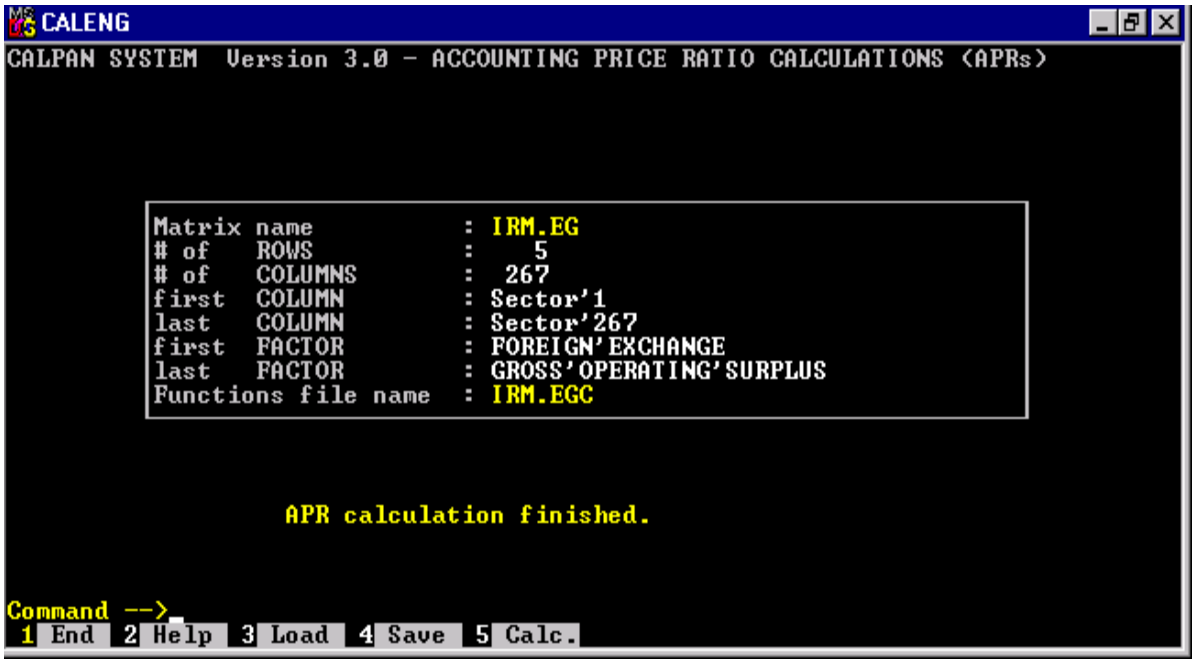
Upon starting it, RPC displays a window on the screen to show the characteristics of the type G matrix that will be loaded, i.e. its dimensions, the names of the first and last columns as well as the first and last rows. RPC includes 5 function keys. "F1 Exit" and "F2 Help" were explained in Chapter 2.

F3 Load

When the F3 key is pressed, the RPC program will ask for a name of a type G matrix with the following message:

```
Name of Matrix (Type G) -->_
```

If you specify a matrix name, RPC will verify that it exists on the disk and current directory. If the matrix does not exist, RPC will give an error message and ask for a new matrix name. If it exists, it will be loaded to memory and its name will be shown on the screen. Next, the program will ask for the name of the apr_h^f functions file; if at this point you press only ENTER, the program will assume that the name is the same as the type G matrix with the extension ".EGC". If the apr_h^f functions file does not exist or the dimensions do not coincide, RPC will give an error message and will assume that no matrix has been loaded. After loading the functions file, the following items will appear in the window: the name of the matrix, its number of columns and rows, and the first and last names of its columns and rows.



Screen 9. Program to calculate the **apr** (RPC)

F4 Save

The F4 key permits you to save on disk the **apr** and **apr**^{f} vectors calculated by using the F5 function (see below). Therefore, the F4 function should be used after F5 Calc. The program copies the **apr** and **apr**^{f} in different files. The first is saved in a type S file (for Sectors) and the **apr**^{f} vector is saved in a type I file (for Inputs). These vectors, although they are row vectors (see Chapter 1), are saved as matrices of dimensions (n,1) (n rows, 1 column) and (k,1) (k rows, 1 column) respectively, so that the printing program can read them. The name of the only column is "Values of the APR", and the numerical entries should be interpreted as the values of the accounting price ratio from the sector or the nonproduced input whose name appears in the corresponding position.

Upon pressing the F4 key, the program will give the message

```
Name of Matrix for INPUTS (Type I) or ENTER -->
```

giving the user the opportunity to specify a name for the file where the apr_h^f values will be stored. If only the ENTER key is pressed, the program will save the file with the same name as the type

G matrix used in the calculation. It will then give the message

```
Name of Matrix for SECTORS (Type S) or ENTER -->
```

also giving the user the opportunity to specify a file name or to use the same name as the type G matrix by pressing ENTER.

F5 Calc

This function starts the calculation of vectors **apr** and **apr**^{f}. Upon pressing the F5 key, RPC displays a message indicating that it is calculating the **apr**. Upon starting the calculation, vectors **apr** and **apr**^{f} are initialized with 1.0. In other words:

$apr_h^f = 1 \qquad h = 1, 2, \ldots k$

$apr_j = 1 \qquad j = 1, 2, \ldots n$

Next, the cycle of iterations described in Chapter 1 starts; the number of iterations calculated is shown on the screen. The user may stop the program at this point by pressing the "c" key, in which case the value of the vectors **apr** and **apr**^{f} will correspond to those of the last iteration calculated, without completing the convergence criterion.

In each iteration, the RPC program calculates the arithmetic expression defined for each apr_h^f using the available apr_j values, thus obtaining a new **apr**^{f} vector. Once that all of the apr_h^f arithmetic values have been calculated, RPC calculates the expression

$$\mathbf{apr} = \mathbf{apr}^f \, \mathbf{G}$$

and starts a new iteration. The process continues until the values of **apr** and **apr**^{f} converge on a stable value. The vectors **apr** and **apr**^{f} are considered to have stable values when for each one of its values, the absolute value of the difference between the value corresponding to the previous iteration (v_i^{n-1}) and the current (v_i^n) is less than one one hundred thousandth of v_i^n; in other words, the convergence criterion can be expressed as

$$\frac{\left| v_i^{n-1} - v_i^n \right|}{v_i^n} < 0.00001 \quad \text{for } i = 1, 2, \ldots n$$

CHAPTER 12
THE CALCULATION OF DOMESTIC RESOURCE COSTS OF FOREIGN EXCHANGE

CALPAN includes a program called CALCID that calculates the domestic resource costs of foreign exchange (*drc*) of sectors included in a type G matrix. The logic underlying the CALCID calculations requires an explanation. In order to calculate the *drc* of a sector, you need its production costs decomposed "backwards" into total requirements of nonproduced inputs and transfer payments. Therefore, CALCID requires a type G matrix containing these costs that it will call the "costs matrix." The domestic production of the sector in question substitutes imports or increases exports by a certain cif or fob value, respectively. However, it is necessary to value the additional substituted imports or the increase in exports at the same level as the respective sectors of the matrix. Thus, for example, if substituting import sector *k* is valued at user's prices in the matrix, the unit value of the substituted imports at user's prices, or savings, will be

$$p_k^m = F_{dk} + F_{sk} + A_{tk} + A_{ck} - E_k \qquad [12.1]$$

where

p_k^m = domestic price of imports, made equal to the price of domestic production (E_k absorbs the adjustment)

F_{dk} = unit foreign exchange cost

F_{sk} = tax per unit

A_{tk} = transportation cost per unit

A_{ck} = commercial cost per unit

E_k = excess protection

and can be expressed in coefficient form as

$$a_{tk} + a_{ck} + f_{dk} + f_{sk} - e_k = 1 \qquad [12.2]$$

Note that in the savings composition, if there is excess protection, $f_{sk} - e_k$ is the tax increase that equals the domestic price of the domestic production with that of imports. CALCID requires that

the savings (or earnings) be provided in the form of nonproduced inputs and transfer payments, for which it will be necessary to prepare a type G matrix (of savings). Since CALCID compares the matrices of costs and savings, if the savings structure includes the excess protection (necessary to calculate the coefficients in the type C savings matrix), it will be necessary to add an empty row to the type G matrix with the same name as the row that contains the excess protection in the "savings" matrix.

CALCID also uses the $\mathbf{apr}^f$ vector of accounting price ratios of nonproduced inputs and transfer payments, which must be prepared using the RPC program or using DEFINE and MATRANS when you do not need to pass through FUNCION. Note that if it was necessary to add a row to the type G "costs" matrix for the excess protection, it will be necessary to calculate a $\mathbf{apr}^f$ vector even when one already exists, since the file that contains the apr_h^f will not include the row corresponding to excess protection. Therefore, it will be necessary:

(a) to prepare a type C savings matrix and calculate a type G savings matrix using the BIENES program;

(b) to add an empty row for the excess protection to the costs matrix, if necessary;

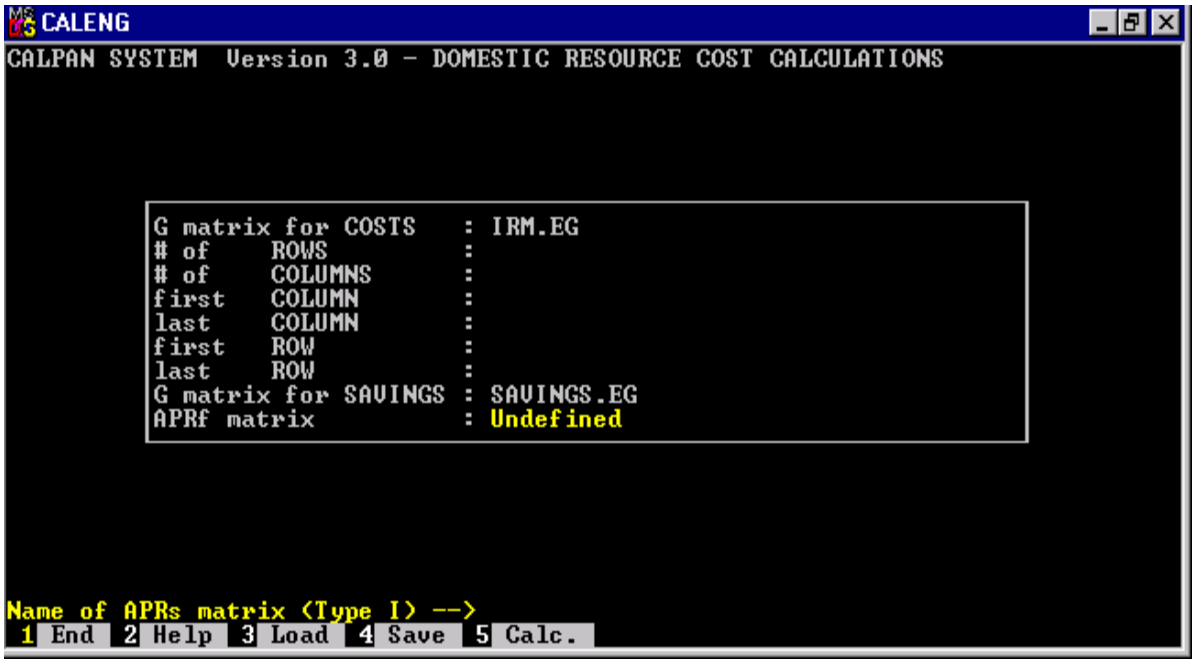
(c) if (b) was necessary, use the FUNCION program to create a new file of apr_h^f functions or create a type I file (using DEFINE and MATRANS) with the values of the apr_h^f if these are available; and,

(d) if it was necessary to create a new functions file, call and execute RPC to calculate the new $\mathbf{apr}^f$ vector

Once these steps have been completed, you can start the CALCID program from the main menu. Upon doing so, the program will display a window that will later show the name and the characteristics of the type G "costs" matrix, the name of the "savings" matrix and the name of the matrix containing the accounting price ratios of the nonproduced inputs.

F3 Load

To load the matrices for this calculation, use the F3 load function. Upon pressing the F3 key, the program will ask for the name of the "costs" matrix (type G). Once it has been loaded to memory, the program will ask for the name of the "savings" matrix (also type G) and, once



Screen 10. Program for the calculation of domestic resource costs of foreign exchange (CALCID)

loaded, will ask for the name of the **apr**^{f} matrix (vector). If the program discovers during the loading process that one of these matrices does not exist on disk, the program will again ask for the name of the matrix in question until the user provides the name of an existing one or presses the F1 Exit function key. In the latter case, the program will return to "Command--> and will operate as if no matrix has been loaded. Once these matrices have been loaded, CALCID will ask for the names of the rows on the type G matrix that represent foreign exchange. The program allows various types of foreign exchange with each one represented by a row on the type G matrix, even though only one row would normally be used for foreign exchange. The user must specify at least one row on the type G matrix that represents foreign exchange.

F4 Save

Once the matrix of domestic resource costs of foreign exchange has been calculated, it can be saved using the F4 key. Upon pressing it, the program will ask for the name of the file where the matrix containing the drc_j is to be saved. If you press only ENTER, the program will

save this matrix with the same name as the "savings" matrix, but as a type S since it contains the sectoral *drc* and therefore should not be interpreted as a vector of sectoral accounting price ratios as in the RPC program.

This matrix consists of $2n + 1$ columns, where n is the number of columns on the "savings' matrix. The first column contains the $\mathbf{apr}^f$ vector and the remaining columns contain the values of SAVINGS and COSTS for each good included in the savings matrix. The last row contains the numerical value from the *drc* corresponding to each one of the sectors on the savings matrix.

F5 Calc

Upon pressing the F5 key, the program will start the process of calculating the *drc*, so long as the necessary matrices have been loaded using the F3 function.

WARNING: Since the work space is limited, the use of the CALCID program is subject to the following constraint:

$$NRMG + NRMA < RMAX$$

where *NRMG* is the number of rows on the type G "costs" matrix, *NRMA* is the number of rows on the type G "savings" matrix and $RMAX = 300$. If you exceed these limits, the program will give an error message and will not load the matrices requested.

CHAPTER 13
ARITHMETIC OPERATIONS WITH MATRICES

The ARIMAT program provides a simple language to perform arithmetical operations with matrices, to extract partitions and to reconstruct a matrix from submatrices or partitions. Its use only requires the knowledge of a limited number of operators listed in Section 13.2.

When started, the program builds in memory a directory of the matrices in the data directory, i.e, the one designated as such by the Change_Dir function of the ARCHIVOS program, (see Chapter 8). The program will then use this directory as a reference to recognize the type and name of the matrices used in the arithmetic expressions. A window shows the type, names and dimensions of the matrices present in the data directory.

The functions available in ARIMAT (besides F1 End and F2 Help) are the following:

F3 Calc

This key permits the user to type an arithmetic matrix expression, which will later be evaluated by the program, building a new matrix or replacing an existing one.

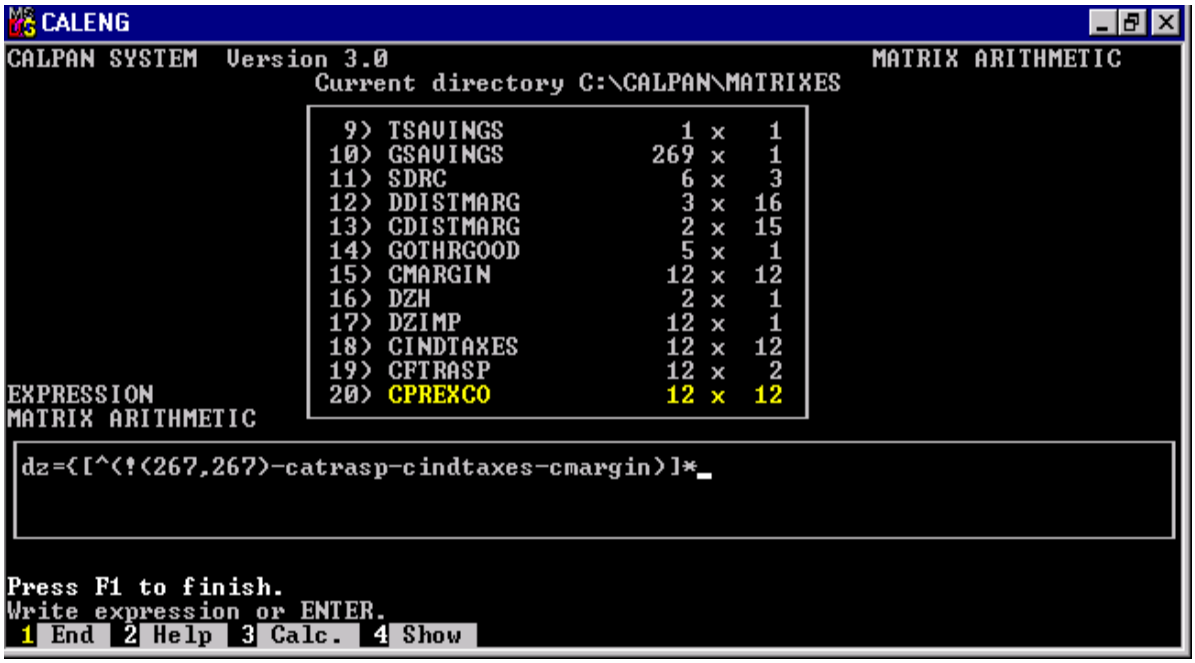
F4 Show

This function moves the cursor to a window showing the names of the matrices in the data directory, and permits the use of the vertical arrows to scroll the window on the directory.

As before, to select one of these functions, it is only necessary to press the corresponding function key. Then, the program will ask the user the information needed to execute the desired function.

13.1 Syntax

With the purpose of using the names of the matrices as "variables", the program constructs the name of a matrix beginning with the type and name of the file in which it is stored: for example, if a type G matrix called MAT1 exists in the disk, the program will construct the name GMAT1 for this matrix. In general, a matrix variable has a name with the format



Screen 11. Program to perform arithmetic operations with matrices (ARIMAT)

<type><matrix> where <type> is the type of matrix and <matrix> is the name of the file where it is stored.

An arithmetic expression has the following general format:

<variable>=<expression>

where <variable> is the name of a (new or existing) matrix variable, and <expression> is a combination of constants (or scalars), operators, parentheses, and matrix variables. These constants should be introduced in integer or floating point formats; *the use of the exponential format is not permitted*.

Some examples of arithmetic matrix expressions are:

DMAT2=5.25*DMAT1	multiplies the DMAT1 matrix by the scalar 5.25
GMAT=FMAT*TMAT	multiplies the FMAT matrix by the TMAT matrix
DM=DA+DB	Adds DA and DB matrices

The operators are classified into unary (applies to only one operand), binary (applies to two) and mixed (applies to one or two operands, depending on its location in the expression).

The only mixed operand which the program uses is the "–" sign, which means subtraction when it is used as a binary operator and "change sign" when it is used with only one operand. Unary operators must have their corresponding operand *to the right* of the operation sign, and two operands must not be written one after the other, parentheses should be used to separate them.

One <expression> can be defined recursively in the following manner, where the sign ::= means "comprised of" or "consisting of":

1) <expression> ::= <variable>
2) <expression> ::= <constant>
3) <expression> ::= (<expression>)
4) <expression> ::= <unary_operator><expression>
5) <expression> ::= <expression><binary_operator><expression>

The preceding notation means that: i) an <expression> can be only a <variable>, e.g. CA=CMAT, or a <constant>; ii) an expression can be enclosed between parentheses (), brackets [] or braces {} (rule 3) to construct operands, as in CA=(CMAT+DM)*(DA-TG); iii) unary or binary operators can be applied to the results of a simpler expression (rules 4 and 5) as in CA=-[(CMAT+DM)*(DA-TG)], where the "-" operator changes the sign of the result of the expression to its right, and the multiplication operator "*" multiplies the matrices resulting from the addition and the subtraction indicated between parentheses. *The final result of evaluating an expression should be a valid matrix*, since it will be assigned to a matrix variable.

It is indifferent to use parentheses, brackets or braces, but the closing sign should be the same as the opening one. For example, DA=DB+(~DC) is a valid expression, but DA=DB+[~DC) will give an error message and the operation will be canceled.

13.2 Operators

The following pages contain a description of each one of the operators that can be used in an arithmetic matrix expression.

Transpose ~ This operator creates a matrix by transposing the operand to its right. In other words, if the operand was evaluated as a matrix with dimensions NxM, the result

of the transposing is a matrix with dimensions MxN. Names of rows and columns will be interchanged. For example, given matrix

```
DB =  ⎡ 4  8  2  3 ⎤
      ⎢ 6  5  5  5 ⎥
      ⎣ 2  5  7  5 ⎦
```

the operation DBTRAS=~DB will provide the following result:

```
DBTRAS =  ⎡ 4  6  2 ⎤
          ⎢ 8  5  5 ⎥
          ⎢ 2  5  7 ⎥
          ⎣ 3  5  5 ⎦
```

Expansion / The / operator expands a row vector (1xN matrix) or a column vector (Nx1 matrix), which should be the result of evaluating the operand to the right, to a diagonal matrix with dimensions NxN. The components of the vector are placed over the diagonal of the resulting matrix and the rest of the matrix contains zeroes. The names of the rows will be the same as the names of the columns. For example, given the following vector

```
DVECTOR =  ⎡ 4 ⎤
           ⎢ 5 ⎥
           ⎣ 7 ⎦
```

the expression DDIAG=/DVECTOR will result in the following type D matrix called DIAG:

```
DIAG =  ⎡ 4  0  0 ⎤
        ⎢ 0  5  0 ⎥
        ⎣ 0  0  7 ⎦
```

Diagonal \ The \ operator extracts the diagonal of a square matrix resulting from the evaluation of the operand to its right. The result of the operation is a row vector.

This vector has "diagonal" as the name of the row and the names of the columns are copied from its operand. This operation is the inverse from the previous one, in other words DVECTOR=~(\DDIAG).

Vert Sum < The < operator (vertically) sums the elements of each column of the matrix that results from evaluating the operand to its right. For example, given matrix

```
      ┌          ┐
      │  4  8  2  3 │
DB =  │  6  5  5  5 │
      │  2  5  7  5 │
      └          ┘
```

the operation DSUMCOL=<DB will result in the following type D matrix called SUMCOL:

```
DSUMCOL=[12 18 14 13]
```

This is equivalent to premultiplying the matrix by a row vector of ones; in other words, this operator performs the operation [1i]*[Aij], where matrix A has dimensions NxM and the vector of ones has dimensions 1xN.

Hor Sum > This operator (horizontally) sums the elements of the matrix that results from evaluating the operand to its right. For example, in the following matrix

```
      ┌          ┐
      │  4  8  2  3 │
DB =  │  6  5  5  5 │
      │  2  5  7  5 │
      └          ┘
```

the operation DSUMFIL=>DB will provide the following result:

```
          ┌    ┐
          │ 17 │
DSUMFIL = │ 21 │
          │ 19 │
          └    ┘
```

This is equivalent to postmultiplying the matrix by a column vector of ones; i.e. performs the operation [Aij]*[1j], where A is a matrix with dimensions NxM and the vector of ones has dimensions Mx1.

Columns " The " (quotes) operator calculates the number of columns in the matrix that results from evaluating the operand to its right. The result is a scalar that can only be used as an argument by other operators requiring an indicator of the dimensions of the matrix (operators !, # described in the following pages). For example, with reference to the preceding DB matrix, the operation DNCOL="DB will result in an error message.

Rows : The : operator (colon) calculates the number of rows in the matrix that results from evaluating the operand to its right. The result is also a scalar that can only be used as argument by other operators that require an indicator of the dimensions of the matrix (operators !, # described in the following pages).

Addition + The binary operator + (sum) performs the algebraic sum of its two operands. These can be two scalars or two matrices, but scalars must not be mixed with matrices in the operation. If operands are matrices, they must be conformable for the operation, in other words they must have the same dimensions. If this were not the case, the program would send an error message and would stop evaluating the expression. For example, given matrices DA = $[a_{ij}]$ and DB = $[b_{ij}]$, the expression DD=DA+DB will result in the following matrix:

$$DD = [d_{ij}] = [a_{ij} + b_{ij}]$$

Subtraction - The binary operator - (subtraction) performs the subtraction of its two operands. These can be scalars or matrices, but they must not be mixed in the operation. If operands are matrices, they must be conformable for the operation. For example, according to the previous example, DD=DA+DB, therefore, DA=DD-DB. This

operator can also be used as a unary operator, i.e., applied to only one operand to its right. In this case, it will change the sign of its operand. For example, given matrix

$$DB = \begin{bmatrix} 4 & 8 & 2 & 3 \\ 6 & 5 & 5 & 5 \\ 2 & 5 & 7 & 5 \end{bmatrix}$$

the operation DNEGB=-DB will result in:

$$DNEGB = \begin{bmatrix} -4 & -8 & -2 & -3 \\ -6 & -5 & -5 & -5 \\ -2 & -5 & -7 & -5 \end{bmatrix}$$

Mult * The binary operator * (multiplication) performs the multiplication between two matrices, between one scalar and one matrix, or between two scalars. If the operands are matrices, they must be conformable for the operation. When multiplying matrices, the names of the rows of the resulting matrix will be those of the first operand, and the names of the columns, those of the second. The result of multiplying a scalar by a matrix will have the same row and column names as the matrix.

Inverse ^ The ^ operator calculates the inverse of the matrix that results from evaluating the operand to its right. For example, the expression RINV=^DMAT calculates the inverse of DMAT and assigns the result to the RINV matrix, while the expression TA=^(DMAT1-DMAT2), calculates the inverse of the difference between DMAT1 and DMAT2. If the matrix to be inverted is not square, the program will give an error message and will stop evaluating the expression.

Vertex , The vertex binary operator [simple comma (,)] is used to define vertices or corners of a matrix, and it is later used by other operators to define partitions of

a matrix by specifying the upper left and lower right corners of the block. A vertex is not defined with reference to a specific matrix, but it represents the coordinates for constructing blocks or submatrices. The operand to the left must be (or must result in) a scalar and it should represent the row where the vertex is located; the operand to the right should also be (or result in) a scalar representing the matrix column where the vertex is located. Generally, a vertex is defined with its operands between parentheses, e.g., (1,5), (20,100), etc. However, the result of operators " and : can also be used as operands. For example, [1,("DA)] defines a vertex in row 1, column ("DA), i.e., the number of columns in matrix DA. Similarly, [(:DA),("DA)] defines a vertex whose row number is the number of rows in the DA matrix and whose column number is the number of columns in the DA matrix.

Partition # The partition operator (#) is used to define a block or partition of a matrix, using as operands "vertices" defined with the preceding operator. The operand to the left defines the upper left vertex of the partition, while the one to the right defines the lower right vertex. For example, the expression (1,7)#(5,50) defines the partition consisting of the intersection of rows 1 through 5 and columns 7 through 50 of a larger matrix. A partition does not have any meaning by itself; it becomes meaningful when applied to a specific matrix. For that purpose, it is still necessary to introduce the "extract" operator. Notwithstanding, it is logical that if partition (i,j)#(k,m) is defined to later be extracted from a matrix, such partition should comply with the following conditions: $0 < i < k + 1$ and $0 < j < m + 1$; i.e., both the upper left and the lower right vertexes must be defined.

Extract @ The binary operator @ (extract a partition) permits the extraction of a submatrix from a larger one, using as an operand a partition defining the position of the block within the larger matrix. The first operand must be a partition, as was previously defined, while the second operand must be a matrix (or result in a

matrix after being evaluated). Thus, if DMAT is a matrix with dimensions 20x50, the expression DA=(5,1)#(9,50)@DMAT extracts the columns between the 5th and 9th rows, and it assigns it to the DA matrix, which will have dimensions 5x50. For example, given matrix

$$DB = \begin{bmatrix} 4 & 8 & 2 & 3 \\ 6 & 5 & 5 & 5 \\ 2 & 5 & 7 & 5 \end{bmatrix}$$

the operation DPART=(1,1)#(3,3)@DB will result in

$$DPART = \begin{bmatrix} 4 & 8 & 2 \\ 6 & 5 & 5 \\ 2 & 5 & 7 \end{bmatrix}$$

If the partition defines a block that is not included in the larger matrix, the program will give an error message and will not evaluate the expression.

Horizontal Concatenation |

This operator permits the construction of larger matrices using other matrices, horizontally concatenating them, one after the other. Its two operands must be (or result in) matrices with the same number of rows, or else the program will send an error message and will cancel the operation. For example, given matrices

$$DM11 = \begin{bmatrix} 4 & 8 & 3 \\ 6 & 5 & 5 \\ 2 & 5 & 5 \end{bmatrix} \qquad DM12 = \begin{bmatrix} 8 & 1 \\ 3 & 7 \\ 2 & 7 \end{bmatrix}$$

the operation DM=DM11|DM12 will result in

$$DM = \begin{bmatrix} 4 & 8 & 3 & 8 & 1 \\ 6 & 5 & 5 & 3 & 7 \\ 2 & 5 & 5 & 2 & 7 \end{bmatrix}$$

In general, if DM11 is a matrix with dimensions NxM and DM12 is a matrix with dimensions NxQ, the expression DM=DM11|DM12 will result in a matrix with dimensions Nx(M+Q) that will be assigned to the DM variable.

Vertical Concatenation _

This operator concatenates two matrices vertically i.e., one below the other. In this case, its two operands must be (or result in) matrices with the same number of columns; otherwise the program will send an error message and cancel the operation. For example, given matrices:

$$DT11 = \begin{bmatrix} 4 & 8 & 3 \\ 6 & 5 & 5 \\ 2 & 5 & 5 \end{bmatrix} \qquad DT21 = \begin{bmatrix} 3 & 7 & 9 \\ 2 & 7 & 1 \end{bmatrix}$$

the operation DT=DT11_DT21 will result in

$$DT = \begin{bmatrix} 4 & 8 & 3 \\ 6 & 5 & 5 \\ 2 & 5 & 5 \\ 3 & 7 & 9 \\ 2 & 7 & 1 \end{bmatrix}$$

In general, if DT11 is a matrix with dimensions NxM and DT21 is a matrix with dimensions QxM, the expression DT=DT11_DT21 will result in a matrix with dimensions (N+Q)xM that will be assigned to the DT variable.

Unitary ! This operator constructs a matrix with ones in the main diagonal and zeroes in the remaining positions, i.e., $a_{ij} = 1$ if $i = j$, and $a_{ij} = 0$ if $i \neq j$. Its operand has to be a vertex that specifies its dimensions (rows, columns). The matrix does not necessarily have to be square, although if it is, an identity matrix will be constructed. The names of the rows and columns, although not normally used, are generated to be compatible with the rest of the matrices and are numbered

consecutively (row1, row2, column1, column2, etc.). For example, the expression DONE=!(3,4) will result in a type D matrix with the name ONE. In other words

$$\text{DONE} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

and the expression DUNIT=!(3,3) will result in

$$\text{DUNIT} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

If DA is a type D matrix called A with dimensions 3 x 3, the expression DIDENT=![(:DA),("DA)] will have the same result as DUNIT=!(3,3) since :DA = "DA = 3.

13.3 Some Applications

Introduction

This section presents examples that use the operators and the arithmetic matrix language to solve familiar expressions in input-output analysis. The reader may find it useful to do these examples in the microcomputer while reading this section. For that purpose, we will use the hypothetical input-output matrix shown in Table 13.1 and we will assume that we have already created it with the name IO (type D) using the DEFINE and MATRANS programs. When we create these matrices, however, we do not introduce all the titles of Table 13.1 (that is not even possible), but we limit ourselves to naming columns (and corresponding rows) as S1, S2, S3, C and I, and the rows of the value added submatrix as WS, WUS and GOS. Since MATRANS calculates coefficients, we have created the matrix without including the corresponding row and column totals (gross value of production, GVP). As we have doubts regarding the row and column totals, we will calculate them using the "<" and ">" operators of the AMATRIZ program.

Table 13.1. Example of input-output matrix

Products			P U R C H A S E R S					
			Intermediate			Final		
Inputs			S1	S2	S3	C	I	VBP
S E L L E R S	Industries	S1	70	120	90	70	50	400
		S2	80	150	30	150	90	500
		S3	30	60	10	60	40	200
	Other	WC	70	10	15			
		WNC	30	30	25			
		EBE	120	130	30			
GVP			400	500	200			1100

In order to do this, we call the program from the main menu, press the "F3 Calc." key and type

DSUMCOL=<DIO

This expression means: create a type D matrix with the name SUMCOL and assign as components the sum of each one of the columns of the type D matrix called IO. To calculate the sum of the rows, after finishing this calculation press F3 and type

DSUMROW=>DIO

After finishing both calculations, we can see that the window showing the current matrix directory has incorporated the names and the respective dimensions of the matrices just created. Then, to

verify the sums of Table 13.1, we can load the SUMCOL and SUMFIL matrices in MATRANS or print them using the IMPRIME program.

The "open" Leontief model

Since we are interested in the input-output model, we should start by calculating the coefficients of the IO matrix. Even though we can perform this calculation using ARIMAT, it would be simpler to use MATRANS. To do that we call the program, load (F3 from set 1) the matrix and normalize it (F7 from set 2). By doing this, we obtain a type C matrix that we save with the same name as the original (IO). Now we have all we need to begin our exercises.

We will first be interested in having coefficient matrices $\mathbf{A} = [a_{ij}]$ and $\mathbf{V} = [v_{hj}]$, for current inputs and value added, respectively. Since these matrices are partitions of the recently calculated coefficient matrix, to obtain them we will use operators "extract" (@), "partition" (#), and "vertex" (,) as follows. First we will determine the vertex of the partition we want to extract; beginning with matrix $[a_{ij}]$ the corresponding vertices are: upper left = (1,1), and lower right = (3,3). That is, we want to extract the block (1,1)#(3,3), and we want to extract it from the type C matrix called IO. Now we can call the ARIMAT program, press the "F3 Calcula" key and type

CA=(1,1)#(3,3)@CIO

which means: create a type C matrix with the name A and assign it the partition delimited by vertices (1,1)#(3,3) extracted from the type C matrix called IO. In the same way, to obtain the $[v_{hj}]$ matrix we again press the F3 key and type

CV=(4,1)#(6,3)@CIO

Then, expressions

DDD=(1,4)#(3,5)@DIO

DGVP=(1,1)#(1,3)@DSUMCOL

provide us with the final demand matrix and the vector of gross value of production.

Now, with the matrices recently calculated, we can write the equation of the "open" Leontief model, i.e.

$$\mathbf{X} = (\mathbf{I} - \mathbf{A})^{-1}\,\mathbf{D}$$

in the following manner:

```
DX={^[!(3,3)-CA]}*(>DDD)
```

This expression may be interpreted as follows. Invert the difference between a 3x3 identity matrix and the type C matrix called A, postmultiply the result by a vector equal to the sum of the rows of type D matrix called DD, and assign the result to a type D matrix named X. Now we can verify that, obviously, DX =~DGVP.

The input-output price model

According to the input-output price model, given constant profit margins, prices of produced goods depend on physical input (produced and non-produced) coefficients and on prices of non-produced inputs. Thus, if we denote the index of prices of sector j by z_j , it will be equal to

$$z_j = (a_{jj} + t_j + g_j)\, z_j + \Sigma_{i \neq j}\, a_{ij}\, z_i + \Sigma_{h \neq t,g}\, v_{hj}\, z_h \qquad [13.1]$$

where t_j is the sales tax rate, v_{hj} are the value added coefficients, excluding those for sales taxes and profit margins, z_h is the index for the respective prices, and g_j is the profit margin. The set of all expressions [13.1] for $i = j = n$ constitutes a system of equations that, written in matrix form, allows us to express column vector $[z_i]$ as

$$\mathbf{z} = [\mathbf{I} - (\mathbf{A}' + \hat{\mathbf{t}} + \hat{\mathbf{g}})]^{-1}\, \mathbf{V}'\mathbf{z}_h \qquad [13.2]$$

where **I** is the identity matrix, **A** is the transactions matrix, ' indicates transposing, $\hat{\mathbf{t}}$ and $\hat{\mathbf{g}}$ are diagonal matrices whose non-zero elements are the coefficients of the sales taxes and of the profit

margins, respectively, **V** is the value added coefficient matrix (excluding sales taxes and profit margins), and $\mathbf{z}_h$ is the vector containing the price indices of the remaining non-produced inputs and transfers, that are assumed to be exogenous to the system.

Now we can use our matrix in Table 13.1 to exemplify the use of the program. For that purpose, and with the objective that the matrix presents the data in the required form, we have: i) separately stated sales taxes, ii) expressed gross operating surplus at factor cost, and iii) assumed that profit margin g_j is equal to the gross operating surplus coefficient. We assigned the results, shown in Table 13.2, to type C matrix named IOMOD, that constitutes our starting point.

Matrix **A**' in expression [13.2] is simply the transpose of CA obtained in the preceding section. If we name the transpose of CA as CATR, we can obtain it by

```
CATR=~CA
```

To obtain diagonal matrices $\hat{\mathbf{t}}$ and $\hat{\mathbf{g}}$, that we will call CTDIAG and CGDIAG, we use the following expressions:

```
CTDIAG=/{[(6,1)#(6,3)]@CIOMOD}
CGDIAG=/{[(7,1)#(7,3)]@CIOMOD}
```

Table 13.2. Coefficient Matrix

	S1	S2	S3
S1	0.175	0.240	0.450
S2	0.200	0.300	0.150
S3	0.075	0.120	0.050
WS	0.175	0.020	0.075
WUS	0.075	0.060	0.125
T	0.050	0.030	0.030
GOS	0.250	0.230	0.120
GVP	1.000	1.000	1.000

Source: Table 13.1

(Note that the [] could have been avoided). Similarly, we obtain the **V**' matrix, which we will call CVTR, by doing

```
CVTR=~{[(4,1)#(5,3)]@CIOMOD}
```

Finally, we have to create vector $\mathbf{z}_h$ using DEFINE and MATRANS. We will call it IZH (i.e., ZH, type I) and it will be a column vector whose rows will have the names of the non-produced inputs, i.e. WS and WUS. Let IZH be, for example,

```
IZH = 1.30
      1.10
```

which indicates that nominal wages of skilled laborers rose 30 percent, while those of unskilled laborers rose only 10 percent.

Now we have all the matrices needed to calculate expression [13.2]. This may be written in our arithmetic matrix language as

```
IZ=^[!(3,3) - (CATR + CTDIAG + CGDIAG)] * CVTR * IZH
```

by so doing, we will obtain

```
IZ = 1.22
     1.20
     1.21
```

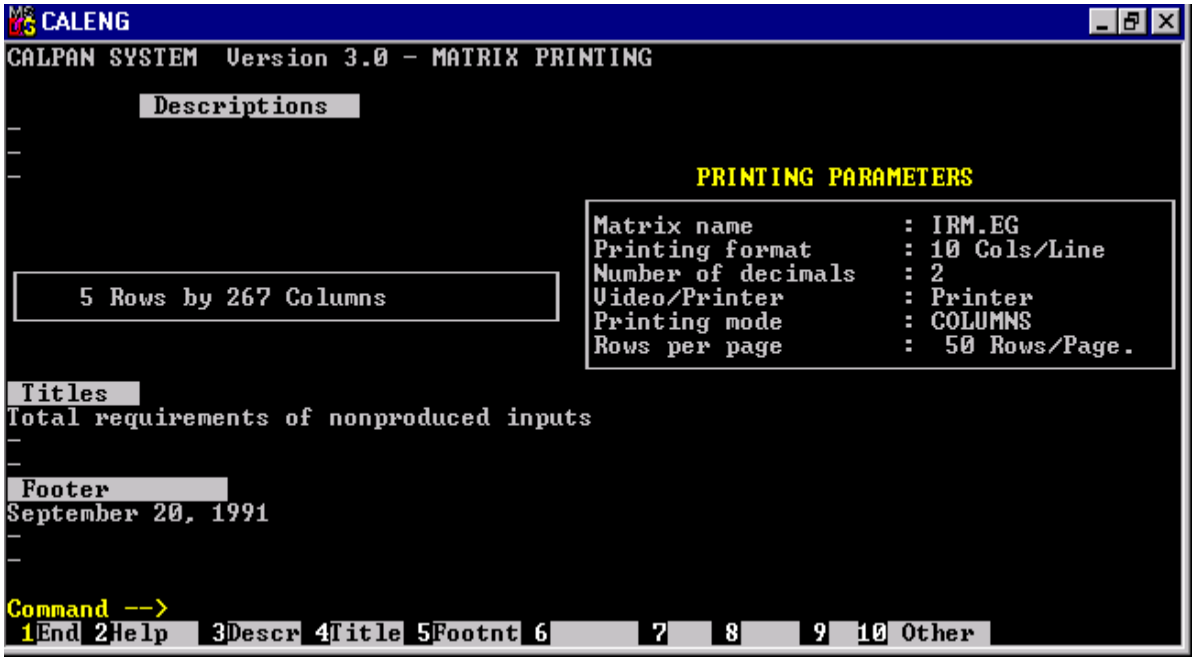
In other words, prices of goods S1, S2 and S3 would rise 22, 20 and 21 percent, respectively.

CHAPTER 14

THE PRINTING OF MATRICES

IMPRIME is a program for printing matrices on the printer or on the screen. Upon starting it from the main menu, IMPRIME displays a window showing the printing parameters and the first set of the program's function keys. (Table 14.1 lists the two sets of function keys). The printing parameters have initial default values that are shown when IMPRIME is started. These values can be changed using the function keys described later in this chapter. The program also allows for the specification of titles and footnotes. All of the printing parameters are copied to a file with the same name as the matrix to be printed and with extension ".EXP", where X is the type of matrix to be printed.

The IMPRIME program can print matrices in the COLUMNS mode or in the ROWS mode (transposed matrices), and can print numerical values in different forms (see function "F9 Dec" in Chapter 5). If you use the COLUMNS mode, the matrix in memory is printed in vertical blocks (horizontal in the case of the ROWS mode); in other words, first all of the rows of a



Screen 12. Program for the printing of matrices (IMPRIME)

Table 14.1. IMPRIME function keys

GROUP 1	GROUP 2
F1 Exit	F1 Exit
F2 Help	F2 Help
F3 Load matrix to be printed	F3 Enter descriptions
F4 Print the matrix	F4 Enter titles
F5 Printing mode (ROW/COL)	F5 Enter footnotes
F6 Not implemented	F6 Rows per page
F7 Print on SCREEN/PRINTER	F7 Not implemented
F8 Format (col./page)	F8 Not implemented
F9 Number of decimals	F9 Not implemented
F10 Switch to Set 2	F10 Switch to Set 1

block of columns are printed, then all of the rows of the next block of columns and so on.

14.1 First Set of Function Keys

F1 Exit (Sets 1 and 2)

This function returns the program to the menu after saving the printing parameters in a file whose name is the name of the matrix with the extension ".EXP", where X is the type of matrix.

F2 Help (Sets 1 and 2)

Displays a help text and then returns to the printing screen (see Chapter 2).

F3 Load

This function loads to memory the matrix to be printed. Upon pressing the F3 key, IMPRIME asks the user the type of matrix to be loaded and its name. If the file exists on disk with the same name as the matrix with the extension ".EXP", IMPRIME will read its previously assigned printing parameters; if the file does not exist, the program will assign default values to all of the printing parameters. The F3 function shows the name of the matrix in the printing parameters window.

F4 Prnt

F4 is the function that will print the matrix on the screen (video) or the printer according to the parameter value V/Prt. If you select the printer (Prnt), IMPRIME will ask you to prepare the printer before starting (in other words, to make sure that the printer is on, that it is "online", the paper is properly adjusted, etc.).

Before printing, you must set the printing parameters to the desired values. This is done using the function keys described in this chapter. Once the printing is started, on the screen or on the printer, it can be canceled by pressing the "c" key.

In order to print the maximum possible number of columns on a standard page using a "dot matrix" printer, IMPRIME will send a control character to the printer so that it will print in compressed mode (132 characters per line); this character control is the ASCII 15 (Hexadecimal OF). Upon finishing the printing, the program sends the character 18 (Hexadecimal 12). If your printer is not "dot matrix", these control characters may be interpreted by the printer in an unpredictable manner.

F5 Mode

The F5 key allows the user to change the printing parameter "Printing Mode" from COLUMNS to ROWS and vice versa. If you select the ROWS mode, the matrix will be printed transposed; this can be useful when printing type G matrices that normally have many more columns than rows.

F6 V/Prt.

The F7 key allows the user to specify the type of printing desired (on the screen or on the printer). Each time the F7 key is pressed, the program toggles from one to the other. If you choose the screen, IMPRIME will wait for you to press <ENTER> after showing each page. You may cancel the process by pressing the "c" key (cancel). If you choose the printer, you may also cancel the operation using the "c" key.

F7 Rows/Pag

With this function, the user can control the maximum number of rows from the matrix that will be printed on each page. The default value is 50 rows/page, which will fit on paper 11 inches long. If you use paper of a different length, you must modify this parameter using the F6 key. It is possible to print vertical sections of the matrix (with only one heading and one footnote) by changing this parameter to the number of rows in the matrix; in this case, the rows will print one after another without any space between pages. If you do not modify the printer defaults (changing switch settings), the program will print 6 rows per inch. If this is modified, it is possible to print 8, 10 or up to 12 rows per inch. This parameter is ignored if the printing is on the screen.

CALPAN uses 12 lines of the page to print the titles and footnotes. The user should keep this in mind when figuring the number of rows that will fit on a printed page.

F8 Cols/Lin

This function allows the user to control the number of columns of the matrix that are printed on each line of the screen or printed page. If you select the screen, the program will always display 5 columns of numbers on each line since the screen has a limit of 80 characters per line; if you specify a number of columns per line other than 5, IMPRIME will still use only 5. If you select the printer, the user may enter a value up to 20 for this parameter (if you use a wide carriage dot matrix printer). When using paper with a width of 8 and 1/2 inches (21.6 cm) you may print up to 10 columns per line.

F9 Dec

This function key allows you to specify the printing format for the numerical cells of the matrix, that may be exponential, integer or decimal (with 1-5 decimal places: see function "F9 Dec" in Chapter 5).

F10 Others

By pressing this function key, the program toggles between the two sets of function keys.

14.2 Second Set of Function Keys

F3 Descr

This function allows the user to enter three descriptions. Each one may be no longer than 27 characters. A description is a text that the program prints on the upper left corner of each printed page (above the row names). Upon pressing the F3 key, IMPRIME will ask you to enter three descriptions one by one, followed by ENTER. The descriptions will appear on the screen as they are typed. By pressing F1, the program returns to the Command state.

F4 Title

This function allows you to enter up to three lines of titles, each one containing up to 79 characters, which will be printed as headers on the screen or printed page. When you press F4, IMPRIME places the cursor at the first title line and waits for the user to type each one followed by ENTER. If only the ENTER key is pressed (without titles), the cursor moves to the next line. If F1 is pressed, the program returns to Command. If you "print" to the screen, only the first 48 characters from each line of titles will be displayed to avoid disordering the margins.

F5 Footnt

This function operates exactly as F4, except the text appears at the bottom of each page (footnote).

CHAPTER 15

FILE MANAGEMENT

The purpose of the ARCHIVOS program is to facilitate the management of files without exiting the CALPAN system. Upon starting this program, two working windows appear on the screen, one of which displays the seven available functions. To select one of these functions, place the cursor at the name of the function and press ENTER. The program will show the name of the function that can be selected in reverse video. The program also includes function keys "F1 Exit" and "F2 Help" which are described in preceding chapters.

The functions available in the ARCHIVOS program are as follows:

1. Directory Displays on the screen the files in the selected directory.
2. Erase Erases a file from the current directory.
3. Print Sends to the printer the contents of a file in the current directory.
4. Copy Copies the contents of one file to another in the current directory.
5. Rename Changes the name of a file in the current directory.

Screen 13. Program for file management (ARCHIVOS)

6. Backup Selectively copies the files of an existing directory to another existing directory (in the same or in another disk unit).
7. ChangeDir Changes the current data directory.

These functions are executed on the current data directory, whose name is shown on the screen, except for the functions "Directory" and "Backup" in which the user may specify the disk drive and the directories.

Directory

This function can show the names of the files contained in any directory on any disk. Upon selecting it, the program erases the right window on the screen and asks for the disk drive and/or the directory whose contents you want to know, through the message:

```
d:\directory name -->_
```

If you press only the enter key, the program will show the contents of the current data directory (that shown on the upper part of the screen). Upon starting, the ARCHIVOS program always executes this function automatically, so the user can see the files contained in the current data directory immediately after starting the program from the main menu. If you wish to know the contents of another directory, you must enter its name, followed by ENTER. For example:

```
d:\directory name -->B:\MATRICES<ENTER>
```

will list the names of the files found on disk B:, directory \MATRICES. Note that you have to type the character "\".

If the directory does not exist on the specified disk, the program will not list any file name. If a directory does not contain files or you do not type the character "\", the program will give the message, "File not found."

The meaning of CALPAN file extensions is described in Appendix B.

Erase

This function allows the user to erase a file from the current directory. Upon executing it,

the program will ask for the name of the file to be erased. The file names should be specified in the following manner:

<filename>.<extension>

where <filename> is a name of 1-8 letters or digits, beginning with a letter, and <extension> is another name of 1-3 letters or digits. The period that separates the filename from the extension is necessary and required, and there must not be blank spaces in the whole filename. Appendix B presents the file system used by CALPAN and the meaning of the extensions. If the specified file does not exist in the current directory, the program will give an error message and return to the menu.

This function should be used with care since the program will erase the file from the directory. If you do not have a back up copy, you could lose weeks of work.

Print

This function allows the user to have a printed copy of the information contained in a file in CALPAN internal format. It functions similar to the PRINT program of the operating system. Upon executing this function, the program will ask for the name of the file to be printed; if a name is not specified, the program will return to menu. If you specify a file that does not exist in the current directory, the program will give an error message. Upon starting the printing, the program will give the following message:

```
Printing NAME.EXT, press C to cancel
```

If you press the “c" key, the program will cancel the printing and return to the menu.

Copy

This function permits the user to copy a file from the current directory to another file, with a different name, in the same directory. Upon executing this function, the program will ask for the name and the extension of the file to be copied. If you specify the name of a valid file, the program will ask for the name and the extension of the file where the contents of the first are to

be copied. If the latter file already exists in the current directory, the program will ask the user whether to replace the old information with that from the first file or to cancel the copying operation. Note that this program copies files, not matrices. A matrix is composed of two files of the same name, but different extensions (see Appendix B).

Rename

The function "rename" permits the user to change the name of a file found in the current directory. Upon executing this function, the program will ask for the name of the file to be renamed. You must enter the complete name (including the extension) followed by ENTER. If the file specified does not exist in the current directory, the program will give an error message. If the file exists, the program will ask for the new file name with the message

```
NAME.EXT new name?_
```

such that the user will enter the new file name (including the extension). If this new name is the name of a file that already exists, the program will give the message

```
NAME.EXT already exists, press ENTER
```

Upon doing so, the program will return to the menu of the ARCHIVOS program.

Backup

This function allows the user to selectively copy all of the files contained in a directory to another existing directory, which can be on another type of disk. This function may used for back up as well as to retrieve files from the CALPAN data directory. The function may be used for any directory.

Upon executing this function, the user will have to provide the source disk drive and directory (from where the files are to be copied) editing the current unit and directory that the program takes as default value, or accept them by pressing <ENTER>.

Next, the program will ask for the disk drive and the directory where the files are to be copied (it must be the name of an existing directory). If you press F1 during any of these steps,

the program will return to the menu. If you press only ENTER when the program asks for the "destination" disk drive, the program will assume the current disk by default. If you press only ENTER when the program asks for the "destination" directory, the program will assume the ROOT DIRECTORY in the destination disk drive. The program will immediately display the names of the origin and destination directories such that the user can verify that they are correct; the program will then ask you to place disks in the appropriate drives and will wait until the user presses any key to continue. If at this moment the user presses F1 or “c", the program will cancel the back up operation and return to the menu.

If the user decides to continue, the program will continue by showing the file names that are being transferred, through the message:

```
NAME.EXT --> <destination disk drive and directory> NAME.EXT (Y/N)_
```

where <directory> is the name of the directory to which the file "XXXXXXXX.EEE" will be copied. The program will ask permission to complete the copying through the message (Y/N)_. If you wish to copy this file, press the "Y" key; if you do not wish to copy it, press the "N" key or ENTER. In this way you can make a selective transfer of files from one directory to another and back up or retrieve only those files you want. The program will ask for permission to transfer each one of the files in the origin directory and upon finishing will inform that the back up has been completed with the message:

```
End Backup
Load Calpan system disk if necessary
Press any key when ready
```

This message is included to facilitate the work on computers that do no have a hard disk drive.

ChangeDir

This function serves to change the name of the current data directory or that where CALPAN will look for the data files and copy the matrices it saves. Upon executing it, the program will ask for the name of the disk drive where the new data directory is found; the user must press the letter that identifies the disk drive (A, B, C...Z) followed by ENTER. Next, the

program will ask for the name of the directory. *This name must be that of an existing directory*, created by use of the command "MD" from the operating system, since CALPAN cannot create its own directories.

If you want to work in the root directory (not in a directory), press only the ENTER key when the program asks for a directory name, in which case the current directory will be A:\, B:\ or C:\ depending on the disk drive selected.

CHAPTER 16
FILE CONVERSION

The TRADUCE program includes five functions for the conversion of files between various format types with the purpose of facilitating the interchange of information between and other programs.

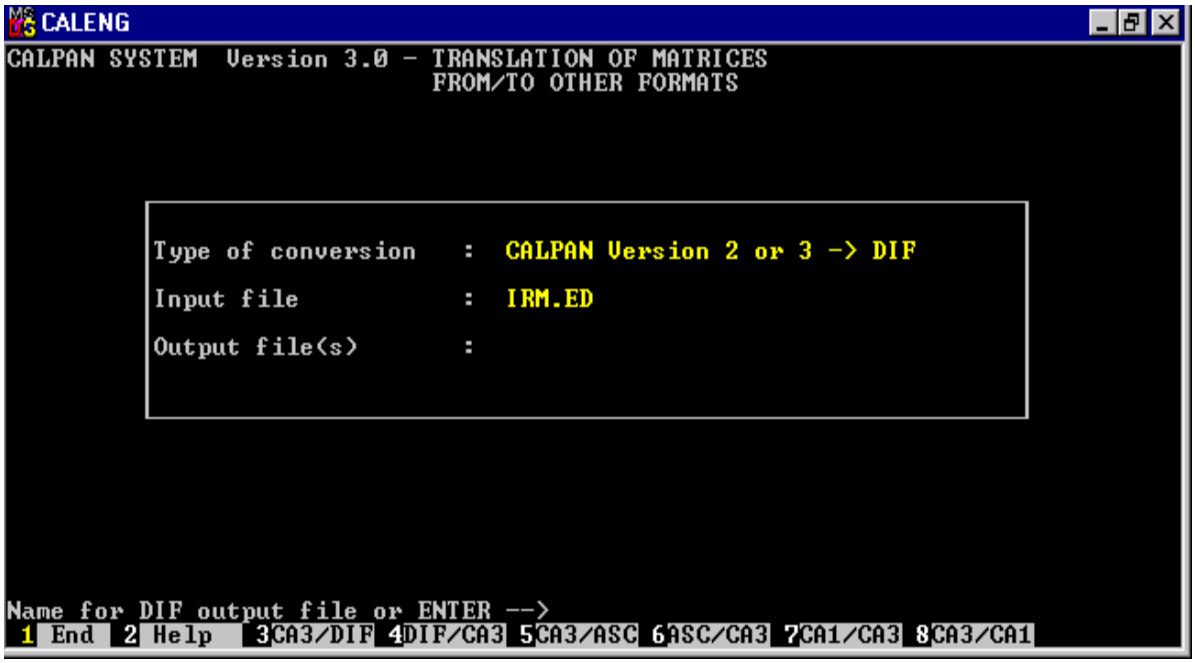
The functions available in the TRADUCE program, in addition to F1 Exit and F2 Help, are the following:

F3 CA3/DIF	Conversion from CALPAN format to DIF format.
F4 DIF/CA3	Conversion from DIF format to CALPAN format.
F5 CA3/ASC	Conversion from CALPAN format to ASCII format.
F6 ASC/CA3	Conversion from ASCII format to CALPAN format.
F7 CA1/CA3	Conversion from CALPAN (Ver. 1.2) format to CALPAN (Ver. 3.0) format.
F8 CA3/CA1	Conversion from CALPAN (Ver. 3.0) format to CALPAN (Ver. 1.2) format.

The F7 CA1/CA3 and F8 CA3/CA1 functions have been included for the conversion of files created with CALPAN Version 1.2. Appendix B provides a description of the CALPAN , DIF and ASCII formats.

As always, to execute one of these functions, you only need to press the corresponding function key, and the program will ask for the necessary information to complete the desired conversion.

At all times, the names of the files involved in the conversion appear in the work window as well as the type of conversion being completed. The program assigns names by default to the exit files. If the user wants these matrices to have these names, simply press the ENTER key when the program asks for such names. The exit default name is the same as the entrance name, except for the extension, which depends on the type of conversion to be completed. Once the entrance and exit files have been specified, TRADUCE will start the required operation.



Screen 14. Program for the conversion of Matrices (TRADUCE)

F3 CA3/DIF

Upon pressing the F3 key, the program will ask for the name and type of matrix to be transformed to the DIF format. This file will always have the extension '.DIF'. If you press only ENTER, the file will copy the assigned name by default. TRADUCE will copy the DIF file in the format specified in Appendix B.

F4 DIF/CA3

When you press the F4 key, TRADUCE will ask for the name of the DIF file to be converted to CALPAN, as well as the type of exit matrix. Then the program will ask for the name of the matrix to be copied. If you press only ENTER, it will take the name of the entrance DIF file by default.

The DIF file must contain information that permits the program to construct a valid CALPAN matrix. If it does not, the program will give an error message an will not copy any entrance matrix. The DIF file should be constructed as shown in Appendix B.

F5 CA3/ASC

This function allows the user to convert a CALPAN matrix to ASCII format. Upon executing it, the program will ask for the name and type of matrix to be converted. Next, the program will ask for the name of the ASCII file to be copied. It will accept the name by default by pressing only ENTER.

Upon converting to the ASCII format, TRADUCE copies two files with the same name, but different extensions. The file with the extension '.ASC' contains the numerical values of the matrix while the file with the extension '.DIM' contains its dimensions. The format of these files is described in Appendix B.

F6 ASC/CA3

With this function, you can convert the numerical values of matrices created with other programs to CALPAN format. Upon executing it, the program will ask the user the name of the ASCII file to be converted, as well as the type of matrix to be created. If the ASCII file exists in the current directory, the program will ask for the name of the matrix to be copied. It will accept the name by default if only the ENTER key is pressed. If the program detects any error at the moment it is loading the file, it will send an error message and will not copy an exit file.

The F6 ASC/CA3 function uses a file that contains numerical values, has an extension '.ASC' and corresponds to the format in Appendix B as an entrance file. However, it generates the two files that can produce a CALPAN matrix, in other words, one that contains the numerical values and another that contains the names of the rows and columns. This Traduce function automatically generates the names of the rows and columns, numbering them consecutively.

F7 CA1/CA3

This function is provided to satisfy the need of transferring matrices from Version 1.2 to Version 3.0 of CALPAN. Upon executing this function, the program will ask for the name and the type of the entrance matrix (Ver. 1.2). If the entrance matrix exists in the current directory, the program will ask for the name of the exit matrix. It will accept the name by default if you press only ENTER.

F8 CA3/CA1

This function uses a Version 3.0 matrix as an input, to generate a Version 1.2 matrix as an output file.

APPENDIX A
INSTRUCTIONS FOR THE INSTALLATION OF THE CALPAN SYSTEM

The CALPAN system is provided on two double sided, double density floppy disks. Table A.1 shows the files that come on these disks and Table A.2 shows the minimum equipment necessary to run the system.

CALPAN Version 3.0 uses a work space in memory with a maximum capacity for matrices of up to 200 rows x 200 columns. However, it can operate with matrices up to 300 x 300. For that purpose, it requires 1 Mb of working disk space in the directory where the program is installed. For its correct operation, CALPAN requires that there must not be any programs in memory other than itself and the operating system.

A.1 Floppy Disk Installation

Although installing CALPAN in a 1.2 Mb diskette is technically feasible, it will only be operationally practical when using matrices not greater than 200x200. A diskette based system will be extremely slow when used with matrices larger than indicated.

Before using the CALPAN system, it is convenient to make a working copy of the distribution disks, and to keep the original disks in a safe place.

To start the CALPAN system, type:

calpan<ENTER>

and the system's main menu will appear on the screen. The first time you call the program, you will have to wait approximately fifteen minutes while the program creates and initializes two files (CALPANR.MTX y CALPANC.MTX) that will later be used for work space.

Note that CALPAN will look for data in the directory A:\ in disk drive A. To change the current directory, use the Change_Dir function from the file administration program (ARCHIVOS).

Table A.1. Files of the CALPAN system *

Programs	Help Texts
CALPAN.COM	AYUDA100.HLP
DEFINE.COM	AYUDA200.HLP
MATRANs.COM	AYUDA300.HLP
CALCULAG.COM	AYUDA400.HLP
AJUSTE.COM	AYUDA500.HLP
PRECIOS.COM	AYUDA600.HLP
BIENES.COM	AYUDA700.HLP
CALCID.COM	AYUDA800.HLP
FUNCION.COM	AYUDA900.HLP
RPC.COM	AYUDA910.HLP
IMPRIME.COM	AYUDA920.HLP
ARCHIVOS.COM	AYUDA930.HLP
TRADUCE.COM	AYUDA940.HLP
ARIMAT.COM	AYUDA950.HLP

* Creates two additional files, CALPANR.MTX y CALPANC.MTX, that are used as a working space.

Table A.2. System and equipment requirements to operate the CALPAN system

Microcomputer:	IBM-PC o compatible
Operating System:	PC-DOS Ver. 2.1 or later. MS-DOS Ver. 2.1 or later.
Memory (RAM):	640K
Disk drives:	Hard disk. (Use of a two disk drives system is feasible if at least one of them has a capacity of 1.2 Mb or more. However, in that case the operation of the system will be very slow when using matrices greater than 200x200)
Printer:	dot matrix
Monitor:	monochrome, CGA, EGA, VGA

Do not try to copy matrices on the CALPAN disk since there is not enough space. Use a newly formatted disk (FORMAT command of the operating system), placing it in disk drive B.

A.2 Hard Disk Installation

To install CALPAN on a hard disk, it is only necessary to copy the system files (listed in Table A.1) to a directory created on the hard disk. To do it, follow these steps:

1. Make the operating system point to drive C:, and the message C> o C:\> will appear.
2. Place the 1/2 CALPAN distribution disk in drive A:.
3. At the ">", type the following:

   md cal<ENTER>

   which will create a subdirectory called "CAL". You may use any name you want for the subdirectory.
4. Next, type:

   copy a:*.* c:\cal<ENTER>

   and the operating system will copy all of the files from the floppy disk to the subdirectory "CAL" on the hard disk, and will show the name of each file as it is being copied.
5. Take the 1/2 CALPAN distribution disk from disk drive A and put it in a safe place. Place the 2/2 CALPAN distribution disk in drive A: and repeat 4.
6. To start the system, type the following after the ">" sign:

   cd cal<ENTER>

   and the operating system will call the subdirectory "CAL", and will give the message "C>" or "C:\CAL>". Next, in order to call the CALPAN program, type

   calpan<ENTER>

and CALPAN's main menu will appear on the screen. If you prefer to do this with one command, type the following when you receive the message "C>" o el "C:\>":

c:\cal\calpan<ENTER>

If you gave the subdirectory a different name, you must enter that name in place of CAL. The first time you call the program, you will have to wait a few minutes while the program creates and initializes two files (CALPANR.MTX and CALPANC.MTX) that will later be used as a working space.

Upon starting CALPAN, it assumes that the current directory is that from which it was called. For example, on the preceding case, CALPAN will look for the current directory in the subdirectory C:\CAL, which is from where it was called. If you want your data to be in another subdirectory, you must create it first using the MD (Make Directory) command of the operating system. For example, if you give the following instruction from the root directory (C:\>)

md matrices <ENTER>

the system will create the directory C:\MATRICES on the hard disk. Once this subdirectory has been created, you can have CALPAN use it as the current directory by executing the Change_Dir command from the ARCHIVOS program.

CALPAN can be started from a directory other than the one containing the system; for example, from the data directory. To do this, you should type the name of the subdirectory where CALPAN is located and then the name of the program. For example, if CALPAN is in the directory C:\CAL and the data is in C:\MATRICES, you could call CALPAN from the latter with the command

C:\cal\calpan<ENTER>

In this case, CALPAN will look for the data in the directory C:\MATRICES, i.e. that from which it was called. The drive name may be omitted if both directories are on the same one.

If CALPAN was installed on a hard disk under a subdirectory (the subdirectory CAL, for

example) you may find it convenient to create a "batch" file in the root directory (with the extension .BAT) containing the instructions

```
ECHO OFF
CD <name of current directory>
<X=>\<directory>\CALPAN
CD \
ECHO ON
```

where <X=> is the disk drive, and <name of current directory> is the name of the subdirectory where the CALPAN system is located. In this case, CALPAN will be called from the subdirectory indicated in the second instruction, and will assume that it should look there for the data. This file can be created using the EDLIN program of the operating system or any text editor. For example, assuming that CALPAN is installed on the hard disk (drive C) under the subdirectory CAL, and the data in the subdirectory MATRICES, the file would be:

```
ECHO OFF
CD MATRICES
C:\CAL\CALPAN
CD \
ECHO ON
```

Once this file has been created, the user can call CALPAN directly from the root directory, typing the name of the "batch" file followed by ENTER. For example, assuming the user created the file CA.BAT in the root directory of the hard disk (drive C:), the CALPAN system is started by typing

```
C:\>ca<ENTER>
```

where the underlined text is all that you type. Logically, the subdirectories and the CALPAN system must exist or the system will give an error message. When you use this "batch" file, the operating system will return to the root directory in the disk drive where the batch file is located.

A.3 Configuration of the Operating System

Before using the CALPAN system, the user must verify that the file CONFIG.SYS (used by the operating system at the moment the system is initialized) meets the following conditions:

1. The minimum number of files that the operating system can open simultaneously must be equal to or greater than 16.
2. The number of work areas for data transfer to/from disk (BUFFERS) should be greater than or equal to 15.

If these conditions are not met, CALPAN may abort the loading or saving of a matrix, at which point it will give the message

```
I/O Error F3 PC=XXXX
Program aborted
```

This message indicates that you tried to open more files than the maximum allowed by the system ("Too many open files").

To meet these requirements, it is necessary that the file CONFIG.SYS of the operating system has, in addition to other needs for the installation desired, the instructions "Files=<N>" and "Buffers=<M>" where <N> and <M> are natural numbers greater than 16 and 15, respectively. For example,

```
<other instructions for the operating system>
FILES=20
BUFFERS=15
<other instructions for the operating system>
```

will be a configuration that allows CALPAN to operate without problems. To create or modify the file CONFIG.SYS, consult your system analyst.

APPENDIX B
THE FILE SYSTEM

B.1 CALPAN File Formats

The CALPAN system stores each matrix in a pair of files. In the first, the program saves the dimensions of the matrix as well as the names of its rows and columns; the second file contains the values of the nonzero numeric entries. Both files have the same name, but different extensions. The complete name of each file is written in the following manner:

Names File	<matrix>.<v><type>
Values File	<matrix>.<v><type>V

where <matrix> is the name the user assigns the matrix to be stored and <type> is the type of matrix (A, B, ..., Z), according to the description in section 2.1 of this manual. The element <v> in the name of the file is a letter that designates the format in which each file is recorded; for the 3.0 version of CALPAN, <v> has the value "E". For example the files MAT1.EG stores the dimensions, and the names of the columns and rows while MAT1.EGV stores the values of a type G matrix named MAT1 corresponding to the 3.0 version of CALPAN.

The names file is a text one composed of lines ending in the pair of characters <CR><LF> (carriage return and line feed). This means that it can be displayed on the screen using the TYPE command of the operating system. Table B.1 presents the corresponding format.

In order to store larger matrices more efficiently (up to 300x300), the values file is written as triplets <ROWi><COLUMNj><VALUE>, where each triplet is placed one after the other, without <CR> <LF> separators or blank spaces, and the numerical entries are copied in binary form. <ROWi> and <COLUMNj> are natural numbers that indicate the indexes of the matrix elements whose <VALUE> is other than zero. <VALUE> is always written in exponential notation to 11 digits of precision. The data each consist of 2 bytes each, whose binary values are, respectively, the row and column numbers of the nonzero elements of the matrix. <VALUE> uses 6 bytes to store the real number. In fact, each triplet is a record definable in Pascal in the following form:

Table B.1. Format of the names file

Contents of the line	Meaning
<NF> <NC>	matrix dimensions (*)
row1	name of first row
row2	name of second row
.	
.	
.	
rowNF	name of the NFth row
column1	name of the first column
column2	name of the second column
.	
.	
.	
columnNC	name of the NCth column

(*) <NF> is the number of rows in the matrix and <NC> is the number of columns, and they are separated by blank spaces.

```
TYPE Triplete = RECORD
                  row      : INTEGER  {2 bytes}
                  column   : INTEGER  {2 bytes}
                  value    : REAL     {6 bytes}
                END;                  {total 10 bytes per triplet}
```

(In Version 1.2 of CALPAN, the values are stored in a text file where each line ends with a <CR><LF> (carriage return, line feed), as presented in Table B.2).

B.2 The Structure of the DIF Format

DIF (Data Interchange Format) is a format for recording tabular data that can be read by many other programs. The format uses a series of descriptors to create text files compatible with application programs, languages and operating systems.

The Dif format was designed mainly to be used with tabular data, organized in rows and columns, such as those used for a spreadsheet. DIF uses the terms "vector and "tuple" to designate rows and columns, respectively. The DIF files consist of two sections: a section of headers and a section for data. Each section consists of a series of elements each of three lines,

Table B.2. Format of the values file in Version 1.2

Line No.	Contents of the line	Meaning
1	<ROWi> <COLUMNj> <VALUE>	Numerical value of the nonzero element placed in the position (i,j)
2	<ROWk> <COLUMNr> <VALUE>	Numerical value of the nonzero element placed in the position (k,r)
. . .		
M	<ROWn> <COLUMNr> <VALUE>	Numerical value of the nonzero element placed in the position (n,r)

and follows the format:

<Keyword>
<vector #>, <numerical value>
"<text string>"

where keyword is a reserved word that indicates the type of row descriptor, <vector #> is equal to zero, and <text string> contains a name if <vector #> is equal to one.

To record a CALPAN matrix in DIF format, the program considers that the matrix is in memory as if it had been created by a spreadsheet, in the format shown in Table B.3.

CALPAN copies each one of the DIF file elements row by row; i.e. first all of the elements from tuple 1, and then all of the elements from tuple 2, etc. Some application programs recognize this copying method as "Column major order", even though they actually copy the matrix entries row by row.

When CALPAN converts a DIF file to the CALPAN format, it expects that the file was recorded in that same order, and also, that the matrix has been constructed with the structure shown in Table B.3. If this is not the case, CALPAN will give an error message, since it will not be able to construct an IRM using a different format. In Table B.3, the entries between quotes

Table B.3. Format of the CALPAN matrix in memory for conversion to DIF

column	A	B	C	D	...	J	...	X
row	1	2	3	4	...	j+1	...	m+1
1	"TIT"	"C1"	"C2"	"C3"		"Cj"		"Cm"
2	"F1"	M11	M12	M13		M1j		M1m
3	"F2"	M21	M22	M23		M2j		M2m
i+1	"Fi"	Mi1	Mi2	Mi3		Mij		Mim
n+1	"Fn"	Mn1	Mn2	Mn3		Mnj		Mnm

"TIT" = "name of the CALPAN file"; "Fi" = name of row *i*; "Cj" = name of column *j*.

represent alphanumerical values (chains of characters) and the others represent numeric entries of the *n* by *m* matrix.

"TIT" = "name of CALPAN file"

"Fi" = name of row i

"Cj" = name of column j

Table B.4 shows a scheme of the DIF file structure produced by the CALPAN system.

B.3 The Structure of the ASCII Format

The ASCII format provides great flexibility for the exchange of information between programs, at the expense of a greater standardization and additional disk space, since text files are recorded (ASCII). CALPAN, uses ASCII format only for the files that describe the dimensions and the column and row names of the matrix (extension '.E'<type>').

The format described in this section is more general than the one used by CALPAN, and can be used to exchange information between CALPAN and other programs, written in computer languages such as BASIC, PASCAL, C, FORTRAN, etc., or application programs capable of

Table B.4. Structure of the DIF file created by CALPAN

Element descriptor	Comments *
TABLE 0,1 ""	File heading
VECTORS 0,<number of columns> ""	Number of columns in the file (m+1)
DATA 0,0 ""	Starts the data section
-1,0 BOT	Start of tuple 1
1,0 "CALPAN matrix"	Title ("Tit")
1,0 "<name of column 1>" 1,0 "<name of column 2>" . . . 1,0 "<name of column m>	All of the column names of the CALPAN matrix (First tuple from DIF file)
-1,0 BOT 1,0 "<name of row i>" 0,<value Mi1> V 0,<value Mi2> V . . . 0,<value Mij> V . . . 0,<value Mim> V	Start of tuple number i Name of CALPAN row >--> Repeated for all rows (i = 1, 2, 3 ... n) >----------> All values for row i (j = 1, 2, 3 ... m)
-1,0 EOD	End of file

* Comments are not part of the file.

reading the ASCII format, since the files are copied in "free format"; i.e. the programs can read the numeric values of a matrix from a file as if they were reading from the microcomputer keyboard, following some simple rules.

In the case of the ASCII format, CALPAN only converts the numeric values of a matrix, since the row and column names are already copied in free format (see Table B.1). The matrix numeric values are copied in a file with the extension '.ASC' and are found ordered by row (Column Major Order); i.e. first row 1, then row 2, the 3rd, etc. Each numeric entry is copied in exponential notation in the form 1.2345678901E+12 with 11 digits of precision. Table B.5 shows the structure of the '.ASC' file. Note that all of the numeric entries are copied, even if they are zero.

For CALPAN to be able to read and construct an IRM using that information, the file must have the structure shown in Table B.5. Although CALPAN records all the numeric entries in exponential notation in order not to lose precision, it can read whole and decimal numbers from the ASCII file (see the end of Chapter 5).

When converting from its format to ASCII, CALPAN generates two files of the same name with different extensions. The first contains only the matrix dimensions and uses the extension '.DIM'. This file has just one line of text ending with the characters <CR><LF> and contains two numeric entries separated by blank spaces; the first entry represents the number of rows, while

Table B.5. Structure of the ASCII file produced by CALPAN

Line of ASCII File	Explanation
<M11><M12> <M13> <M14> ... <M1m><CR><LF>	row 1
<M21><M22> <M23> <M24> ... <M2m><CR><LF>	row 2
.	.
.	.
.	.
<Mi1> <Mi2> <Mi3> <Mi4> ... <Mim><CR><LF>	row i
.	.
.	.
.	.
<Mn1> <Mn2> <Mn3> <Mn4> ... <Mnm><CR><LF>	row n

<CR> = Character "Carriage Return" [ASCII(13)]
<LF> = Character "Line Feed" [ASCII(10)]
<Mij> = Entry i,j of the matrix in exponential form (1.2345678901E+12)
with 11 digits of precision and a total of 18 characters.

the second represents the number of columns of the matrix. The second file contains the matrix values specified in Table B.5.

How to create a CALPAN matrix in LOTUS™ or SYMPHONY™

"LOTUS 1-2-3" and "SYMPHONY" are integrated packages of multifunctional programs and share many file structures. Thus, when we refer to LOTUS, we will be talking about both programs: 1-2-3 and Symphony.

It is possible to create an intersectoral relations matrix (IRM or CALPAN matrix) using LOTUS and later converting it to the CALPAN format, using the DIF format as a link, since LOTUS provides a function to convert from its own format to DIF. Accordingly, a CALPAN matrix can converted to the DIF format and from there to LOTUS. In this section, we will only discuss the first part of the conversion (LOTUS -> CALPAN), since the second part is a trivial reading task in LOTUS.

To create an IRM in LOTUS, it is necessary to construct a spreadsheet (or model) like the one shown in Table B.3. In fact, this is how the LOTUS screen would look, except for the number of columns, which in LOTUS would be replaced by one or more letters (A,B,C,... AB, AC, etc.). The meaning of the entries or cells of the spreadsheet page is as follows:

a) "TIT" in cell A1 (1,1 in Table B3) should only be a text (string in Lotus), but CALPAN only uses it for identification and ignores it. The cell may be empty, but must not contain a number.

b) The entries in the cells B1, C1, D1,... ("C1", "C2", "C3"... in Table B3) should be the names of the columns of the IRM and be formed of text type (string) data. These names must follow the rules of CALPAN column names; in other words, they must begin with a letter, continue with letter or digits and can be no longer than 27 characters.

   If the width of the column in Lotus is such that all of the names cannot be seen on the screen, it can be expanded by using the "width" format command. This has no effect on the storage of data in the ".WKS" or ".DIF" files if the constraints

regarding the column names are not violated.

c) The entries in the cells A2, A3, A4... ("F1", "F2", "F3"... in Table B.3) represent the names of rows in the IRM and are subject to exactly the same constraints as the column names.

d) The entries Mij in the cells whose range is B2...X(n+1) represent the values of the IRM and must contain the numeric values or be left blank (empty), in which case the values are interpreted as zero.

Once the matrix has been loaded in LOTUS, it should be saved using the "FILE Save" function of the program, giving a name to the file. After saving it, select the TRANSLATE function from the main menu and then the subfunction DIF.

LOTUS provides two types of conversions to the DIF format:

a) Row Major Order

b) Column Major Order

Using type a) the matrix will be copied in the following order:

"TIT", "F1", "F2",..."Fn",
"C1", M11, M21, M31...Mn1,
.
.
.
"Cm", M1m, M2m, M3m...Mnm

Using type b) the matrix will be copied in the following order:

"TIT", "C1", "C2", "C3"..."Cm",
"F1", M11, M12, M13...M1m,
"F2", M21, M22, M23...M2m,
.
.
.
"Fn", Mn1, Mn2, Mn3...Mnm

This last copying order is the one CALPAN uses to convert a DIF file to CALPAN. Therefore, you must specify type b) Column Major Order.

APPENDIX C
USE OF DISK SPACE

The CALPAN system does not provide a mechanism to detect the space available on disk; thus the user should take some precautions. One is to verify that enough space exists on the system's work disk. This can be done using the commands DIR or CHKDSK of the operating system; these commands provide the available workspace on disk.

If the CALPAN system tries to copy a matrix or other file to disk, and there is not sufficient space, the program will cancel the operation giving one of the following error messages:

```
I/O Error F0, PC=XXXX
Program Aborted

I/O Error F1, PC+XXXX
Program Aborted
```

The code XXXX (in Hexadecimal notation) represents the address in memory where the program was executing the last instruction before canceling. The code F10 means that THERE IS NOT SUFFICIENT SPACE ON DISK TO COPY MORE INFORMATION. The code F1 means that THERE ARE NO FREE ENTRIES IN THE DIRECTORY TO COPY THE NAME OF THE NEW FILE.

In either case, the program will not copy the new information and the operating system will take control. Consequently, such that the matrix that was in memory at that moment will be lost. Thus, it is necessary that you estimate the amount of space you will need to work with the specific matrices of your project. To facilitate this estimation, follow these general guidelines.

A matrix is stored in two files (se Appendix B), the names file and the values file. The names file occupies a maximum of 19140 bytes
((300 rows + 300 columns) x 29 characters = 17400, plus 10% for security). The values file occupies a maximum, i.e. assuming all of the numeric entries are not zero, of 990000 bytes (300 rows x 300 columns x 10 bytes + 10% for security).

Therefore, to be able to save a matrix onto disk, it will be necessary to have the following space available:

File	Required disk space (in bytes)
Names	19140
Values	990000
Total	1009140

Note that the dimension of the matrices used (number of rows by number of columns) plays an important role. The conditions mentioned correspond to the maximum space needed. However, most of the time, the matrices only contain values in approximately 50% of the entries, and since only values other than zero are saved, the previous totals can also be reduced by the same percentage.

You can use a practical rule to estimate the necessary space, building on the dimensions, and the "density" (amount of nonzero elements) of the matrices with which you are working. Thus, N will be the number of rows and M the number of columns of the matrices you are using, and P the percentage of nonzero elements. The following formula calculates the minimum disk space necessary (E) to work with CALPAN.

$$E = \frac{N * M * P}{90000} * 1009140 \text{ (bytes)}$$

For example, assume that a project requires working with 50x120 matrices and the density of these matrices is 75% (75% of the values are nonzero). Thus, you need the following minimum space on disk:

$$E = \frac{50 * 120 * 0.75}{90000} * 1009140 = 50457 \text{ bytes}$$

This minimum space should be available on disk each time you try to save a matrix with these characteristics.

APPENDIX D

THE UPDATING FOR RELATIVE PRICE CHANGES

D.1 Introduction

This appendix explains the limitations, and the assumptions underlying the formulas that the AJUSTEG program uses to update the matrices of total requirements of nonproduced inputs and transfers (type G) for relative price changes.

Existing matrices, or the data to prepare them, often correspond to a date prior to changes in relative prices of nonproduced inputs (foreign exchange and labor, for example) that may have affected physical coefficients, the relative prices among the produced goods, and between them the traded goods. This may have affected the "tradeability" of some goods, i.e. goods that were nontraded before, now become traded and vice versa.

In the case of input-product matrices, there are updating methods that aimed at capturing the changes in relative prices as well as those in physical coefficients. The best known of these procedures is the RAS method and several authors have analyzed the margins of error involved in its use.[2] The characteristic common to all of these procedures is that they were designed keeping in mind input-output matrices inserted in a system of national accounts. When you use this type of matrix, and the data required by the RAS method is available, it is to be preferred to the method used by the AJUSTEG program. However, very often this data is not available, or *ad hoc* IRMs are used, what suggest the need to consider other methods.

Let us begin by distinguishing between the changes in the "tradeability" of the inputs on one side and the effects over the physical coefficients and relative prices on the other. The former should be resolved before trying to incorporate the latter. If you are working with an existing matrix, it will be convenient to return to the data used to prepare it and reconstruct the cost structures that have been affected by tradeabilty changes. On the other hand, if you try to prepare a matrix using the data from before the relative price changes, the problem of the change in

---

[2] For a description of the RAS method, see Bates and Bacharach (1963), United Nations (1973) or Bulmer-Thomas (1982). Allen (1974), Lecomber (1975) and Lynch (1979) provide evaluations of the method with empirical bases.

tradeability should be resolved during the preparation. In both cases, the result is an *ad hoc* matrix that reflects the tradeability of the goods corresponding to the situation after the price change, but whose value coefficients are those corresponding to the situation before this change took place. Therefore, it will be necessary to incorporate, the effects over the physical coefficients and the relative prices of the nonproduced inputs. The question is, how can we incorporate these effects into the matrix?

It will not be feasible to incorporate the physical coefficients unless you build a new matrix using the data that reflects the changes in relative prices, in which case both types of effects (quantity and price) will be incorporated in the new matrix. On the other hand, if you can assume that the effect over the physical coefficients can be overlooked, you may use a series of procedures that assume fixed physical coefficients, and that incorporate the price changes of the produced and nonproduced goods using their respective prices indices.

D.2 Using Price Indexes

To present the general procedure, we will start with a simple example. We assume that from the data corresponding to the old relative price system, but incorporating the changes in the tradeability of the products resulting from the relative prices changes, we have prepared the system of intersectoral relations described by the equations

$$\Sigma_i \, q_{ij} \, p_i + \Sigma_h \, u_{hj} \, p_h = p_j$$

where q_{ij} is the quantity of the produced input i needed to produce one unit of product j, p_i (p_j) is the price of the input (product) i (j), u_{hj} is the quantity of the nonproduced input j, and p_h is the respective price. The system of corresponding value coefficients will be

$$\Sigma_i \, a_{ij} + \Sigma_h \, f_{hj} = 1$$

were the matrix coefficients are

$$a_{ij} = q_{ij}\frac{p_i}{p_j} \qquad f_{hj} = u_{hj}\frac{p_h}{p_j}$$

The change in the relative prices of the nonproduced inputs means that instead of having the old prices pi and ph, we will have new ones pin and phn. *If these price changes have not affected the physical coefficients,* q_{ij} and u_{hj},[3] the new system will be

$$\Sigma_i\, q_{ij}\, p_i^n + \Sigma_h\, u_{hj}\, p_h^n = p_j^n \qquad [D.1]$$

where p_j^n are the new prices of the produced goods. Starting from the new price system [D.1], we can write another system dividing each equation by the old product price (p_j) and multiplying and dividing each input coefficient q_{ij} and u_{hj} by its respective old price (p_i and p_h). Thus we obtain the system

$$\Sigma_i\, q_{ij}\,\frac{p_i}{p_j}\,\frac{p_i^n}{p_i} + \Sigma_h\, u_{hj}\,\frac{p_h}{p_j}\,\frac{p_h^n}{p_h} = \frac{p_j^n}{p_j} \qquad [D.2]$$

that we will call price ratios. The system [D.2] of price ratios presents two important characteristics. The first is that we can express the system of price ratios as a function of the old value coefficients, or

$$\Sigma_i\, a_{ij}\, z_i + \Sigma_h\, f_{hj}\, z_h = z_j \qquad [D.3]$$

where z_i, z_j and z_h are the ratios of the new to the old prices. The second is that it has *the same value coefficients* as the new price system. In effect,

$$q_{ij}\,\frac{p_i}{p_j}\,\frac{p_i^n}{p_i}\,\frac{p_j}{p_j^n} = q_{ij}\,\frac{p_i^n}{p_j^n} = a_{ij}^n$$

[3] Remember that the changes in tradeability of the products have already been incorporated.

$$\sum_h u_{hj} \frac{p_h}{p_j} \frac{p_h^n}{p_h} \frac{p_j}{p_j^n} = u_{hj} \frac{p_h^n}{p_j^n} = f_{hj}^n$$

Therefore, using a 3x3 example to illustrate the matrix algebra involved, we can transpose matrices **A** y **F**, and write the system of price ratios as a function of the old price system, obtaining

$$\begin{bmatrix} a11 & a21 & a31 \\ a12 & a22 & a32 \\ a13 & a23 & a33 \end{bmatrix} \begin{bmatrix} z1 & 0 & 0 \\ 0 & z2 & 0 \\ 0 & 0 & z3 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + \begin{bmatrix} f11 & f21 \\ f12 & f22 \\ f13 & f23 \end{bmatrix} \begin{bmatrix} z1 & 0 \\ 0 & z2 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} z1 & 0 & 0 \\ 0 & z2 & 0 \\ 0 & 0 & z3 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

or simply

$$\mathbf{A}' \hat{\mathbf{z}} [1] + \mathbf{F}' \hat{\mathbf{z}}_h [1] = \hat{\mathbf{z}} [1] \qquad [D.4]$$

where **A'** and **F'** are the transaction and the nonproduced inputs matrices, transposed, $\hat{\mathbf{z}}$ and $\hat{\mathbf{z}}_h$ are diagonal matrices (indicated by the circumflex accent), and [1] is a column vector of ones. To calculate the system's coefficients, we premultiply by $\hat{\mathbf{z}}^{-1}$. Thus we obtain

$$\hat{\mathbf{z}}^{-1} \mathbf{A}' \hat{\mathbf{z}} [1] + \hat{\mathbf{z}}^{-1} \mathbf{F}' \hat{\mathbf{z}}_h [1] = [1] \qquad [D.5]$$

We have already mentioned that this is the system of coefficients corresponding to the new prices and, therefore, for given physical coefficients, we can obtain it using the matrix of value coefficients corresponding to the old coefficients and indices of the ratios between the new and the old prices for the produced (vector **z**) and the nonproduced inputs (vector $\mathbf{z}_h$) through the formulas[4]

$$\mathbf{A}^{n\prime} = [a_{ji}^n] = \hat{\mathbf{z}}^{-1} \mathbf{A}' \hat{\mathbf{z}} \qquad [D.6]$$

---

[4] Bulmer-Thomas (1982) and United Nations (1973) provide a more detailed discussion of this type of procedure.

$$\mathbf{F}^{n\prime} = [f^{n}_{jh}] = \hat{\mathbf{z}}^{-1}\,\mathbf{F}'\,\hat{\mathbf{z}}_{h} \qquad [D.7]$$

Starting from the new coefficients matrices, we can now calculate our new matrix of total requirements of nonproduced inputs and transfer payments by

$$\mathbf{G}^{n\prime} = (\mathbf{I} - \mathbf{A}^{n\prime})^{-1}\,\mathbf{F}^{n\prime} \qquad [D.8]$$

It can be shown[5] that, replacing $\mathbf{A}^{n\prime}$ and $\mathbf{F}^{n\prime}$ in [D.8] by their expressions [D.6] and [D.7], the new matrix of total requirements of nonproduced inputs and transfer payments can also be directly calculated as

$$\mathbf{G}^{n\prime} = \hat{\mathbf{z}}^{-1}\,(\mathbf{I} - \mathbf{A}')^{-1}\,\mathbf{F}'\,\hat{\mathbf{z}}_{h} \qquad [D.9]$$

It is important to point out that even though the physical coefficients have not been affected by the price changes $\mathbf{z}$ and $\mathbf{z}_h$, the price updating will still involve levels of error that are difficult to avoid. In the first place, the preparation of *ad hoc* matrices generally does not include the construction of a matrix for traded inputs. Instead, the traded inputs are decomposed into foreign exchange, taxes and other costs, and the totals for foreign exchange and taxes are assigned to two lines in matrix **F**. This simplified method makes it difficult to incorporate the changes in international prices. In effect, the value coefficient of the foreign exchange will be

$$f_{dj} = (\Sigma_k\, u_{djk}\, p_{k\,e}) \,/\, p_j$$

in which u_{djk} is the physical coefficient of the traded input k, p_k its foreign exchange price and e the exchange rate. Therefore, the assumption of fixed physical coefficients requires having an index for the international prices of the traded inputs of sector j and one for the exchange rate, and we will be left with the problem of the "currency basket".

A second aspect to which we referred are the effects of the changes on the tradeability of the products. If a nontraded input on the original matrix becomes imported, the corresponding change will be incorporated in the original coefficients matrix starting from its domestic price at

---

[5] See the note at the end of this Appendix.

that moment. Then, when updating, it will be affected only by the exchange rate and international prices, without considering the effect of the difference between the domestic prices of the domestically produced input and the imported one.

Finally, a third problem in the application of the method just described is that in many cases, the required price indexes are not available; in particular, you will not have the data to prepare vector **z**, which leads us to discussion of a second procedure: that used by the AJUSTEG program.[6]

D.3 Updating with Endogenous Price Indexes of Produced Goods

If the price indexes for the produced goods are not available, it is possible to use the input-product price model to calculate them as a function of the price indexes of the nonproduced inputs, and later use them to update the matrix. Recall from [D.4] that we can present the system of equations that determines the price relations as

$$\mathbf{A}'\hat{\mathbf{z}}[1] + \mathbf{F}'\hat{\mathbf{z}}_h[1] = \hat{\mathbf{z}}[1] \qquad [D.4]$$

and thus write the vector **z** of the price relations of produced goods as

$$\mathbf{z} = (\mathbf{I} - \mathbf{A}')^{-1}\mathbf{F}'\mathbf{z}_h \qquad [D.10]$$

Once vector **z** is calculated, we replace it in the system of price ratios [D.4] premultiply by $\hat{\mathbf{z}}^{-1}$, and so obtain the updated value coefficient matrices as

$$\hat{\mathbf{z}}^{-1}\mathbf{A}'\hat{\mathbf{z}}[1] + \hat{\mathbf{z}}^{-1}\mathbf{F}'\hat{\mathbf{z}}_h[1] = [1]$$

where

$$\mathbf{A}^{n\prime} = \hat{\mathbf{z}}^{-1}\mathbf{A}'\hat{\mathbf{z}}$$

---

[6] Londero (1992) proposes and exemplifies an approach for the case when some of the z_i are available.

$$\mathbf{F}^{n\prime} = \hat{\mathbf{z}}^{-1}\,\mathbf{F}'\,\hat{\mathbf{z}}_h$$

result from using the vector **z** calculated using the input-output price model. Then, as in the previous case, we can calculate the matrix $\mathbf{G}^{n\prime}$ of total requirements of nonproduced inputs corresponding to the new price system starting from the expression [D.8]

$$\mathbf{G}^{n\prime} = (\mathbf{I} - \mathbf{A}^{n\prime})^{-1}\,\mathbf{F}^{n\prime} \qquad [D.8]$$

or, replacing [D.6] and [D.7] in [D.8], directly from the coefficients matrices corresponding to the old prices as

$$\mathbf{G}^{n\prime} = \hat{\mathbf{z}}^{-1}\,(\mathbf{I} - \mathbf{A}')^{-1}\,\mathbf{F}'\,\hat{\mathbf{z}}_h \qquad [D.9]$$

Expressions [D.9] and [D.10] are the ones used by AJUSTEG to calculate the new type G matrix when all of the coefficients z_h are exogenous, but without transposing it, i.e. as $\mathbf{G}^n = [g^n_{hj}]$.

Therefore, according to what we have presented in this section, the procedure for correcting a matrix of total requirements of nonproduced inputs when vector **z** is endogenous, must follow this sequence:

i) estimate the price relations between the nonproduced inputs z_h and order them as column vector $\mathbf{z}_h$;

ii) calculate the ratios between the prices of the produced goods z_j as $\mathbf{z} = (\mathbf{I} - \mathbf{A}')^{-1}\,\mathbf{F}'\,\mathbf{z}_h$; and

iii) calculate the total requirements of nonproduced inputs corresponding to the new prices as $\mathbf{G}^n = \hat{\mathbf{z}}^{-1}\,(\mathbf{I} - \mathbf{A}')^{-1}\,\mathbf{F}'\hat{\mathbf{z}}_h$, where the vectors **z** and $\mathbf{z}_h$ have now been arranged as diagonal matrices.

The AJUSTEG program carries out the operations described in ii) and iii).

Finally, it is logical to point out that the calculation of vector **z** by this process will always be an approximation and the margin of error will depend on the precision with which the input-output price model [D.10] captures the price determination mechanism.

D.4 Nonproduced Goods or Transfer Payments with Price Indexes Endogenous to the System

We now consider the case of the sales tax and profit margins calculated over the price. To avoid complicating the notation and the interpretation of the calculations, it is convenient to start only with the sales tax. The system of the price ratios when a sales tax exists can be presented in matrix form as

$$(\mathbf{A}' + \hat{\mathbf{T}})\,\mathbf{z} + \mathbf{V}'\,\mathbf{z}_h'' = \mathbf{z} \qquad [D.11]$$

in which $\hat{\mathbf{T}}$ is the diagonal matrix of the coefficients t_j (i. e., the sales tax rate), $\mathbf{V}'$ is the matrix $\mathbf{F}'$ without the column corresponding to the taxes t_j and $\mathbf{z}_h''$ the vector of the z_h excluding the corresponding to t_j. If the price ratios vector of the produced goods ($\mathbf{z}$) is not available, it can be calculated from [D.11] as

$$\mathbf{z} = (\mathbf{I} - \mathbf{A}' - \hat{\mathbf{T}})^{-1}\,\mathbf{V}'\,\mathbf{z}_h'' \qquad [D.12]$$

Then, knowing vector $\mathbf{z}$, we can write the price relations system [D.11] as

$$\mathbf{A}'\,\hat{\mathbf{z}}\,[1] + \mathbf{V}'\,\mathbf{z}_h''\,[1] + \hat{\mathbf{T}}\,\hat{\mathbf{z}}\,[1] = \hat{\mathbf{z}}\,[1]$$

and obtain the coefficient matrices corresponding to the new prices from those of the old prices premultiplying by $\hat{\mathbf{z}}^{-1}$

$$\begin{aligned} &\hat{\mathbf{z}}^{-1}\,\mathbf{A}'\,\hat{\mathbf{z}}\,[1] + \hat{\mathbf{z}}^{-1}\,\mathbf{V}'\,\hat{\mathbf{z}}_h''\,[1] + \hat{\mathbf{z}}^{-1}\,\hat{\mathbf{T}}\,\hat{\mathbf{z}}\,[1] = [1] \\ &\mathbf{A}^{n\prime}\,[1] \quad + \quad \mathbf{V}^{n\prime}\,[1] \quad + \quad \hat{\mathbf{T}}^{n}\,[1] \quad = [1] \end{aligned} \qquad [D.13]$$

From the same matrices, we can calculate the total requirements of nonproduced inputs corresponding to the new prices as

$$\mathbf{W}^{n\prime} = (\mathbf{I} - \mathbf{A}^{n\prime})^{-1}\,\mathbf{V}^{n\prime} \qquad [D.14]$$

$$\mathbf{t}^{n\prime} = (\mathbf{I} - \mathbf{A}^{n\prime})^{-1}\,\hat{\mathbf{T}}^{n}\,[1] \qquad [D.15]$$

where $\mathbf{W}^{n\prime}$ is matrix $\mathbf{G}^{n\prime}$ excluding the taxes column and $\mathbf{t}^{n\prime}$ is the vector of total tax requirements. Replacing $\mathbf{A}^{n\prime}$, $\mathbf{V}^{n\prime}$ and $\hat{\mathbf{T}}^{n}$ in [D.14] and [D.15] by their expressions in [D.13] as a function of coefficient matrices corresponding to old prices, it can be demonstrated that

$$\hat{\mathbf{z}}^{-1} (\mathbf{I} - \mathbf{A}')^{-1} \mathbf{V}' \hat{\mathbf{z}}''_{h} [1] + \hat{\mathbf{z}}^{-1} (\mathbf{I} - \mathbf{A}')^{-1} \hat{\mathbf{T}} \hat{\mathbf{z}} [1] = [1] \qquad [D.16]$$

We can now interpret each one of the terms in the previous expression. In the first, the product

$$(\mathbf{I} - \mathbf{A}')^{-1} \mathbf{V}' = [r_{ji}] [v_{hj}, h \neq \text{sales tax}] = [w_{ih}, h \neq \text{sales tax}]$$

is the matrix of total requirements of nonproduced inputs and transfer payments corresponding to the old prices, excluding the sales tax. Therefore,

$$\mathbf{W}^{n\prime} = \hat{\mathbf{z}}^{-1} (\mathbf{I} - \mathbf{A}')^{-1} \mathbf{V}' \hat{\mathbf{z}}''_{h} = [w_{jh} z_h \neq z_j] = [w^{n}_{jh}, h \neq \text{sales tax}]$$

corrects the total requirements of nonproduced input h by the ratio of the new to the old prices z_h, and then calculates the coefficients again dividing by z_j. This will be the matrix of total requirements of nonproduced inputs and transfer payments corresponding to the new prices $\mathbf{W}^{n\prime} = [w^{n}_{jh}]$, for h different from the sales tax. The second term in [D.16] is more complicated. The total sales tax requirements depend on the total requirements of the produced inputs; these prices will change in z_i. Thus, the second term will be

$$\hat{\mathbf{z}}^{-1} (\mathbf{I} - \mathbf{A}')^{-1} \hat{\mathbf{T}} \hat{\mathbf{z}} [1] = [r_{ji} t_{ii} z_i / z_j] [1]$$

which can be interpreted in the following manner. The product $r_{ji} t_{ii}$ are the total tax requirements for product j attributable to the total production of input i necessary per unit of j. Next, this product is corrected by the ratio between the prices of i and, finally, the coefficients are recalculated dividing by z_j. Upon multiplying the matrix resulting from the vector [1], we obtain the vector

$$\mathbf{t}^{n\prime} = \hat{\mathbf{z}}^{-1} (\mathbf{I} - \mathbf{A}')^{-1} \hat{\mathbf{T}} \hat{\mathbf{z}} [1] = [r_{ji} t_{ii} z_i / z_j] [1] = [\Sigma_i r_{ji} t_{ii} z_i / z_j]$$

that contains the new total requirements of sales tax for sector *j*, or vector $\mathbf{t}^n$. Now, we can reconstruct the complete matrix of nonproduced inputs and transfer payments as

$$\mathbf{G}^{n\prime} = [\mathbf{W}^{n\prime} \mid \mathbf{t}^{n\prime}]$$

In the case of a devaluation, the preceding presentation assumes that tariffs are not modified. However, the devaluation may be accompanied by, for example:

a. a reduction in the import taxes if the objective is to "liberalize" trade; or

b. increase in the export taxes for agricultural products if the object is to stimulate industrial exports without affecting the domestic prices of the agricultural products and their derivatives while at the same time collecting the additional revenue generated by the devaluation.

We will consider this case with our example from sector 2. Let

$$a_{12} + a_{22} + a_{32} + d_2 + t_2^m + t_2^x + t_2 + w_2 = 1$$

be the cost structure corresponding to the old prices to which we have incorporated an export tax t_2^x. Thus, the effect of the devaluation over the final price will be

$$a_{12}\, z_1 + a_{22}\, z_2 + a_{32}\, z_3 + d_2\, z_d + t_2^m\, z_t^m + t_2^x\, z_t^x + t_2\, z_2 + w_2\, z_w = z_2$$

which we can solve using the procedure already shown. However, we should point out that the correction will be possible if:

a. the size of *all* of the *ad valorem* import or export taxes on inputs vary in approximately the same proportion;

b. import taxes have been distinguished from export taxes and the latter from the respective subsidies; and

c. the specific taxes have been distinguished from the *ad valorem* taxes.

This makes it necessary to avoid the aggregation in the matrix of direct requirements of nonproduced inputs **F** so that we do not find ourselves affected later by self-imposed limitations.

We will now consider the case of the profit margins. If these are calculated as a fixed

proportion of the price of the product, then they are treated in the same way as the sales tax. If we call the diagonal matrix of coefficients of profit per gross value of production B, the price ratio replacing [D.10] will be

$$(\mathbf{A}' + \hat{\mathbf{T}} + \hat{\mathbf{B}})\,\mathbf{z} + \mathbf{V}'\hat{\mathbf{z}}''_h = \mathbf{z} \qquad [D.17]$$

in which $\mathbf{V}'$ is now the matrix $\mathbf{F}'$ excluding the columns corresponding to the sales taxes and the profit margins, and $\hat{\mathbf{z}}''_h$ is the vector of the z_h excluding those corresponding to the mentioned columns. If the price indexes for the produced goods were not available, vector $\mathbf{z}$ can be calculated from the preceding equation as

$$\mathbf{z} = (\mathbf{I} - \mathbf{A}' - \hat{\mathbf{T}} - \hat{\mathbf{B}})^{-1}\,\mathbf{V}'\hat{\mathbf{z}}''_h \qquad [D.18]$$

Next, knowing vector $\mathbf{z}$ the system can be expressed as

$$\mathbf{A}'\hat{\mathbf{z}}[1] + \mathbf{V}'\hat{\mathbf{z}}''_h[1] + \hat{\mathbf{T}}\,\hat{\mathbf{z}}[1] + \hat{\mathbf{B}}\,\hat{\mathbf{z}}[1] = \hat{\mathbf{z}}[1]$$

Premultiplying by $\hat{\mathbf{z}}^{-1}$ we now obtain the value coefficients corresponding to the new prices as a function of the coefficient matrices valued at the old prices, or

$$\hat{\mathbf{z}}^{-1}\mathbf{A}'\hat{\mathbf{z}}[1] + \hat{\mathbf{z}}^{-1}\mathbf{V}'\hat{\mathbf{z}}''_h[1] + \hat{\mathbf{z}}^{-1}\hat{\mathbf{T}}\,\hat{\mathbf{z}}[1] + \hat{\mathbf{z}}^{-1}\hat{\mathbf{B}}\,\hat{\mathbf{z}}[1] = \hat{\mathbf{z}}[1] \qquad [D.19]$$

$$\mathbf{A}^{n\prime}[1] + \mathbf{V}^{n\prime}[1] + \hat{\mathbf{T}}^{n}[1] + \hat{\mathbf{B}}^{n}[1] = [1]$$

From here, we will be able to calculate, as in the preceding cases, the total requirements of nonproduced inputs as

$$\mathbf{W}^{n\prime} = (\mathbf{I} - \mathbf{A}^{n\prime})^{-1}\,\mathbf{V}^{n\prime}$$

$$\mathbf{t}^{n\prime} = (\mathbf{I} - \mathbf{A}^{n\prime})^{-1}\,\hat{\mathbf{T}}^{n}[1]$$

$$\mathbf{b}^{n\prime} = (\mathbf{I} - \mathbf{A}^{n\prime})^{-1}\,\hat{\mathbf{B}}^{n}[1]$$

We now replace $\mathbf{A}^{n\prime}$, $\mathbf{V}^{n\prime}$, $\hat{\mathbf{T}}^{n}$ and $\hat{\mathbf{B}}^{n}$ by their expressions in [D.19] as a function of the coefficient

matrices corresponding to the old prices and obtain

$$\hat{\mathbf{z}}^{-1} (\mathbf{I} - \mathbf{A}')^{-1} \mathbf{V}' \hat{\mathbf{z}}_h'' [1] + \hat{\mathbf{z}}^{-1} (\mathbf{I} - \mathbf{A}')^{-1} \hat{\mathbf{T}} \hat{\mathbf{z}} [1] +$$

$$\hat{\mathbf{z}}^{-1} (\mathbf{I} - \mathbf{A}')^{-1} \hat{\mathbf{B}} \hat{\mathbf{z}} [1] = [1] \qquad [D.16]$$

where

$$\mathbf{W}^{n\prime} = \hat{\mathbf{z}}^{-1} (\mathbf{I} - \mathbf{A}')^{-1} \mathbf{V}' \hat{\mathbf{z}}_h''$$

$$\mathbf{t}^{n\prime} = \hat{\mathbf{z}}^{-1} (\mathbf{I} - \mathbf{A}')^{-1} \hat{\mathbf{T}} \hat{\mathbf{z}} [1]$$

$$\mathbf{b}^{n\prime} = \hat{\mathbf{z}}^{-1} (\mathbf{I} - \mathbf{A}')^{-1} \hat{\mathbf{B}} \hat{\mathbf{z}} [1]$$

whose interpretation is the same as in the case of the sales taxes. Finally,

$$\mathbf{G}^{n\prime} = [\mathbf{W}^{n\prime} \mid \mathbf{t}^{n\prime} \mid \mathbf{b}^{n\prime}]$$

will be the new matrix of total requirements of nonproduced inputs obtained from the matrices corresponding to the old prices. The AJUSTEG program uses equations [D.18] and [D.20], but presents the results in transposed form,

$$\mathbf{G}^n = [g_{hj}^n] = \left[\begin{array}{c} \mathbf{W}^n \\ \hline \mathbf{t}^n \\ \hline \mathbf{b}^n \end{array}\right]$$

Note to Appendix D

Replacing [D.6] and [D.7] in [D.8] we obtain

$$\mathbf{G}^{n\prime} = [\mathbf{I} - \hat{\mathbf{z}}^{-1} \mathbf{A}' \hat{\mathbf{z}}]^{-1} \hat{\mathbf{z}}^{-1} \mathbf{F}' \hat{\mathbf{z}}_h \qquad [D.21]$$

Now, we define

$$\mathbf{C}^{-1} = [\mathbf{I} - \hat{\mathbf{z}}^{-1}\,\mathbf{A}'\,\hat{\mathbf{z}}]^{-1} \qquad [D.22]$$

that, replaced in [D.21] allows us to obtain

$$\mathbf{G}^{n\prime} = \mathbf{C}^{-1}\,\hat{\mathbf{z}}^{-1}\,\mathbf{F}'\,\hat{\mathbf{z}}_h \qquad [D.23]$$

Recalling that $\mathbf{C}^{-1}\,\hat{\mathbf{z}}^{-1} = (\hat{\mathbf{z}}\,\mathbf{C})^{-1}$ we can write

$$\begin{aligned}\mathbf{C}^{-1}\,\hat{\mathbf{z}}^{-1} = (\hat{\mathbf{z}}\,\mathbf{C})^{-1} &= [\hat{\mathbf{z}} - \hat{\mathbf{z}}\,\hat{\mathbf{z}}^{-1}\,\mathbf{A}'\,\hat{\mathbf{z}}]^{-1} \\ &= [\hat{\mathbf{z}} - \mathbf{A}'\,\hat{\mathbf{z}}]^{-1} \\ &= \hat{\mathbf{z}}^{-1}\,(\mathbf{I} - \mathbf{A}')^{-1} \qquad [D.24]\end{aligned}$$

Replacing [D.24] in [D.23] we arrive at

$$\mathbf{G}^{n\prime} = \hat{\mathbf{z}}^{-1}\,(\mathbf{I} - \mathbf{A}')^{-1}\,\mathbf{F}'\,\hat{\mathbf{z}}_h.$$

which is expression [D.9].

REFERENCES

Allen, R. (1974), "Some Experiments with the RAS Method of Input-Output Coefficients", *Bulletin of the Oxford University Institute of Economics and Statistics*, Vol. 36, No.3, pp. 215-228.

Bates, J. and Bacharach, M. (1963), *Input-Output Relations 1954-1966*, Vol. 3 of the series *A Programme for Growth*, Cambridge University Department of Applied Economics, Chapman and Hall, London.

BID-NAFINSA (1986), *Los precios de cuenta en México*, NAFINSA, México, D.F.

Bulmer-Thomas, V. (1982), *Input-Output Analysis in Developing Countries*, John Wiley, New York.

Flament, M. (1987), "Estimación de precios para el Uruguay", mimeo, Inter-American development Bank, Washington DC., and Presidencia de la República, Montevideo.

Lecomber, J. (1975), "A Critique of Methods of Adjusting, Updating and Projecting Matrices", in R. Allen and W. Gossling, *Estimating and Projecting Input-Output Coefficients*, Input-Output Publishing Co., London.

Londero, E. (1987), *Benefits and Beneficiaries. An Introduction to Estimating Distributional Effects in Cost-Benefit Analysis*, Inter-American Development Bank, Washington, DC.

Londero, E. (1989), "Sobre el uso de técnicas insumo-producto para la estimación de precios de cuenta", *Desarrollo y Sociedad*, No. 24, pp. 131-158.

Londero, E. ed. (1992), *Precios de cuenta. Principios, metodología y estudios de caso*, Inter-American Development Bank, Washington, DC.

Lucking, R. (1993), "Technical Problems in the Appraisal of Projects Using the Semi-Input-Output Methdology", *Project Appraisal*, Vol. 8, No. 2, pp. 113-23.

Lynch, R. (1979), "An Assessment of the RAS Method for Updating Input-Output Tables", in *Proceedings of the Seventh International Conference on Input-Output Techniques*, UNIDO, United Nations, New York.

United Nations (1973), *Input-Output Tables and Analysis*, Series F, No. 14, Rev.1, United Nations, New York.

Powers, T. ed. (1981), *Estimating Accounting Prices for Project Appraisal*, Inter-American Development Bank, Washington, DC.

Scott, M., Macarthur, J. and Newbery, D. (1976), *Project Appraisal in Practice*, Heinemann, London.

* * *