The Theoretical Structure of the MONASH Model represented in TABLO Language

14. Introduction

MONASH computations are performed by the GEMPACK programs.[1] Central to these programs is TABLO which, among other things, reads representations of models written in TABLO language. The main part in this chapter (section 18) is the TABLO representation of MONASH. This representation is not only the means by which we communicate the MONASH theory to computers, but it is close enough to ordinary algebra to be useable as the principal means for communicating this theory to humans. Supplemented by the detailed commentaries in Chapter 5, the TABLO code provides a complete, accurate and comprehensible account of the structure of the MONASH model.

We see little virtue in including in this book both a TABLO representation of MONASH and a representation in ordinary algebra.[2] By using the TABLO representation in our explanation of the model, we avoid imposing on MONASH users the task of translating from one set of notation (ordinary algebra) to another set (TABLO). This will reduce the effort required from users to understand and apply the model. It will also reduce the likelihood of inconsistencies between our discussion of the model's theory and its computer implementation. A final advantage of our strategy is that it will allow readers to acquire a working knowledge of TABLO as a low cost by-product of their work in understanding the MONASH model.

Although close to ordinary algebra, the TABLO language has a particular structure and some unavoidable idiosyncrasies in its vocabulary and syntax. A complete explanation is given in the GEMPACK manuals (Harrison and Pearson, 1994 and 1998a, b and c). With a view to keeping the present book self-contained, this chapter provides brief background material on the GEMPACK programs. Section 15 places TABLO in the context of the suite of GEMPACK programs used in MONASH computations. Section 16 contains notes on TABLO vocabulary and syntax and on non-compulsory conventions that we have adopted with the aim of improving the human readability of the TABLO code. Section 17 is an overview of the structure of the TABLO representation of MONASH given in section 18.

15. Overview of the GEMPACK computations for the MONASH model

As explained in section 11, MONASH computations are performed as a sequence of solutions of linear systems of the form

---

[1] For a comprehensive description of GEMPACK and details of how to obtain it, see Harrison and Pearson (1996).

[2] An earlier presentation of a detailed CGE model in TABLO language is Horridge *et al.* (1993).

$$A(\overline{V})v = 0 \tag{15.1}$$

where

A ($\overline{V}$) is a matrix of coefficients (e.g., cost and sales shares) evaluated at a solution $\overline{V}$ of the model, and v is the vector of deviations in the model's variables away from $\overline{V}$.

The sequence of solutions may be single- or multi-step computations for a single year, or for a series of years. At each step in the sequence, the A matrix is evaluated at a different vector $\overline{V}$. As suggested in section 12, the vector ($\overline{V}(q)$) used in the qth step (q>1) is usually derived substantially from the solution of (15.1) obtained in the (q-1)th step and the vector ($\overline{V}(1)$) is derived from data.

Figure 15.1 is a flow diagram showing how GEMPACK creates and solves the sequence of linked equation systems (15.1).

The GEMPACK program with which we are mainly concerned is TABLO. This creates an executable image of MONASH, MONASH.EXE, by operating on the TABLO representation given in section 18. TABLO also processes condensation instructions aimed at reducing the model to a manageable size. As presented in section 18, MONASH contains about 1.3 million equations and 1.8 million variables, giving an A($\overline{V}$) matrix of approximately 1.3m by 1.8m. It is not practical to solve directly equations systems of this size. The problem is made manageable in two ways: by omitting arrays of exogenous variables that are not shocked and by substituting out arrays of endogenous variables that are not of interest. The arrays that are targeted for these treatments typically have large numbers of components.

An example of an array that is usually omitted is

fa1marg(i,s,j,r)   for i∈COM, s∈SOURCE, j∈IND and r∈MARGCOM.

This is an array of technical change shifters concerned with the usage of margin commodity r to facilitate the flow of commodity i from source s to industry j for the purpose of current production. The array is occasionally useful in simulating technical changes but in most applications it is set exogenously on zero. Thus in most applications it can be deleted, allowing the elimination of 223,910 columns (=115 commodities x 2 sources x 113 industries x 9 margin commodities) of the A($\overline{V}$) matrix.

Figure 15.1. GEMPACK solution of MONASH
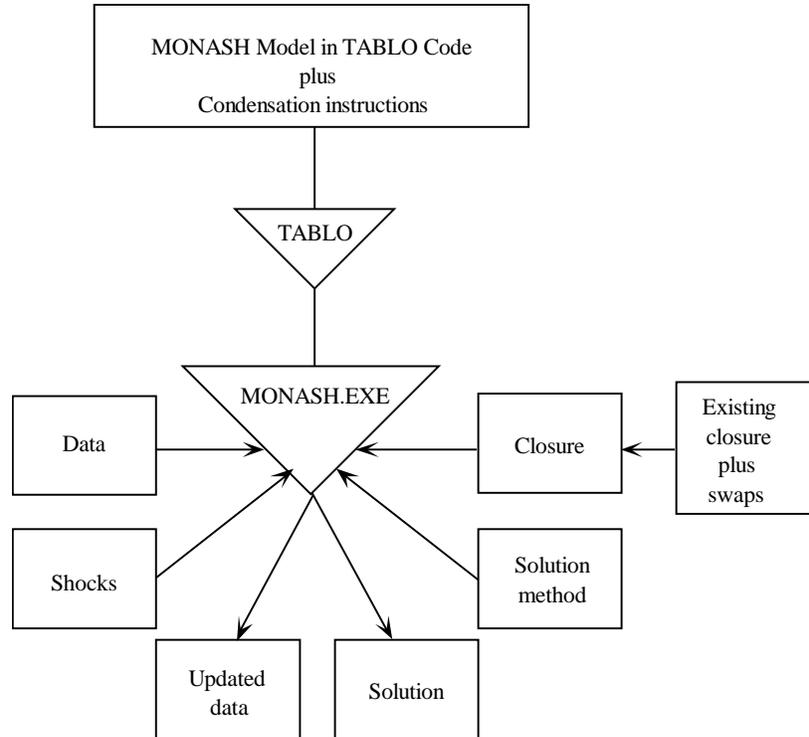
An example of an array that is usually substituted out is the intermediate usage of commodity i from source s by industry j for the purpose of current production, x1csi(i,s,j) for i∈COM, s∈SOURCE, j∈IND. This array contains 25,990 variables (=115 x 2 x 113) and is substituted out using equation E_x1csi in subsection 18.8b. This equation has the form

$$x1csi(i,s,j) = \sum_k a_k(i,s,j) * v_k \qquad i\in COM,\ s\in SOURCE,\ j\in IND \qquad (15.2)$$

where the a_ks are coefficients (components of A matrix) and the v_ks are variables (deviations in prices, activity levels and technical change variables). Following a condensation instruction, TABLO will create an A matrix reflecting an equation system in which E_x1csi does not appear and x1csi(i,s,j) has been replaced by the RHS of (15.2). Both the row and the column dimensions of this A matrix are 25,990 less than those of the A matrix for the original equation system. Although x1csi(i,s,j) has been eliminated, we can still obtain solutions for it. TABLO can give instructions that generate solutions for x1csi(i,s,j) by backsolving, that is by using the RHS of (15.2).

As indicated in Figure 15.1, MONASH.EXE calls for: data to be used in the evaluation of the coefficients in the A matrix; a specification of the closure which can be derived from a pre-existing closure by a series of swap statements interchanging endogenous and exogenous variables; a specification of the solution method, e.g. Johansen/Euler with 4 steps in each year; and the values of the shocks to be given to the exogenous variables. MONASH.EXE then produces the solution showing the effects on the endogenous variables of the shocks to the exogenous variables. It also produces updated data files. These have identical format to the data input files and reflect the post-solution situation. For example, the input-output flows in the updated file are those from the input file altered by the changes in prices and quantities which form part of the solution. The updated files from the (q-1)th step in a sequence of solutions of (15.1) normally become the input data for the qth step.

16. *Notes on TABLO syntax and vocabulary, and on conventions observed in the TABLO representation of MONASH*

The following list gives an incomplete coverage of the TABLO syntax and vocabulary and of our conventions. Many other points relevant to these topics will emerge in the rest of this chapter and in the next. The points we have chosen to list here will be helpful to readers when they are following the examples in section 17 and looking for the first time at the TABLO code in section 18.

- Material in the TABLO code enclosed by exclamation marks (!) is not processed by TABLO. It merely provides explanatory comments for human readers.
- Material enclosed by cross-hatches (#) does not play a role in computations. However, it is recorded by TABLO and used in the labelling of various GEMPACK printouts.
- All TABLO statements end with a semicolon(;).
- Among the key words in the TABLO language are: File, Set, Coefficient, Read, Formula ,Variable, Update and Equation. As can be seen from section 18, these words need not always be repeated when they are applicable to an unbroken sequence of statements. For example, if we are declaring two variables, a and b, we can write

  Variable   a   # example variable # ;
  Variable   b   # another example variable # ;   .

  Alternatively we can write

  Variable
  a # example variable # ;
  b # another example variable # ;   .

- TABLO includes a device which is sometimes convenient for avoiding zero-divide difficulties. This is illustrated by the following TABLO code:

  Zerodivide Default 1.0 ;

Formula
A = B/C ;
E = F/G ;
Zerodivide off ;   .

If C happens to be zero, then A is evaluated as 1.0. Similarly, if a division by zero occurs on the RHS of any other formula listed before the command Zerodivide off, then the LHS of the formula is evaluated as 1.0. Thus if G is zero then E is 1.0. Another approach used in TABLO code to deal with potential divisions by zero is to add the coefficient TINY to denominators. TINY is set at a very small number, 10^{-12}.

- TABLO does not distinguish between upper- and lower-case letters. To enhance the human readability of the TABLO representation of MONASH, we have used lower-case letters for variable names and upper-case letters for the names of coefficients, sets and files. For TABLO key words, we have capitalized the first letter, e.g., Sum, Equation, Update, etc.

- TABLO restricts the lengths of names and of comments contained between cross-hatches. This requires the use of abbreviations in the TABLO code.

- TABLO does not allow Greek letters or subscripts and superscripts. This makes the TABLO notation in section 18 cumbersome for performing algebraic manipulations. In Chapter 5, which contains derivations of the MONASH equations, we often work with shorter notation than that in the TABLO code.

- In naming variables and coefficients, we use: 0 to indicate production; 1 to indicate intermediate usage; 2 to indicate demand for use in capital creation; 3 to indicate demands by households; 4 to indicate exports; 5 to indicate government demands; and 6 to indicate demands for inventories. The letters c, s and i are often used to indicate commodity, source and industry. For example, x1csi refers to intermediate usage distinguished by commodity, source and industry.

17. *Overview of the structure of the TABLO representation of MONASH*

The TABLO code in section 18 is concerned with a single step in the sequence of solutions of (15.1). It specifies rules: for reading the data input (such as input-output flows and substitution parameters); for forming the equation system (15.1); and for revising the data in preparation for the next step in the sequence of solutions.

17.1. *Data files*

The code starts in subsection 18.1 by indicating that the data input needed for creating the system (15.1) are drawn from ten files, with logical (i.e., general-purpose) names FID, EXTRA, ..., ROREXT. In implementing the TABLO representation of MONASH we must specify the specific files in our computer to

play the roles of FID, EXTRA, etc. The contents of these specific files are indicated in subsection 18.1 by comments between cross-hatches (#).

Subsection 18.1 also lists a data output text file (New, Text) with the logical name WRITFILE. The specific file which plays the role specified in the TABLO code for WRITFILE collects information useful in checking and explaining results. Implicit in the TABLO code are instructions for the creation of several other output files. These are explained in subsections 17.7 and 17.9 where we discuss update statements, and writes and displays.

There are no restrictions in TABLO governing the number of data files or the distribution of data across them. In choosing to divide the MONASH input data between several files, our main consideration was data independence. If data items X and Y are both in the same input file Z, then we are limited to solutions of (15.1) in which $\overline{V}$ reflects values of X and Y arising from the process which generates the file Z. If this process imposes a link between X and Y which should not apply at each step in the sequence of solutions of (15.1), then X and Y should be in different files. For example, as is apparent from sections 30 and 44, MONASH computations with forward-looking expectations involve a sequence of solutions of (15.1) in which the values of rates of return used in forming $\overline{V}_q$, q>1,

are derived from the solution in the (q-1)th step. On the other hand, the input-output data used in the qth step are not always generated in the (q-1)th step. Consequently, we place rates of return in a different file from the input-output data, thus allowing the input-output data to be independent of the data for rates of return.

17.2. *Sets and subsets*

The next subsection of the TABLO code first declares in alphabetical order the names of sets and either specifies their contents or indicates where these can be found. For example, SOURCE is a set consisting of two objects, "dom" and "imp"; COM is a set consisting of objects whose names will be found in a section of the SETINFO file labelled "COM "; and COUNTSET is a set consisting of 20 objects named "CS1", "CS2", ..., "CS20". Following the cross-hatched descriptions of the sets readers will notice a comment of the form ! size x(y) !. This indicates that there are x elements in the set in standard applications of MONASH and y in the aggregated version. The aggregated version can be obtained by following instructions at http://www.monash.edu.au/policy/monbook1.htm.

By glancing through the set declarations we gain an immediate impression of the objects with which MONASH is concerned. These include 115 commodities (COM) from two sources (SOURCE) produced by 113 industries (IND). Nine of the commodities (COM_JP) are produced by more than one industry and there are seven industries (IND_JP) producing these multi-industry commodities. MONASH deals with several industry classifications including: INDSAGGTO20, a 20-industry classification; and INDSAGGTO26, a 26-industry classification.

Similarly, the model uses several different classifications of commodities and of employment.

The set declarations are followed by statements indicating that some sets are subsets of others. For example, COM_JP is a subset of COM. Via this subset declaration, TABLO knows that the object "C1Wool" in COM_JP is the same object as "C1Wool" in COM.

17.3. Coefficients

Subsection 18.3 contains coefficient declarations in alphabetical order. The coefficients are the main building blocks in the construction of $A(\overline{V})$. Apart from numerous zeros and ones, and a few other numbers, the components of $A(\overline{V})$ are coefficients or functions of coefficients.

For each coefficient, the TABLO code gives evaluation instructions. Evaluation can either be from a data file via a read statement, or in terms of other coefficients via a formula. As discussed in subsection 17.5, some formulas are implemented in every step of a multi-step computation whereas others are operated only in the first step for each year. For each coefficient, we have indicated in the cross-hatched comment the nature of the evaluation instruction: R for Read; F for Formula operated in every step; and FI for Formula operated only in each year's initial step. The read statements can be found in subsection 18.4, listed alphabetically according to the name of the coefficient. The formulas are in subsection 18.5. As explained in subsection 17.5, the formulas cannot be listed in any simple order. Consequently, we have followed the F and FI notation in the cross-hatched comments in subsection 18.3 with lower-case letters indicating the location within subsection 18.5 of the relevant formula.

For many of the coefficients evaluated via read statements, the TABLO code contains update instructions, i.e., instructions on how the post-solution value of the coefficient should be computed. Where an update instruction is provided, we have included a U in the cross-hatched comments. The update statements are in subsection 18.7 and are discussed in subsection 17.7.

To finish the present subsection, we give a few examples that may be helpful in clarifying the meaning of the coefficient declarations in subsection 18.3.

Example 1

AGGCAP is interpreted in our economic theory as the total payments to capital. It is evaluated as a function of coefficients according to a formula given in subsection 18.5i. Because in each step of a sequence of solutions of (15.1) the value of AGGCAP is obtained via a formula, AGGCAP is not updated. We do not need to create a post-solution value to be stored in readiness for our next solution of (15.1).

Example 2

BAS1(i,s,j) is interpreted in our economic theory as the basic value of the flow of good i from source s for use by industry j in current production. From a TABLO

point of view, BAS1 is a three dimensional array taking values for all triples of the form (i,s,j) where i is in the set COM, s is in the set SOURCE and j is in the set IND. As indicated by the R in the cross-hatched comment, values for BAS1 are obtained from data accessed via a Read statement (given in subsection 18.4). As indicated by the U, post-solution values of BAS1 are computed via an update statement (given in subsection 18.7).

Example 3

SIGMA1(i) is interpreted in our economic theory as the elasticity of substitution between domestic and imported good i for use in current production. In TABLO, SIGMA1 is a vector taking values for all i in the set COM. As indicated by the R in the cross-hatched comment, values for SIGMA1 are obtained from data accessed via a Read statement. No update instructions are given. Import/domestic substitution elasticities are usually held constant throughout a sequence of solutions of (15.1), i.e., they are treated as parameters.

Example 4

INDTOIA20(j) is concerned with the aggregation of the 113 MONASH industries to 20 sectors (*IND*ustries *TO I*ndustries *A*ggregated to *20*). It assigns an integer between 1 and 20 to each j in IND according to which of the 20 sectors j belongs. As indicated by the word Integer in its declaration statement, INDTOIA20(j) takes integer values only. The Integer specification is not compulsory. However, it facilitates conditional statements involving equalities. For example,

+ Sum(j,IND:INDTOIA20(j) = 6, XXX)

causes the addition of XXX when INDTOIA20(j) is 6. If INDTOIA20 is not declared as an integer there is a danger that the value of INDTOIA20(j) for the industries in the 6th sector will be stored in the computer as 5.999...99. In this case it is possible that the instruction that XXX should be added for industries in the 6th sector will be ignored.

17.4. Read statements

As already mentioned, subsection 18.4 contains instructions for the evaluation of coefficients from data files. For example, the instructions for BAS1 indicate that values for this coefficient are obtained from locations in the FID file marked F001 and F002. The command Read, used throughout subsection 18.4 instructs post-TABLO programs to access numerical data. By contrast, the command in subsection 18.2, Read Elements, instructs post-TABLO programs to access alpha-numeric information.

17.5. Formulas

Subsection 18.5 has twenty-five parts, marked a to y. Each contains a group of formulas related to a theme described in notes between exclamation marks.

Formulas cannot be arranged in a simple order, such as alphabetical. This is because no coefficient can appear on the RHS of a formula unless instructions for its evaluation have appeared earlier in the TABLO code. Thus, if coefficient A is to be set equal to coefficient B, and B is to be set equal to 10, we cannot use the alphabetical ordering:

Formula A = B ;

Formula B = 10 ;   .

Instead we must write

Formula B = 10 ;

Formula A = B ;   .

The TABLO code allows a wide range of operators to be used in formula statements. The meaning of most of these is clear. Here we interpret a few of the formulas in subsection 18.5 with the aim of clarifying meanings which may not be immediately obvious.

Example 1

Taken from subsection 18.5k:

Formula (Initial)

$$\text{SUMDELTA} = \text{Sum}(i,\text{COM}, \text{DELTA}(i)\ ); \qquad\qquad (17.1)$$

The word Initial in parentheses indicates that in a multi-step solution for year t, the coefficient SUMDELTA is evaluated via formula (17.1) only in the first step. Provided there are no further instructions in our TABLO code regarding its evaluation, SUMDELTA will retain this first-step value throughout the remaining steps for year t. Thus, for example, if the sequence of solutions of (15.1) consists of four solutions for year 1 and four for year 2, then SUMDELTA will be evaluated via (17.1) in solution 1, held constant through solutions 2, 3 and 4, re-evaluated in solution 5 and held constant through solutions 6, 7 and 8.

The Initial option is particularly valuable in the modelling of processes involving lags. Examples in MONASH can be found in several sections, including 18.5q to 18.5t dealing with capital accumulation and rates of return. With lags, we often need to hold a coefficient in the year t computation at a value reflecting the solution for year t-1. This can be achieved via the Initial option provided the initial solution for year t is the final (required) solution for year t-1 (Figure 12.3).

Apart from the word Initial, the only other aspect of (17.1) requiring comment is the Sum notation used on the RHS. This notation means that DELTA(i) is to be summed over all values of i in COM.

Example 2

Taken from section 18.5a:

$$\text{Formula}\qquad \text{NUMCOM} = 1.0/\text{Sum}(i,\text{COM},1);\quad . \qquad\qquad (17.2)$$

The denominator on the RHS is a summation of ones over all i in COM. If there are 115 objects in COM, NUMCOM has the value 1/115.

Since the value of NUMCOM will be constant throughout any sequence of solutions of (15.1), and certainly throughout the solution for year t, we could replace (17.2) by

  Formula (Initial) NUMCOM = 1.0/Sum(i,COM,1) ; .   (17.3)

The choice between (17.2) and (17.3) is unimportant.

Example 3

Taken from subsection 18.5a:

Formula TINY = 0.000000000001 ; .   (17.4)

TINY, which is permanently set at 10^{-12}, is often used to avoid division by zero or the occurrence of an endogenous variable with a zero coefficient in all equations.

Example 4

Taken from section 18.5c:

Formula (All,i,COM_JP)(All,j,IND_JP)

DENOM(i,j) = Sum (qq,COM_JP:MAKE(qq,j) ne 0,

    Sum(cc,COMPCOM:CCPROD(i,j,cc) ne 0

    and CCPROD(qq,j,cc) ne 0, MAKE(qq,j)) ;   (17.5)

Here we have conditions applying to summations. DENOM(i,j) is a summation, over selected values of qq in COM_JP [those for which MAKE(qq,j) is non-zero] of the expression:

Sum(cc,COMPCOM:CCPROD(i,j,cc) ne 0 and CCPROD(qq,j,cc) ne 0, MAKE(qq,j)).

This expression simplifies to

N(qq,i,j)* MAKE(qq,j)

where N(qq,i,j) is the number of values of cc in COMPCOM for which both CCPROD(i,j,cc) and CCPROD(qq,j,cc) are non-zero. Consequently, the restriction of the first Sum operation on the RHS of (17.5) to values of qq for which MAKE(qq,j) is non-zero does not affect the eventual value of DENOM(i,j). It merely improves computational efficiency by reducing the number of arithmetic operations required on the RHS of (17.5).

While our main purpose in the current example is to provide an illustrative interpretation of some relatively difficult TABLO instructions, readers may find an economic interpretation to be reassuring. As explained in section 20, each industry in the set IND_JP produces composite commodities composed of commodities in the set COM_JP. The coefficient CCPROD(r,j,cc) for any r in COM_JP is non-zero if and only if commodity r is part of industry j's composite commodity cc. Because each r is included in no more than one of j's composite commodities, N(qq,i,j) is one if and only if qq and i are in the same composite commodity for industry j. Otherwise it is zero. Thus, DENOM(i,j) is the summation of MAKE(qq,j) over all values of qq for which commodities qq and i are in the same composite commodity for industry j. With MAKE(qq,j) being the value of commodity qq produced by industry j, we see that DENOM(i,j) is the value of j's output of commodities in the same composite group as commodity i.

Example 5

Taken from subsection 18.5w:

Formula (All,s,COUNTSET)

$$\text{COUNT(s)} = \$POS(s) - 1 \; ; \tag{17.6}$$

TABLO keeps track of the order in which it reads the elements of a set. Consider for example the set COUNTSET. From the specification of the contents of COUNTSET in subsection 18.2, TABLO interprets the first element as CS1, the second as CS2 etc. $POS(s) takes the value of the position of s in the relevant set. Thus in (17.6), $POS("CS1") equals 1, $POS("CS2") equals 2 and $POS("CS20") equals 20. Hence COUNT("CS1") equals 0, COUNT("CS2") equals 1 and COUNT("CS20") equals 19.

17.6.   Variables

Subsection 18.6 contains variable declarations in alphabetical order. These give names and dimensions to the components of v in (15.1), or, equivalently, they give names and widths to collections of columns of $A(\overline{V})$.

  Most of the variables are interpreted in our economic theory as percentage changes. For example, caprev is the percentage change, away from its value in $\overline{V}$, of aggregate payments to capital. However, many variables are changes. For example, del_b is the change in the balance of trade, not the percentage change.

  For variables that are to be interpreted as changes, TABLO requires us to include in their declarations the word Change in parentheses. We have also elected to give change variables names starting with either d_ or del_.

  Failure to distinguish in the variable declarations between change and percentage-change variables will not affect single-step solutions for year t . Difficulties arise, however, in the calculation of results from multi-step solutions. If the (Change) instruction is omitted from the declaration of del_b, for example, then in a two-step calculation, we obtain:

$$\text{del\_b}_{result2} = 100*\{(1 + \text{del\_b}_1/100)(1 + \text{del\_b}_2/100) - 1\} \; ,$$

where $\text{del\_b}_{result2}$ is the result for del_b in the two-step computation and del_b_1 and del_b_2 are the solutions in steps 1 and 2. With del_b declared as a change variable, we obtain the correct two-step result; i.e.,

$$\text{del\_b}_{result2} = \text{del\_b}_1 + \text{del\_b}_2 \; .$$

17.7.   Update statements

Earlier in this section, we saw that MONASH takes data from ten files with logical names FID, EXTRA etc. For each of these data input files, the TABLO code contains implicit (hidden from the GEMPACK user) instructions for the creation of ten post-solution files with logical names: updated FID; updated EXTRA, etc.

The implicit TABLO instructions require that the updated files have structures identical to those of the corresponding input files, i.e., identical header (location) names and data arrangements. The data values will also be identical except where explicit instructions are given for their alteration.

Explicit alteration or update instructions are given by Update statements. For MONASH, these are listed in alphabetical order (according to the name of the coefficient being updated) in subsection 18.7. Here we provide a few illustrative interpretations. Among other things, readers should deduce from these examples the meanings in Update statements of (Change), (Explicit) and the absence of either.

Example 1

Update (Change) (All,i,COM:BAS4(i) ne 0)

$$BAS4(i) = [BAS4(i)/100]*[p0(i, "dom") + x4(i)] \; ; \quad . \tag{17.7}$$

Assume that we are in the qth step of a sequence of solutions of (15.1). Then TABLO requires the coefficients on the RHSs of update statements to be set at the values derived earlier in the qth step via read statements and formulas, i.e., the values used in the formulation of $A(\overline{V}_q)$. We will call these the qth-step values.

TABLO requires the variables on the RHSs of update statements to be set at the values derived (or set) in the qth solution of (15.1). Similarly, we will refer to these variable values as qth-step values.

In our current example, the RHS of (17.7) is the change that should be made to the value of the coefficient BAS4(i), for selected values of i, to derive its post-qth-solution value. The selected values of i are those in the set COM such that the qth-step value of BAS4(i) is non-zero.

For these selected values of i, the post-qth-solution value of BAS4(i) is given by

$$BAS4(i)_{post\_q} = BAS4(i)_q + RHS(17.7)_q \quad . \tag{17.8}$$

Because $BAS4(i)_q$ is read from the header "F007" in the FID file (subsection 18.4), TABLO's implicit instructions mean that $BAS4(i)_{post\_q}$ will be stored in the corresponding position in the header designated "F007" in the updated FID file.

In the economic interpretation of our TABLO code, BAS4(i) is the basic value of exports of good i. This can be expressed as the product of the basic price [P0(i, "dom")] and the exported quantity [X4(i)], that is,

$$BAS4(i) = P0(i, "dom")*X4(i) \quad . \tag{17.9}$$

To a first order approximation, the change in BAS4(i) is given by

$$\Delta BAS4(i) = [BAS4(i)/100]*[p0(i, "dom") + x4(i)] \quad , \tag{17.10}$$

where p0(i, "dom") and x4(i) are the percentage changes in P0(i, "dom") and X4(i). The TABLO code in (17.7) is an instruction for the implementation of (17.10).

Example 2

Update (All,j,IND)   CAPITAL(j) = p1cap(j)*cap_at_t(j) ; .   (17.11)

   The absence of either (Change) or (Explicit) means that (17.11) is in short-hand. It implies that the post-qth-solution value of CAPITAL(j) is given by

$$\text{CAPITAL(j)}_{post\_q} = \text{CAPITAL(j)}_q (1 + \text{p1cap(j)}_q/100 + \text{cap\_at\_t(j)}_q/100) \ .$$

Because the qth-step value of CAPITAL(j) is read from file FID Header "F011" (subsection 18.4), the post-qth-solution value is stored in the corresponding position in updated file FID Header "F011".

   In the economic interpretation of the TABLO code, CAPITAL(j) is the rental on capital in industry j. This is the product of the rental price [P1CAP(j)] and the quantity of capital in industry j [CAP_AT_T(j)], that is

$$\text{CAPITAL(j)} = \text{P1CAP(j)}*\text{CAP\_AT\_T(j)} \ .$$

When the rental price changes by p1cap(j) per cent and the quantity of capital changes by cap_at_t(j) per cent, then, to a first-order approximation, the new value of the rental is given by

$$\text{CAPITAL(j)}_{new} = \text{CAPITAL(j)}_{old}(1 + \text{p1cap(j)}/100 + \text{cap\_at\_t(j)}/100) \ . \quad (17.12)$$

The TABLO code in (17.11) is an instruction for the implementation of (17.12).

Example 3

Update LEV_CPI = xi3 ;   .   (17.13)

   The post-qth-solution value of LEV_CPI is given by

$$\text{LEV\_CPI}_{post\_q} = \text{LEV\_CPI}_q(1+\text{xi3}_q/100) \ . \quad (17.14)$$

Example 4

Update (Explicit)

FRISCH = -100/[100-Sum(i,COM,SS3COM(i)(100+p3(i)-c+q)

   + S3COM(i)*d_gamma(i))]; .   (17.15)

   The post-qth-solution value of FRISCH is given by the RHS of (17.15) evaluated at the qth-step values of the coefficients SS3COM(i) and S3COM(i) and of the variables p3(i), c, q and d_gamma(i). The economic interpretation of FRISCH is discussed in sections 22 and 40.

17.8.   *Equations*

Subsection 18.8 contains the TABLO representation of the MONASH equations. Unlike formulas, equations can be listed in any order. Unlike sets, coefficients, reads, variables and updates, equations cannot be listed in an order which is both mechanical (e.g. alphabetical) and useful. The ordering in subsection 18.8 was chosen to facilitate the exposition in Chapter 5.

TABLO requires that each equation has a name. We use names of the form E_xx...x where xx...x is the name of the variable determined by the equation. In thinking about which variable is determined by an equation we had the decomposition closure in mind. In fact, our naming system helped us to find the decomposition closure by forcing us to think about the role in this closure of each equation. When all equations had been correctly named, we deduced the exogenous set for the decomposition closure by comparing the equation names in subsection 18.8 with the variable names in subsection 18.6. The exogenous variables were those for which there was no corresponding equation.

The idea that each equation determines a particular variable is not precise. In a simultaneous equation system, the value of each endogenous variable is determined by the whole system. Nevertheless, at an informal level, we have little difficulty in associating a particular variable with each equation. For example, the first equation in subsection 18.8 is concerned with the outputs of composite commodities by industries, x0ccom(cc,j) for cc in COMPCOM and j in IND_JP. Consequently, we call this equation E_x0ccom.

There are a few cases in which our equation-naming system runs into difficulties because the determination of an endogenous vector variable is spread over more than one vector equation. For example, the variable a(j), $j \in$ IND, is determined in two vector equations in subsection 18.8j. We have named these equations E_a_JP and E_a_UP. The first equation determines a(j) for $j \in$ IND_JP and the second determines the remaining components of a(j), i.e., a(j) for $j \in$ IND_UP. Another potential difficulty is that some vector variables may be split between the endogenous and exogenous categories. However, in the decomposition closure which underlies our equation naming system, this difficulty does not arise. There are no split vector variables.

By naming the equations, we have assigned names to the rows or blocks of rows of the matrix $A(\overline{V})$ in (15.1). With the columns already named in the variable declarations, and with instructions for the evaluation of the coefficients already given in subsections 18.4 and 18.5, we have now completed the instructions for the formulation and evaluation of $A(\overline{V})$. Consider, for example, the last equation in subsection 18.8b. This instructs post-TABLO programs to include among the rows of $A(\overline{V})$ a block of J rows (J is the number of industries) labelled E_del_p1oct. At the intersection of these rows with the del_p1oct columns, the TABLO instructions mean that $A(\overline{V})$ is to contain a J×J identity matrix; at their intersection with the del_f1oct columns, $A(\overline{V})$ is to contain a JxJ diagonal matrix with each diagonal element being the coefficient -LEV_CPI; and at their intersection with the xi3 column, $A(\overline{V})$ is to contain a J-order vector whose jth element is the coefficient -LEV_CPI*LEV_F1OCT(j)/100. At the intersection of the E_del_p1oct rows with all other columns, $A(\overline{V})$ is to contain zeros.

17.9.   Display and Write statements

Subsection 18.9 contains two lists of coefficients under the headings Display and Write. Values of the first list of coefficients are written to a Display file which is automatically created (no logical name need be specified) and labelled in a user-friendly manner by the GEMPACK programs. The purpose of the Display file is to provide information for checking and interpreting results. The second list of coefficients are written to files with logical names nominated by the model-builder in the TABLO code. For MONASH we have only one Write file with logical name WRITFILE. The labelling in Write files is less user friendly than that in Display files. However Write files are in a suitable form to be processed by other GEMPACK programs.

   In a simulation for year t, the values that appear in Display and Write files are the values of coefficients reflecting the execution of reads and formulas in the first step of the computation up to the point where the Display or Write command occurs. These values are usually those used in forming the first step's $A(\overline{V})$ matrix. An exception occurs when we Display or Write the value of a coefficient and then subsequently alter this value via a formula occurring after the Display or Write command.