KM 15(1984)
pag 132-151

TOWARDS MORE FLEXIBILITY IN SOFTWARE TOOLS FOR ECONOMETRICIANS: AN IMPLEMENTATION

Drs. V.J. de Jong, University of Groningen*)

Abstract

In this paper it is investigated, whether the gap between theory and practice in econometrics can be decreased by using more flexible software. Nowadays two categories of software tools for econometricians can be distinguished: software packages and programming languages. Neither of these tools allow econometricians to make and investigate changes in the implemented econometrical techniques.

An approach to improve the flexibility of econometrical software products is suggested. The approach is based on a principle used in computer science to make the actual applications (e.g. problem oriented programming languages) as independent as possible of the hardware by introducing intermediate levels (languages). For econometrical software I distinguish three levels. First an application level for users who only want to apply standard econometrical techniques. Second a mathematical level for people with knowledge of the mathematical specification of econometrical techniques. Third a software level for computer scientists. Each level is made fully independent of the other levels. A user on a particular level does not need knowledge required on the other levels to implement changes on his own level.

An implementation of a threelevel system is developed at our University. This system is compared with systems based on related approaches, like GENSTAT with its macro-definitions and the MATRIX facility in SAS.

*) Econometrics Institute, University of Groningen, P.O. Box 800, 9700 AV Groningen, tel. 050-118004.

1. Introduction

There has been a lot of criticism on econometrics recently. Econometricians were accused of only being interested in theoretical subjects. This criticism is well-founded, indeed the gap between theory and practice in econometrics is growing wider. One of the most important problems causing this discrepancy is the huge effort needed to collect data and to implement software for new developed econometrical techniques and models. A large organization is needed to set up something worthwhile. Most small institutes do not have the required man-power and their employees remain theoretical scientists in an empirical science.

What can change this situation? I think more flexible software can. In this paper I will indicate how to create flexible software, which is better suited for use by econometricians than the existing tools. Most existing software tools are developed for users working in the basic disciplines of econometrics: statistics, economics and mathematics. A short evaluation of these tools is given in section 2. In section 3 I will describe why these software tools are less suitable for econometric research and show what is a better approach. The enormous increase in the amount of memory and speed of nowadays computers makes it possible to implement the concepts described in section 3. An example of such an implementation is given in section 4. In section 5 our approach is compared with related approaches: the macro facilities in GENSTAT and the MATRIX language in SAS.

2. Existing software tools in econometrics

Like in other applications areas, the use of the computer in econometrics started in the early sixties. In these days running a problem on a computer was a time-consuming affair. Debugging errors in a program took hours or even days and made that 'the new tool of science' was only used by hobbyists. An important feature of the use of computers at that time was that programs were only

written to solve one particular problem of one particular user. With the introduction of more friendly operating systems this situation began to change. More and more computerprograms were written in such a way that many people could use them. At present lots of these programs exist and econometricians use them frequently.

The introduction of these so-called application programs made that users could solve their problems with a minimal amount of effort. It appeared that there really was a market for this kind of software products. Universities in cooperation with software industry started to produce application programs containing what they believed to be the most important techniques in a certain science. An interesting software package for econometricians is for example Time Series Processor (TSP) originally developed at M.I.T. [11] in 1966. It contains statistical techniques frequently used by econometricians like full information maximum likelihood, instrumental variables, etc. An other good example is a standard package developed by Statistical Analysis System (SAS) Institute [8] and [9], containing a really impressive amount of statistical and time-series analysis techniques.

Besides the software tools mentioned above, the econometrician can use a programming language, in combination with a library of mathematical and statistical routines like IMSL [4], to solve his problems. This still happens a lot, especially in scientific environments, where one wants to investigate new instead of old techniques. Mostly programming languages like FORTRAN, PL/I, APL and PASCAL are used for this purpose. Within SAS it is also possible to use a language called MATRIX, which is a bit like APL. This language contains standard matrix operations like inversion, transposing, etc.

The two groups of software tools, software packages and programming languages, proved to be valuable in econometrics. But by no means they are, or ever will be, perfect. I will now describe some of the criteria which have to be considered in order to evaluate the usefulness of the software tools. A more detailed enumeration of criteria is given by Francis [2] who evaluated existing software packages in statistics. I will only concentrate on criteria important within the scope of this paper. These criteria are:

accuracy, flexibility and maintainability.

2.1. Accuracy

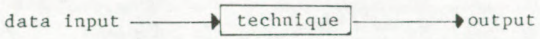
Every software tool should provide the user correct answers to his problem. In practice this goal is difficult to achieve. To prove that a large piece of software is correct is almost always impossible. Therefore it takes extensive testing to get confidence in the correctness of a computerprogram. In particular users of software packages are hardly aware of this problem. An overview of software reliability problems and testing is given by Ramamoorthly [7]. Besides the problem of correctness, the accuracy of a program is depending on the algorithms used to implement a certain technique or model. The fact that a computer can only represent numbers in finite precision, makes that seemingly correct algorithms may provide totally wrong answers on a computer. For a discussion of this subject see for example Forsythe, et.al.[1].

Regarding our two categories of software tools we may state that none of them can guarantee correctness or accuracy. In general however there is a tendency that larger organizations can produce more reliable software simply because they can do more testing. Also on other aspects like documentation and maintenance the larger organizations have a comparetive advantage.

2.2. Flexibility

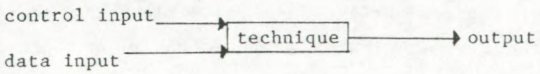
In econometrical research applications one does not know beforehand which models or techniques must be used. Software therefore should provide the possibility to make changes in the implemented techniques and models if required. Though software packages contain quite a few techniques, they do not offer the possibilities to modify these techniques. For a user the implemented techniques are a 'black box'. The user specifies a data set and the software package returns the answers (the output). This approach, visualised in figure 2.1, offers a minimum of flexibility to the user.

Fig. 2.1. Black box approach in standard packages



In some of the software packages the situation is slightly different. Besides data input the user can specify control input for the 'black box'. This offers the possibility to choose between different options of the software, containing variants of the implemented technique. Though more flexible than the 'black box' approach, the control approach

Fig. 2.2. Control approach in standard packages



still offers moderate flexibility. Only those variants that the developers of the package can think of are implemented. It is without doubt that the user will come up with a lot more. And it would be a great improvement if they could make the required modifications themselves.

If flexibility is so much needed, why not use a problem oriented programming language? Of course these languages offer enough possibilities to solve every econometric problem you can imagine. But the resulting software however has low flexibility due to for econometricians totally irrelevant information in the source text. An econometrician most of the time is not interested in problems like the exact implementation of matrix inversion or input-output routines written to make the software easily transportable from one machine to another. Due to the complexity of the source program, he can not easily adapt his software to new needs. We will return to this subject in section 3.

2.3. Maintainability and economic costs

The purchase costs of a software package are a small part of the total costs of most research projects. The costs of man-power form the bulk of the total costs. Most software products can not

be maintained by the user. Therefore the user is depending on the producer to get his software adapted to new needs. This process is very time-consuming and the costs involved in the delay will in fact be larger than the purchase costs of the software. Increasing the possibility for the user to maintain his own software will therefore be very profitable for the users.

3. The econometrician as a user of software

Whenever one develops software one must have a clear picture of the potential user. In this section I therefore briefly examine some definitions of econometrics and try to indicate the consequences of these definitions for the software. Malinvaud [5] gives the following definition of econometrics: 'Econometrics may be broadly interpreted to include every application of mathematics or statistical methods to the study of economic phenomena'. This definition encompasses a wide range of scientific activities. A usefull subdivision of these activities is for example given by Goldberger [3], who distinguishes the following main area's in econometrics:

a. mathematical economics: the mathematical formulation of economic theories.
b. technical econometrics: the development of appropriate techniques for econometric research.
c. empirical econometrics: the actual statistical inference from economic data.

It is clear from the enumeration above that changes in techniques (technical econometrics) and changes in the model specification (mathematical economics) itself are the subject of study in econometrics. Therefore all software developed for econometricians should be developed in such a way that changes in techniques and models can be implemented as easily as possible. The black box approach in most software packages of course does not offer this possibility and is less suitable for econometrical research.

In section 2 we defined flexibility as the possibility for the user to adapt the software to new needs. With regard to the

econometrical software it is useful to distinguish three levels on which changes can be made.

The application level : On this level changes can be made in a fixed set of parameters of a statistical technique or an economic model. For example:

a. changing the number of observations
b. changing the number of variables included in the analysis
c. changing the number of years for which a model has to produce forecasts.

The mathematical level : At this level changes can be made in the mathematical specification of statistical techniques or economic models. The mathematical specification must be as close as possible to the textbook notations of the implemented techniques or models.
For example:

a. modifying a procedure for ordinary least squares into a procedure for generalized least squares, by modifying the matrix notation of the OLS-procedure
b. modifying the specifications of one or more equations in an economic model.

The 'source' level : On this level changes in implementation of the software, the source, can be made.
For example:

a. changes in the memory organisation within a program (hashingtables, symboltables, etc.)
b. changes in the algorithms used for basic mathematical operations such as inversion and the calculation of eigenvalues

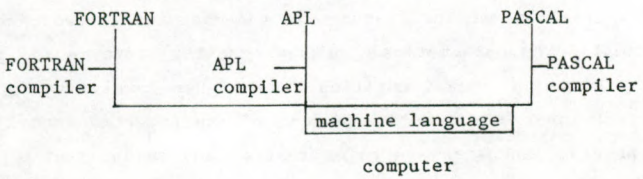
c. changes in the communication of a program with external files.

Of course the user must possess the required knowledge to make changes on a certain level.
No matter how well designed a software product is, it is only flexible if the user possesses the required knowledge to make the changes. For economic reasons most software produced until now, focussed on flexibility on the application level. If econometricians want to be able to implement new developed techniques, they need software which is flexible on the mathematical level as well. In the next section we show how to improve the flexibility for econometricians, making software changeable on application level and mathematical level.

4. Multi-level software for applications

The boundaries between software and hardware are no longer rigid. A lot of the functions performed by the software can be done by the hardware and vice versa. From this extreme point of view it is for example possible to create a FORTRAN computer. This computer would be able to perform FORTRAN-statements directly in his electric circuits. This approach, however, is not very flexible. One has to buy a new computer for every new programming language. A solution for this 'problem' is the introduction of one machine language and a set of compilers to compile other languages into the machine language. For the machine language a computer can be build. Thus different programming languages can run on the same computer. This situation is shown in figure 4.1.

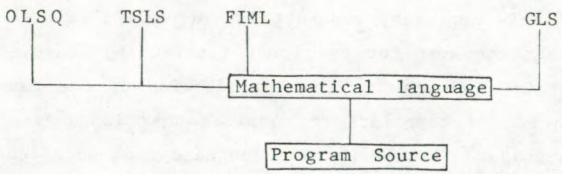
Fig. 4.1. A 'multi-language' computer.



In modern computers many intermediate languages, between the software packages and the actual hardware, exist (see Tanenbaum [10]). The application programs like TSP and SAS are thus just the top of an iceberg. Underneath the surface are a lot of other languages. Using compilation- or interpretation-techniques, the languages at the higher level are transformed to languages on a lower level, which can be executed through the electric circuits of the computer. The introduction of seperate levels makes it possible to locate certain decisions in one particular level of the software/hardware. The resulting flexibility of this approach is impressive. People can, for example, use TSP or SAS without concern for problems like optimality of an algorithm or the computer-architecture.

Now let us return to the problem of improving the flexibility of econometrical software. Can we use the multilevel approach for the construction of application software for econometricians? The answer is yes. The program languages in the above example can easily be replaced by mathematical techniques, the machine language by econometrical language and the computer by program source as shown in figure 4.2.

Fig. 4.2. A multi-technique application program



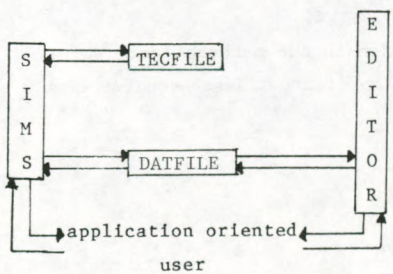
What is the difference with ordinary application programs you will ask. The difference is the introduction of an intermediate level between techniques and application program: the mathematical language. Like the machine language in the computer software/hardware example, the mathematical language greatly improves the possibility to modify or add entities on a higher level. The implemented techniques are made independent of the program source just as the program languages are to a great extend independent of the computer they are running on.

On each of the three levels in figure 4.2 there exists a language. On the highest level we have the application language. This language enables the user to make the required application level changes described in section 3. On this level the user can perform the techniques described in the mathematical language on the mathematical level. In the econometrical language the mathematical level changes of section 3 can be made. On the bottom level there is the programming language in which the program source is written. Here the source level changes of section 3 can be made. For the implementation of these languages translation and interpretation techniques can be used.

The application language and the programming language do not need much further explanation. The application language should resemble the languages used in most standard statistical software packages, the programming language is one of the languages like FORTRAN, PASCAL, APL, PL/I, etc. The mathematical language, however, needs to be developed with care. It must contain the mathematical operations and functions used to describe econometrical techniques. Examples are matrix inversion, eigenvalues, eigenvectors, kronecker products, etc. The language will somewhat resemble MATRIX in SAS. All level 3 facilities, however, should be removed.

At our university we have implemented a system with the features described above. We have build an interpreter that can handle the mathematical language and the application language. This interactive interpreter is called Standard Interactive Multilevel System (SIMS). At the application level SIMS can be used in the following manner. The user can select econometrical techniques or models from a file TECFILE.

Fig. 4.3. SIMS at the application level

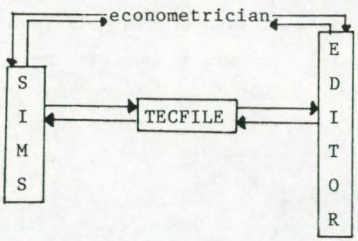


The techniques or models can be used to analyse data, which the user can select from a datafile DATFILE or which can be typed in on his terminal. An example of the application language in SIMS is given below:

```
SELECT USING OLSQ ON TECFILE
SELECT USING DATA1 ON DATFILE
SHOW BETAHEAD,RSQUARED
RUN (during the run the variables in the SHOW-command are
    displayed)
SELECT USING DATA2 ON DATFILE
RUN (now the results of the calculations on the second
    dataset are displayed)
```

In this example a user selects a statistical technique and analyses two data-sets applying this technique. The technique, in this case ordinary least squares, is written in the mathematical language. An econometrician can create or modify these techniques with the use of an editor or in an interactive dialogue with SIMS. This situation is shown in figure 4.4.

Fig. 4.4. SIMS at the mathematical level



The content of the TECFILE, written in the mathematical language of SIMS, for this example of ordinary least squares is:

```
OLSQ
  VAR [10 by 1] Y,E
  VAR [10 by 3] X
  VAR [3  by 1] BETAHEAD
```

```
VAR RSQUARED,N
DEFINE  BETAHEAD  =  INV(X'*X)*X'*Y
DEFINE  E  =  Y  -  X*BETAHEAD
DEFINE RSQUARED = 1 - E'*E/((Y'*Y)-sigma(y)∧2)/N)
```

All variables used in the description of ordinary least squares are declared by VARIABLE (or abbreviated VAR) statements. For some of the variables the necessary equations are described in the DEFINE (or abbreviated DEF) statements.
New statistics and modifications of the existing equations can easily be made by an econometrician. For example the adjusted correlation coefficient can be added simply by appending the following lines to the TECFILE.

```
VAR  K,RSQUAREDADJUSTED
DEFINE RSQUAREDADJUSTED = 1 - ((N-1)/(N-K))*(1-RSQUARED)
```

Appendix A gives an overview of the mathematical operations and functions currently present in SIMS. The econometrician can use the mathematical language described by these operations and functions with practically no knowledge of the implementation on the lower level.

On the lowest level in our implementation we have the programming language in which the program SIMS is written: PASCAL; at this level the source level changes of section 3 can be made. A short enumeration of the problems which had to be solved to implement SIMS is listed below.

A. Memory Organisation

(i) SIMS must be able to store and retrieve all information concerning variables and formulas in an econometrical technique or model as efficient as possible. The number of variables and formulas that a specific technique or model contains can vary. Therefore a symbol table is constructed with dynamic memory allocation.

(ii) The formulas may contain matrices. These matrices must be stored according to their shape. Triangulair-, sparse-, symetric-, identity or normal matrices all should be

treated differently.

(iii) Formulas can best be stored using polish notation. SIMS must be able to convert the infix notation used in the econometric language into the prefix notation in the polish notation.

(iv) If the user specifies a variable or keyword SIMS must be able to allocate the information concerning this item as quick as possible. A hashing function is used to find the required information in the symbol table.

B. Communication with the user

(i) The user can communicate with SIMS using a command language. If the user types in a command, then SIMS must perform a lexical scan. It reads the command line token (a string of characters representing a variable name, function name, operator, etc) by token. After each token SIMS knows what to do next or what input is expected. If errors are made (the system does not understand a certain token in the input line) SIMS gives a diagnose and offers the possibility to correct the error. An example of the error handling in SIMS is given in Appendix B.

(ii) If the user does not know what to do next, then he is able to ask SIMS for information. Help facilities make SIMS more easy to use.

C. Implementation of algorithms

(i) To get accurate results, sophisticated algorithms for the mathematical functions and operators must be used. For example the inverse of a matrix can be calculated using Cholesky decomposition.

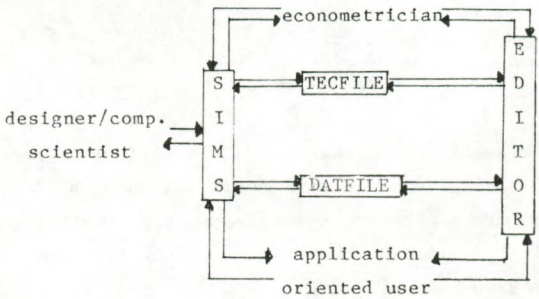
(ii) Once the model is defined within SIMS the user can use it to calculate results for specific data sets. SIMS must locate the specified formulas and the values of the variables in his memory. After the run the results must be stored in the symbol table.

Changes at the source level can only be made by people with fair knowledge of PASCAL and insight in the architecture of SIMS. Be-

cause SIMS is developed in a modular fashion it is relatively easy to add new functions and procedures to SIMS.

The resulting software package is a package that can be used by application oriënted users, econometricians and computer scientists independently. Using their own language they can modify the system to their own needs. The complete system is visualized in figure 4.5.

Fig. 4.5. The SIMS system



5. A short comparison with other software tools

In this section we compare SIMS with other software products, that show some resemblance with SIMS: the macro-facility of GENSTAT [6] and the MATRIX language in SAS [9]. Of course SAS and GENSTAT are developed with other objectives than discussed in this paper. Both packages focus on the efficient implementation of a set of standard statistical techniques.
The macro facility in GENSTAT and MATRIX in SAS are added to allow the user to write his own routines. We will indicate the main differences with SIMS.

5.1. GENSTAT

It is possible to regard the macro facility in GENSTAT as a three level system. A user can write and modify macros (the mathe-

matical level) and the macros can be used by other users through a macro call (the application level). The main difference between SIMS's and GENSTAT's macro facility lies in the implementation of the mathematical language and the independency between the three levels. The language on the mathematical level does not resemble the mathematical language used in the mathematical text book. All operations are performed through procedure calls. The formula of for example OLS must be written as

```
'CALCULATE     XACX     = TPDT(X,X)
               XXINV    = INV(XACX)
               XACY     = TPDT(X,Y)
               BETAHEAD = PDT(XXINV,XACY)
```

which is far from resembling original textbook notations. Furthermore lots of level 3 decisions are made on the mathematical level. For example the output of the macro and memory organization commands like 'devalue' (return occupied space) and 'pointers' are controlled on the mathematical level.

5.2. MATRIX in SAS

The MATRIX language in SAS is a good example of how the mathematical language should be implemented. All formulas can be written in readable form and thus can be used and modified easily by econometricians. Unfortunately MATRIX can not be used as a 'black box' on the application level. It is not possible to call a 'MATRIX-procedure' from a library and assign values to certain parameters. Values are assigned to variables on the same level as the definition of the formulas. As a result the distinction between application level and mathematical level can not be made in SAS. There remain lots of other differences between the software products not discussed in this section. A full enumeration, however, is outside the scope of this paper, where we only focus on flexibility of the software.

6. Conclusions

On nowadays computers it is possible to implement systems, that are far more flexible than the 'black-box' approaches in most standard packages. With these systems it must be possible to decrease the gap between theory and practice in econometrics. Of course there remains always a trade-off between flexibility on the one hand and computing efficiency on the other. Nevertheless I believe that flexible systems will become more and more important, especially in scientific environment. With SIMS I hope to make a small contribution in this direction.

Appendix A. Operators and functions available in SIMS

Operators

'	TRANSPOSE.
^	RAISE TO A POWER.
*	MULTIPLY.
/	DIVIDE.
+	ADD.
-	SUBSTRACT.
<	LESS THAN.
<=	LESS THAN OR EQUAL TO.
>	GREATER THAN.
>=	GREATER THAN OR EQUAL TO.
=	EQUALS.

The operators <=, <, >=, > and = can perform operations on scalars only. The operator ' only operates on matrices. The operators +, -, and * operate both on scalars and matrices. If two operands are of a different type (e.g. a scalar and a matrix) the result of the operation will be a matrix and the operation is performed on the scalar and each matrix element.

For example:

```
VAR[2 BY 2]A
DEFINE A = A + 1
```

In each run the variable A is increased by one.

The operators ^ and / can be used either for operations on two scalars or for operations on a scalar and a matrix.

Functions

Boolean functions

REAL(EXPR) if the boolean expression EXPR is true this

function returns the value 1.0, else 0.0.

matrix functions

EIGVAL(A)	The eigenvalues of A.
EIGVEC(A)	The eigenvectors of A.
IDENT(A)	Creates an identity matrix of the same size as matrix A. A must be square.
INV(A)	Inverse of matrix A. A must be square.
KRONECK(A,B)	The Kronecker product of A and B.
TRANS(A)	Transpose of matrix A.
SIGMA(A)	The sum of the elements of A.

Scalar functions

ABS(X)	The absolute value of X.
EXP(X)	The exponential functi
LN(X)	The natural logarith
MAX(X,Y)	The maximum of X a
MIN(X,Y)	The minimum of

Random number generators

NORMAL(A,B)	Generat norma dev
UNIFORM(A,B)	

Appendix B. The errorhandling in SIMS

Errors in the SIMS-commands should be detected as soon as
sible. By doing so the amount of retyping erroneous commands is
ized. When SIMS detects an error it responds by giving a
tion of the probable cause of the error and indicates where
in the command occured. For example. If a definition
tains an undeclared variable Z:

X + Z

pond

R

and >> in the following line.

...>>, (possibly nothing or
the remaining input from

ated. The user
retyping the
eeds to be

an
rs

Literature

[1] Forsythe, G.E., M.A. Malcolm and C.B. Moler, 'Computer Methods for Mathematical Computations', Prentice Hall, New Jersey, 1977.

[2] Francis, I., 'Statistical Software: A Comparative Review', Elsevier North-Holland, New York, 1981.

[3] Goldberger, A.S., 'Econometric Theory', John Wiley & Sons, Inc., New York, 1964.

[4] IMSL Corporation, IMSL Reference Manual, Houston, 1977.

[5] Malinvaud, E., 'Statistical Methods of Econometrics', North Holland, New York, 1980.

[6] Nelder, J.A., 'GENSTAT manual', Rothamsted Experimental Station, Harpenden, Herts, U.K., 1977.

[7] Ramamoorthly, C.V. and F.B. Bartari, 'Software Reliability-Status and Perspectives', IEEE Transactions on Software Engineering, Vol. SE-8, No. 4, July 1982.

[8] SAS, 'User's Guide: Basics', SAS Institute Inc., Cary, NC, 1982.

[9] SAS, 'User's Guide: Statistics', SAS Institute Inc., Cary, NC, 1982.

[10] Tanenbaum, A.S., 'Structural Computer Organization', Prentice Hall, New Jersey, 1976.

[11] TSP, 'Time Series Processor', User's Manual, Computing Centre of Western Ontario, Canada, 1980.

Ontvangen: 30-5-84