GAUSSX: Version 3.4*

Richard G. Pierse
Department of Economics, University of Surrey, Guildford, Surrey, GU2 SXH, UK

31 May 1996

1 Introduction

Software for econometrics can broadly be classified into three groups. First, there are menu- driven packages such as MicroFit and PcGive. The main advantage of these is ease of use and the main disadvantage is inflexibility — unless it is on a menu, you cannot do it. Second, there are command-driven languages such as TSP and RATS. In these programs, the user enters a sequence of simple commands, each command including an optional set of parameters. The commands can either be executed one by one (interactive mode) or more typically are collected in a file and executed all at once (batch mode). Command-driven packages are less easy to use than menu-driven programs as commands can easily be mistyped, and this often causes obscure error messages. On the other hand, they are somewhat more flexible than menu-driven programs, generally providing some looping facility for repeating commands and matrix operations to allow the user to define estimators and test statistics beyond those provided by the predefined commands.

The third software group consists of high-level programming languages such as GAUSS and MATLAB.[1] In a sense, these are not really econometrics packages at all, since econometric estimators and tests are generally not included in the set of predefined routines of the language. However, the generality of these languages and their matrix-orientated nature means that virtually any estimator or test can easily be defined within the language. Although much more flexible than command-driven programs, the high-level programming languages require a much higher level of skill in computer programming. Learning GAUSS, for example, requires a large investment of both time and effort, and GAUSS routines tend to be written by dedicated experts. Despite this, a large range of third-party software for econometrics written in GAUSS now exists, some available directly from the supplier and some from the authors or through the Internet.[2]

GAUSSX is a program written in the GAUSS language that transforms the GAUSS environment and makes it act as a command-driven econometrics package very much like TSP. Written by Jon Breslaw, a Professor of Economics at Concordia University, Montreal, it is distributed as an optional add-on to GAUSS through Aptech, the GAUSS supplier. GAUSSX aims to combine the advantages of command language and high-level programming language. On the one hand, it provides a set of commands to execute all the standard econometric techniques and can be used without any knowledge of GAUSS. On the other, GAUSSX has access to all the functionality of GAUSS for those who need to use it. GAUSSX is available on the PC in two versions, one for DOS and the other for Windows. The Windows version is discussed in Section 5 of this review.

2 Installation

GAUSSX Version 3.4 comes on two high-density floppy disks; the Windows version (see Section 5) is on a third disk. It requires versions 3.1 or 3.2 of GAUSSI (the current 32-bit version of GAUSS for the PC) and will not run with 16-bit versions or GAUSS 2.x. A student version, which runs under GAUSS-LIGHT, is also available. The hardware requirement is a PC-compatible computer with either

*Published in *Journal of Applied Economics* (1996), Vol. 11, pp. 687–693. The author is grateful to Jon Breslaw and James MacKinnon for helpful comments.

[1]GAUSS and MATLAB have been reviewed before in this journal as well as elsewhere. See Anderson (1992), Rust (1993) and Smith and Smith (1995).

[2]See the Appendix for details about an archive of GAUSS routines available on the Internet.

a 386 CPU (with maths coprocessor) or a 486 or Pentium CPU, with at least 4MB of memory and a hard disk. Installation is relatively straightforward but requires the previous installation of GAUSS. The installation program copies all the files from floppy disk onto subdirectories of the main GAUSS directory and creates the main program by compiling the source code. The manual warns that this last stage is time consuming although on my 66Mh 486 machine it only took a few seconds. The compiled files require about 700 K of disk space. At one point in the installation procedure, the CONFIG.SYS system file is listed to ensure that the FILES and BUFFERS parameters are large enough and that the ANSI.SYS device driver is installed. Since these are requirements of GAUSS rather than GAUSSX and the former should already be installed, this seems rather unnecessary. Later on, two GAUSS configuration files are updated. Both times, the user is left in an editor and has to find the way out (press F7 and agree to save changes) which I found a little confusing.

The program comes with a 220-page manual. Apart from initial chapters on installation and the GAUSSX desktop, and some technical appendices, the bulk of the manual consists of an alphabetical listing of every GAUSSX command in a format very close to that of the GAUSS command reference manual. There is a brief description of each command, a list of available options and some examples of use, but very little background on the econometric and statistical techniques involved and no references to the literature. The alphabetic nature of the manual can make finding the appropriate GAUSSX command a little difficult, although there is a useful summary of commands grouped by type. I found the best introduction to the program was the set of sample command files that come with GAUSSX, which are well commented and cover most of the main features.

3 The Desktop

GAUSSX is started from the DOS prompt by the command GAUSSI GAUSSX, or alternatively from the GAUSS prompt by typing RUN GAUSSX. Entering GAUSSX brings up the desktop screen. This is the main menu from which files of GAUSSX commands can be edited or executed and where the resulting output can be viewed and printed. Although it operates in text mode, the desktop adopts the familiar Macintosh and Windows GUI style with pull-down menus, pop up windows, scroll-bars etc., selected using either keyboard or mouse. The actual implementation of this interface has rather a rough feel; moving the scroll bars, for instance, tends to jump from one end of a list to the other, and sometimes I had trouble getting the program to accept a file name typed into an edit box (rather than selected from a list). These are minor quibbles, however, and the familiar look of the desktop makes learning to use it easy. Options on the pop-up menus at the top of the screen allow the desktop to be customised (changing the colour scheme, default editor, etc.).

The desktop organizes your work into projects, each having an associated command and output file and default paths to be used for reading data and storing temporary files. Up to 25 projects can be maintained, and details of each project are stored in a file called DESKTOP.PRJ.

Selecting the Edit button brings up a text editor, allowing the user to make changes to the file of GAUSSX commands. By default, the editor used is TED.COM although it is possible to configure the desktop to use a different editor. Context-sensitive help on the GAUSSX command at the current cursor position is available with the ALT-H key combination which brings up a screen showing the form of the command and available options. This is an extremely useful facility, making it very easy to correct syntactical mistakes or check up on options without needing to get out the manual.

Once a file of GAUSSX commands has been created in the editor, the commands can be executed by selecting the Run button. The commands are first translated into GAUSS code and then executed by GAUSS itself. Error processing is shared between GAUSSX and GAUSS. Syntactical errors will tend to be trapped by GAUSSX which will return you to the desktop. On the other hand, run-time errors will generally be trapped by GAUSS which returns you to the GAUSS prompt. This is initially rather disconcerting, as it appears that GAUSSX has disappeared. In order to get back to the desktop you need to type GAUSSX or use the ALT-F2 key. On return to the desktop, the output file can be examined in a text viewer or sent to a printer.

4 Commands

The commands available in GAUSSX fall into several categories. Single-equation estimation methods include OLS, 2SLS, and AR (for estimation of linear models with autoregressive errors). System methods include SURE, 3SLS and FIML for linear models and non-linear least squares (NLS), FIML and

GMM for non-linear systems. There is also a general maximum likelihood estimation method (ML) for user-supplied likelihood functions. Options to pre- weight the data and to calculate heteroscedasticity-consistentstandard errors are available for most of these methods and polynomial-distributed lags can be used with all the linear methods. ARCH and GARCH-M processes can be specified and estimated and the KALMAN command allows the setting up of state space models. Non-linear equation systems can be solved, either dynamically or statically, using the SOLVE command.

Time-series methods include the identification and estimation of ARIMA models, moving-average seasonal adjustment (SAMA) and exponential smoothing (EXSMOOTH). Linear models with qualitative dependent variables can be estimated with the QR command using binomial probit, multinomial logit or ordered logit or probit. Non-linear multinomial logit (MNL) or probit (MNP)models can also be set up and estimated using the ML command.

The TEST command computes a variety of diagnostic tests including t-tests, F-tests, likelihood ratio tests, the Davidson and MacKinnon J-test for non-nested models, the Dickey -Fuller test for unit roots, and the Engle-Granger test for cointegration. General non-linear Wald tests can be computed following a regression by using the ANALYZ command.

To readers who know the econometrics package TSP, this list of GAUSSX features will sound very familiar. In fact, the choice of commands offered by GAUSSX is so similar to that available in TSP, many having identical names and options, as to make it necessary to undertake a direct comparison of the two packages.[3] In terms of the predefined commands available, there is not a lot to choose between them. Generally, the output produced by the corresponding procedures is very much alike, with, by and large, the same range of statistics provided. Reassuringly, the estimation methods involving non-linear optimisation[4] converged to the same solution values.

Like TSP, GAUSSX has a GENR command to define data transformations, although the two commands do not always behave in the same way. For example, the command '**GENR Z = X*Y;**' which in TSP multiplies together two variables, causes an error in GAUSSX, because the operator '*' is treated as a matrix multiplication operator and X and Y, being column vectors, are not conformable. To get the desired result in GAUSSX, the GAUSS element by element multiplication operator '.*' needs to be used instead. More generally, GAUSSX transformations can access any of the very wide range of operators and functions of GAUSS, although this requires some knowledge of the GAUSS language.

TSP 4.3 provides some features not available in GAUSSX. For example, Johansen (trace) tests for cointegration can be computed as an option in the TSP COINT command.[5] A PANEL command estimates linear regression models for panel data. Conversely, there are a few GAUSSX commands which are not available in TSP. These include a command (NPR) to estimate non-parametric regression statistics and two statistical routines: one for generating random numbers from truncated multivariate normal distributions (RNDTN) and the other to integrate multivariate normal densities (QDFN). More importantly, however, GAUSSX command files can incorporate pure GAUSS code and so can call on any of the hundreds of mathematical and statistical functions available in GAUSS.

One important difference between the two packages concerns the way in which derivatives are treated in non-linear models. TSP automatically computes analytic derivatives and uses these in estimation and model solution. This capability is one of the most impressive features of the program. GAUSSX does not itself have the facility to compute analytic derivatives, but does allow the user the option of supplying them. The method in which this has to be done varies between commands. For the SOLVE command, the optional parameter JACOB allows the user to supply the Jacobian matrix. For the ML command, the user needs to write a procedure in GAUSS to return the gradient vector (and, if estimation is by the Newton method, also the Hessian matrix). Having the user supply analytic derivatives (especially getting the user to write a GAUSS procedure to compute them) is clearly far less convenient than having the program compute them for you. If analytic derivatives are not supplied, then GAUSSX uses numeric derivatives instead. This will considerably increase computation time and may lead to numerical difficulties, either in finding an optimum or in computing the parameter covariance matrix.

[3]I am grateful to Clint Cummins at TSP International for making available to me a copy of TSP 4.3 for this review.

[4]Interestingly, GAUSSX seems to use different optimisation algorithms from those in the GAUSS Application modules OPTMUM and MAXLIK. More generally, GAUSSX does not depend on the GAUSS Applications modules, with the exception of the mEQ and CROSSTAB commands, which use the BASIC STATISTICS module. Even this dependency will be eliminated in the next version.

[5]'Since writing this, I learned that a cointegration procedure has been added to GAUSSX, although it is not yet in the manual.

5 The Windows Version

GAUSSX 3.4 is also available in a version with a Windows interface. This is a Windows version of the desktop rather than a full Windows implementation of GAUSSX, since GAUSS itself remains a DOS program. Consequently, when the run button is clicked to execute a file of GAUSSX commands, the screen switches to a full-screen DOS box until the command file has finished executing. It is possible to run other Windows applications while the GAUSSX job executes in the background by using the switch tasks key (ALT-TAB).

Although the documentation is very brief (it consists of a three-page leaflet), installation proved to be very straightforward. The desktop itself has very similar buttons and menu options to those of the DOS version, although the operation was much smoother. However, I found the look of the main window rather cramped and the non-standard colour scheme made some of the buttons difficult to read.

The default editor, Gxedit, is different from the editor in the DOS version. It has a Windows help system that gives help both on the use of the editor itself and also on all the GAUSSX commands. An additional button gives context sensitive help analogous to pressing ALT-H in the DOS version. There is a problem with exiting the editor using the Close window option on the system menu (ALT-F4). When I tried this, it closed not just the editor but also the GAUSSX desktop itself, leaving an iconised GAUSS window which it was then impossible to close, so that the only way to exit Windows was by rebooting the computer. Thus the only safe ways to close the editor are the exit option on the file menu (ALT-F x) or the ESC key.

The format used for Project files is also different from that in the DOS version. A conversion utility is available and this seems to automatically detect any changes made to your DOS Project file and offers to convert it for you. Conversion is very quick, but switching frequently between DOS and Windows versions of GAUSSX is a little inconvenient.

Overall, I did not find any real advantages of the Windows version over the DOS version. The problems I encountered may be due to the preliminary nature of the interface, described as ‘a first stab’ by the author. However, even when these are sorted out, it is not clear that there is much benefit to be gained from running what is still basically a DOS program in a Windows environment. Only when the Windows version of GAUSS eventually becomes available will the advantages of a Windows interface for GAUSSX be realised.

6 Adding to GAUSSX

GAUSSX comes with full GAUSS source code. This is not copied from the floppy disks in the standard installation, but it can be found on the second installation disk in the subdirectories MODULE13 and MODULE14. Having the source code means that it is possible for a user with a knowledge of GAUSS, to modify existing GAUSSX commands or even to write completely new ones. Appendix F of the manual gives a brief description of the steps to be followed together with a chilling warning about its being somewhat technical and not to be attempted except by experienced users. Undeterred, I decided to try out this facility by modifying a GAUSS procedure of mine and turning it into a new GAUSSX command. The procedure I chose uses the Johansen (1988) methodology to test for the number of cointegrating vectors among a set of variables; a natural name for the new GAUSSX command was thus JOHANSEN. Following the instructions in the manual, I modified the code for an existing GAUSSX command, AR. Using AR as a template, I designed the new command with syntax:

JOHANSEN (options) *vlist*; METHOD = *methname*; ORDER = *lag*;

where *vlist* is the list of variables, *lag* is the order of the VAR and *methname* specifies the Johansen model and is one of NONE (no intercept), UNREST (unrestricted intercept) or REST (intercept restricted to the error correction term); *options* is the list of standard display options.

The complete routine was compiled in GAUSS and saved. Then it was necessary to make changes to three different files so that the new command would be added to the list of GAUSSX commands. This was a bit laborious and the manual instructions at this point were a little unclear, but in practice the process proved reasonably straightforward. Finally, a new version of GAUSSX had to be created by running a GAUSS job called COMPILE.PRG.

At this stage it was possible to run GAUSSX and test out the new command. Needless to say, it did not work first time round. However, sorting out the problems was much easier than I had initially feared, because GAUSS error messages directed me back to the offending lines in the source code and all that was required was to correct the errors and recompile the routine. The whole process of adding the new command and getting it working took about two days, most of that time being taken up in the learning phase, which is a one-off task.

The facility to add new commands to GAUSSX is a very useful one.[6] If third-party writers of GAUSS procedures could be persuaded to make these available in the form of GAUSSX commands, then it would greatly facilitate the dissemination of knowledge.

7 Conclusions

GAUSSX is a command-driven econometrics package that provides much the same capabilities as TSP and has a very similar command language. Choosing between the two will depend largely on how much the user needs to make use of the additional background facilities of GAUSS. If these are needed then GAUSSX is the obvious choice. In an institution where GAUSS is already used in research, a copy of GAUSSX would be a very useful addition for teaching, particularly if use is made of the facility to augment its command set to make available to students the latest econometric techniques in the form of simple commands.

If the extra facilities are not required, then I would still choose TSP, because it has the advantage of additional simplicity (there is no hidden layer) and consumes less disk space. However, considered purely as an example of a GAUSS program, GAUSSX stands as an impressive illustration of the flexibility and power of the GAUSS programming language.

A Information About GAUSSX and GAUSS

GAUSSX is developed by Econotron Software, 4164 Henri-Julien, Montreal, P.Q., H2W 2K3, Canada; *e-mail: breslaw@vax2.concordia.ca.* The program is distributed by ApTech Systems, Inc., 23804 SE Kent-Kangley Rd, Maple Valley, WA 98038, USA; *e-mail: info@Aptech.com.*

A fair amount of information about GAUSSX and GAUSS is now available on the Internet:

(1) Jon Breslaw maintains a World Wide Web page for GAUSSX at the site: *http://www.netaxis.qc.ca/people/j.breslaw/gaussx/gaussxl.html.* This has a detailed description of all the features available in GAUSSX and includes some nice sample screen images from the program.

(2) An electronic mailing list exists through which users of GAUSS can communicate, exchange information and share problems and solutions. Note that this mailing list is quite heavily used and subscribers can expect several mail messages per day from fellow Gaussians. To join, send e-mail to *Majordomo@mundo.eco.urexas.edu* with the following message:

subscribe gaussians (your e-mail address)

(3) An archive of GAUSS source code is available (for non-commercial use only) from a gopher site run by the Economics Department of the American University in Washington, DC. Many GAUSS procedures may be found there on diverse topics including cointegration, Bayesian analysis and optimisation. The WWW address for this site is: *gopher://gopher.american.edu:70/I I /academic.depts/cas/econ/software/gauss/.*

References

Anderson, R. G. (1992), 'The Gauss programming system: a review', *Journal of Applied Econometrics*, 7, 215–219.

Johansen, S. J. (1988), 'Statistical analysis of cointegration vectors', *Journal of Economic Dynamics and Control*, 12, 231–254.

Rust, J. (1993), 'Gauss and Matlab: a comparison', *Journal of Applied Econometrics*, 8, 307–324.

Smith, J. and P. N. Smith (1995), 'Gauss 3.2 and Coint 2.0', *Journal of Economic Surveys*, 9, 89–101.

[6]TSP for mainframes and workstations has a similar facility to add user-written routines coded in Fortran. However, this is not available in the PC version.