

Bridging the Gap Between Ox and Gauss using OxGauss

Sébastien Laurent¹ and Jean-Pierre Urbain²

July 2003

Preliminary version

Abstract

We review and discuss some improvements brought to OxGauss. OxGauss enables the user to run under Ox a wide range of Gauss programs and codes without having to install Gauss on his/her machine. The advantage of OxGauss is that Gauss codes can be called from Ox programs, or can be run and executed on their own. While the new version of OxGauss (provided with Ox 3.3) is extremely powerful in many situations, it becomes of little use once the purpose is to execute a program that attempts to solve an optimization problem using one of the three well known Gauss application modules (Cml, Maxlik or Optmum). In this paper we also introduce a set of additional procedures that contribute to bridge the gap between Ox and the above mentioned optimizers. The effectiveness of these procedures is illustrated by revisiting a large number of Gauss codes. These codes are freely available on the internet and use Gauss application modules that require numerical optimization. Most of these programs deal with highly non-linear models such as the Markov regime-switching of Hamilton, STAR models and various GARCH-type models.

These illustrations highlight a further potentially interesting implication of OxGauss: it enables non-Gauss users to replicate existing empirical results using freely available Gauss codes.

¹CREST (CNRS) and CORE (Université catholique de Louvain). E-mail: Laurent@core.ucl.ac.be.

Correspondence to CORE, Voie du Roman Pays, 34, B-1348 Louvain-La-Neuve, Belgium.

²Department of Quantitative Economics, University Maastricht, The Netherlands. E-mail: j.urbain@ke.unimaas.nl.

While remaining responsible for any errors in this paper, the authors would like to thank Jean-Philippe Peters for useful comments and Charles Bos and Jurgen Doornik for their accessibility.

1 Introduction

With the important new technical advances made in econometrics over the recent years and the ever increasing use of computer intensive modelling techniques and estimation methods, applied econometricians often face the need to write programs to implement these recently developed approaches and techniques. While existing standard econometric software (RATS, TSP, ...) or pre-packaged routines are sometimes available for the applied researcher, one must observe that many recently developed techniques are not so rapidly implemented in standard econometric software. Examples are the estimation of various non-linear time series models, nonstationary panel data techniques,... The same holds for many computer intensive techniques like bootstrap inference, non-parametric and semi-parametric analysis that require moreover very efficient programming to be used in real world applications. In this case, it is very likely that the researcher that wants to use these techniques will have to put one's shoulder to the wheel. As pointed out by Cribari-Neto and Zarkos (2003) in their review of Ox, programming forces a researcher to think more deeply into the issue that is being investigated. It may however sometimes be rather inefficient to programme complex procedures, estimation techniques that have already been programmed, used and checked by other researchers for the sole purpose of application or replication of results.

The replicability of simulation and empirical results in econometrics is recognized as being a fairly important aspect of research as exemplified by the practice of the *Journal of Applied Econometrics* that asks authors to make their data and possibly specialized codes available to the potentially interested reader. The availability of these enables the reader to replicate the results obtained in a particular study. Not surprisingly, an increasing number of researchers in econometrics are making their codes and routines freely available to the econometrics community. This has led to an increase in the exchange of routines that were initially prepared by researchers for their own work, very often for illustrating theoretical advances using some Monte Carlo simulations or real data sets.

Sharing codes has hence become a fairly standard practice in econometrics. Of the various programming environments used in econometrics, Gauss¹ is probably one of the most, if not the most, popular programming environments for the last two decades along with S-Plus, MATLAB and now

¹Gauss is sold by Aptech Systems, 23804 S.E. Kent-Kangley Rd., Maple Valley, WA, 98038, USA; see <http://www.aptech.com/>.

Ox.^{2,3} The many codes available through several Gauss archives provide for the applied researcher a unique and important opportunity to implement procedures that are otherwise often demanding to programme.

In order to benefit from this availability however, the researcher who is willing to replicate or at least to apply the techniques programmed in these codes should preferably be a Gauss user, or at least should have a direct access to a legally registered version of Gauss. The price of many software packages (including Gauss) has however increased substantially over the years, making sometimes the acquisition of expensive packages very hard for small research groups or department or even more importantly for researchers located in third world countries where the financial resources for softwares are rather scarce. There are (besides the whole GNU project) interesting initiatives of some statisticians or econometricians that have led to a number of freely available (at least for research and teaching purpose) econometric and/or statistics software like Easyreg, ECTS, Vista (Visual Statistics System), GRETEL (Gnu Regression, Econometrics and Time-series Library), or some low-cost alternatives/clowns of Matlab (O-Matrix, Octave) and S-Plus (R). All these do still not give the opportunity to benefit from the large number of Gauss codes available. Hence the importance of OxGauss that provides the researcher with a rather simple and free solution for running Gauss codes easily.

The advantage of OxGauss is that, even with the *free* (or console) version, Gauss codes can be called from Ox programs, or can be run and executed on their own. While OxGauss has in this sense similar possibilities than O-matrix that can run most Matlab codes, the crucial difference is that OxGauss comes freely with the console version of Ox.

While the new version OxGauss (provided with Ox 3.3) is extremely powerful in many situations, it becomes nevertheless of little use once the purpose is to execute a program that attempts to solve an optimization problem using one of the three well know Gauss application modules Cml, Maxlik or Optmum. This was already noted by Viton (2001) ” ... *not every piece of Gauss code will be convertible to OxGauss. As weve seen, one important case is when it uses Gausss Maxlik procedure; or more generally, other extra-cost Gauss add-ons. Even in the Maxlik case it may be possible to substitute Oxs own optimization routine; if anyone knows how to do this, Id like to know. In the mixed-logit case we were fortunate in being provided with an alternative optimization routine domax written in Gauss as a substitute to Maxlik; and this could be converted. So another possibility is to*

²There are two versions of Ox. Oxconsole can be downloaded from <http://www.nuff.ox.ac.uk/Users/Doornik/>, which is the main Ox web page. The console version is free for educational purposes and academic research. The Professional Windows version, or commercial version comes with a nice interface for graphics known as GiveWin (available for purchase from Timberlake Consultants, <http://www.timberlake.co.uk>).

³For a third party comparisons of the relative performances of various mathematical programs, see Steinhaus (2003).

map Maxlik and its output generally to domax. Again, if anyone does this and would like to explain it, Id be glad to include the information here.”

This paper proposes to fill this gap. In particular, we propose a set of additional procedures that contribute to bridge the gap between Ox and the above mentioned optimizers. The effectiveness of the procedures is illustrated by revisiting a large number of Gauss codes that are freely available on the internet and that use Gauss application modules that require numerical optimization. These include many programs dealing with highly non-linear models such as the markov regime-switching of Hamilton, STAR models and various GARCH-type models.

These illustrations will again highlight a further potentially interesting implication of OxGauss as it, for example, enable non-Gauss users to replicate existing empirical results using freely available Gauss codes.

This paper is structured as follows: in Section 2, we propose an overview of OxGauss and give some simple examples as well as a speed comparison between Ox, OxGauss and Gauss 3.5. Section 3 discusses the graphical issue. Section 4 presents the package M@ximize that links Cml, Maxlik and Optmum to the Ox optimizers. Finally, Section 5 concludes.

2 OxGauss

OxGauss is an integral part of Ox 3.3. For a presentation of OxGauss and a detailed listing of the existing Gauss functions/procs currently supported we refer the reader to the `/doc/OxAppendix.pdf` file provided with Ox 3.3. It might nevertheless be important for non-Ox users or newcomers to point out once again that, in contrast to the old `g2ox.exe` program provided with Ox, OxGauss is **not** a translator from Gauss to Ox.

OxGauss enables a user (even of the console version) of Ox to:

- (1) call an existing Gauss code (procedure) under Ox in a similar way than one can call C (dll) or fortran codes from Gauss and Ox;
- (2) run a pure Gauss code.

Depending on the position and experience of the user, both use can prove to be particularly useful. From an Ox user point of view the main objective of OxGauss is probably to allow the many existing Gauss codes to be called from Ox with only a minimum of changes to these codes. This is beneficial for both the Ox user and the writer of the Gauss codes as it increases the visibility and potential use of the underlying statistical technique. This may also help in the transition from Gauss to Ox if this is the purpose of the exercise.

On the other hand, running a pure Gauss code with OxGauss is very attractive for the non-Gauss and potentially even non-Ox users in that it allows for example the replication of published work (theoretical MC simulations, empirical examples) using the free version of Ox. The following two subsections briefly summarize what we believe are the two main advantages of using OxGauss currently.

2.1 Understanding OxGauss

When an OxGauss program is run, it automatically includes the `\include\oxgauss.ox` file. This itself imports the required files:

```

#define OX_GAUSS                                     /include/oxgauss.ox
#import <g2ox>
#import <gauss::oxgauss>
```

These import statements lead to `g2ox.h` and `oxgauss.h` being included. The majority of the OxGauss run-time system is in `\include\g2ox.ox` while keywords are largely in `oxgauss.src`.

A nice feature of OxGauss is thus its transparency. Indeed, most of the codes that link Gauss functions to Ox are gathered in the file `\include\g2ox.ox`. As an illustration, suppose one wishes to use the Gauss function `seqa(const start, const inc, const m)` that creates an additive sequence. Recall that `start` is a scalar specifying the first element of the sequence, `inc` is a scalar specifying the increment and `n` is a scalar specifying the number of elements in the sequence. The output is a column vector containing the specified sequence.

A similar function is available in Ox: `range(const min, const max, const step)`, where `min` and `max` are integers or doubles specifying respectively the first and last values of the sequence, and `step` is an integer or double specifying the increment. The output is a row vector containing the specified sequence.

The Ox code here below (copy from the file `g2ox.ox`) shows how OxGauss makes the link between these two functions.

```

seqa(const start, const inc, const m) part of g2ox.ox
{
  decl st = start, in = inc, mm = m;
  if (::ismatrix(st)) st = double(st);
  if (::ismatrix(in)) in = double(in);
  if (::ismatrix(mm)) mm = int(mm);
  if (in == 0)
    return ::constant(st, mm, 1);
  else
    return st + ::range(0, in * (mm - 1), in)';
}

```

The version of OxGauss provided with Ox 3.3 covers a huge number of Gauss functions. Tables A1 and A2 (see the appendix) give a list of all the Gauss functions supported by OxGauss. To simplify the reading of the list, we report pre-compiled functions (or directly mapped functions) in Table A1 and open source functions (like the `seqa`, see above) in Table A2. Adding adding all the functions leads to a total of 420 functions recognized by OxGauss.

Note that not all the Gauss functions are available in OxGauss. Table A3 in the appendix gives a list of 64 Gauss functions not supported by the current version of Ox (or about 15%). Importantly, when OxGauss does not provide the Ox counterpart of a Gauss function, it prints “Function name”() unsupported.

For instance, there is not equivalent of the Gauss function `intgrat2` in Ox 3.3. For this reason, the corresponding procedure in `OxGauss.ox` just reports the error message `intgrat2() unsupported` (see below).

```

intgrat2(const f, const x1, const g1) part of g2ox.ox
{
  __printunsup__("intgrat2");
  return .NaN;
}

```

Hopefully, if such a function is available in a next version of Ox, mapping `ingrat2` to the corresponding function in Ox would be child’s play!

2.2 Calling Gauss codes from Ox

The main objective of OxGauss is probably to allow Gauss code to be called from Ox. This helps in the transition to Ox, and increases the amount of code that is available to users of Ox.

As an example we consider a small project that mixes both Gauss and Ox codes. The first file, *Gaussprocs.src*, consists of a code file containing the procedure *gengarch(omega,alpha,beta,nu,T_0,T,n)* that simulates a GARCH model. As most of the Gauss codes available on the internet, it is provided with a very detailed preamble that explains the meaning of its inputs and gives information concerning its output. This procedure has been written by Dick van Dijk (see Franses and van Dijk, 2000) and is downloadable from his web site <http://www.few.eur.nl/few/people/djvandijk/nltsmef/nltsmef.htm>. To call this procedure from an Ox code, one first has to create a header file. The second file, *Gaussprocs.h* is actually this header file and consists of the following instructions:

```

Gaussprocs.h
#include <oxstd.h>
namespace gauss
{
    gengarch(const omega,const alpha,const beta,const nu,const T_0,
             const T, const n);
    // Add new procedures here
}

```

The header file communicates the declaration of functions, constants and external variables.⁴ Additional procedures can be added in *Gaussprocs.src* but the header file has to be modified accordingly. It is recommended to use the *.src* extension for the Gauss programs and *.h* for the header files.

In the example *GarchEstim.ox*, we use the Gauss procedure to generate 20.000 observations following a GARCH(1,1) process with Student-t errors and then, rely on the Ox package G@RCH 3.0 (see Laurent and Peters, 2002) to estimate a GARCH(1,1) model by gaussian Quasi-Maximum likelihood. To do that, the OxGauss code must be imported into the Ox program, together with the G@RCH package. The `#import` command has been extended to recognize OxGauss imports by prefixing the file name with `gauss::`.

⁴Arguments declared `const` can be referenced, but cannot be changed inside the function.

```

GarchEstim.ox
#include <oxstd.h>
#import <packages/Garch30/garch>
#import "gauss::Gaussprocs"
main()
{
    decl omega=0.2; decl alpha=0.1; decl beta=0.8; decl nu=10;
    decl T_0=1000; decl T=20000; decl n=1;
    decl y=gauss::gengarch(omega,alpha,beta,nu,T_0,T,n);
    decl garchobj;
    garchobj = new Garch();
    garchobj.Create(1, 1, 1, T, 1);
    garchobj.Append(y, "Y");
    garchobj.Select(Y_VAR, "Y",0,0 );
    garchobj.SetSelSample(-1, 1, -1, 1);
    garchobj.DISTRI(0); //0 for Normal
    garchobj.GARCH_ORDERS(1,1); //p order, q order
    garchobj.MODEL(1); //1: GARCH
    garchobj.Initialization(<>);
    garchobj.DoEstimation();
    garchobj.Output();
    delete garchobj;
}

```

Note that when the OxGauss functions or variables are accessed, they must also be prefixed with the identifier `gauss::`.

To run this program on the command line, enter `oxl GarchEstim.ox`. Alternatively, it can be launched from OxEdit. Indeed, OxEdit 1.62 (or later) is a free but powerful text editor for Windows provided with both versions of Ox 3.3. Like GiveWin, OxEdit supports syntax colouring of Ox programs, and context-sensitive help. The first time you use OxEdit, execute the Preferences/Add Predefined Modules menu and select Ox. From then on you can run your Ox and Gauss programs from the Modules menu, without leaving OxEdit. See also the OxEdit web page <http://www.oxedit.com> for more details. Finally, users of the Ox Professional can run Ox codes within GiveWin by using the menu Modules/Start OxRun.

2.3 Running simple Gauss codes

Ox also has the capability of running a wide range of Gauss programs. As an example, let us consider a simple program that calls the Probability Distributions package *PROBS.SRC* (about 1300 lines of Gauss code), of David Baird available at the *Gauss source code archive at American University* (<http://gurukul.ucc.american.edu/econ/Gaussres/GaussIDX.htm>)

The Gauss code *baird.src* uses this procedure to compute and print the probability density function, cumulative density function and quantile function of various densities.

```
#include probs.src; baird.src

x1=invcdfn(0.1); x2=pdf(1,5); x3=cdf(1,5); x4=invcdf(0.1,5);
x5=pdf(0.1,2,3); x6=cdf(1,2,3); x7=invcdf(0.1,2,3);
x8=pdfchi(1,2); x9=cdfchi(1,2); x10=invcdfchi(0.1,2);
x11=pdfb(1,0.2,3); x12=cdfb(1,0.2,3); x13=cdfbc(1,0.2,3);
x14=invcdfb(0.8,0.1,1); x15=pdfp(1,2); x16=cdfp(1,2);
x17=cdfpc(1,2); x18=invcdfp(0.3,2);

print
x1|x2|x3|x4|x5|x6|x7|x8|x9|x10|x11|x12|x13|x14|x15|x16|x17|x18;
```

To run this program on the command line, enter `oxl -g Baird.src`. Alternatively, it can be launched from OxEdit (Modules/OxGauss menu) or within GiveWin by using the menu Modules/Start OxGauss.

Table 1 here below reports the output produced by OxGauss and Gauss. As expected, they are exactly the same.

Table 1 Outputs of *baird.src* produced by OxGauss and Gauss .

OxGauss	Gauss
-1.2815516	-1.2815516
0.21967980	0.21967980
0.81839127	0.81839127
-1.4758840	-1.4758840
0.85099732	0.85099732
0.53524200	0.53524200
0.10914897	0.10914897
0.30326533	0.30326533
0.39346934	0.39346934
0.21072103	0.21072103
0.38400000	0.38400000
0.89600000	0.89600000
0.10400000	0.10400000
0.00000000	0.00000000
0.27067057	0.27067057
0.40600585	0.40600585
0.59399415	0.59399415
1.0000000	1.0000000

2.4 Speed Comparison

The main strength of Ox is its speed (Cribari-Neto, 1997), but Gauss is not far away from Gauss. In a recent and very detailed comparison between Gauss, Macsyma, Maple, Mathematica, Matlab, MuPad,

O-Matrix, Ox, R-Lab, Scilab and S-Plus performed, Stefan Steinhaus ("Comparison of mathematical programs for data analysis", available at <http://www.scientificweb.de/ncrunch/>) shows that Ox is the winner in terms of speed.

Since OxGauss implements just a layer over Ox, OxGauss is expected to be slower than Ox. But how slower is it and how does it compared to Gauss ? To answer these two questions we first consider the Benchmark tests proposed by Stefan Steinhaus (edition 3). Note that since the functions *intquad2* and *intquad3* (double and triple integration) are not available in Ox, the corresponding tests have been discarded which leads a total of 14 points of comparison.

Table 2 Speed Comparison (timing in seconds) between Ox, Oxgauss and Gauss .

Operation	Ox 3.3	OxGauss	Gauss 3.5
Creation, trans. & reshaping of a 1000x1000 matrix:	1.043	1.153	1.043
1000x1000 random matrix to the power 1000:	1.023	1.003	1.083
Sorting of 2,000,000 random values:	9.190	9.577	9.643
FFT over 1,048,576 random values:	4.777	5.423	5.417
Determinant of a 1000x1000 random matrix:	14.590	14.593	14.593
Creation of an 1400x1400 Toeplitz matrix:	0.167	0.167	0.200
Inverse of a 1000x1000 random matrix:	36.433	36.600	36.930
Eigenvalues of a 600x600 random matrix:	35.573	35.867	36.563
Choleski decomposition of a 1000x1000 random matrix:	4.360	3.350	3.337
Creation of 1000x1000 cross-product matrix:	8.953	12.817	12.777
Calculation of 500000 fibonacci numbers:	1.377	1.360	1.423
Gamma function on a 1000x1000 random matrix:	0.737	0.763	0.730
Gaussian error function over a 1000x1000 random matrix:	0.930	0.750	0.790
Linear regression over a 1000x1000 random matrix:	28.563	28.597	28.713
Total of the 14 tests:	145.884	150.342	151.524

Benchmark programs run (3 replications of each test) on a Pentium III 450 Mhz.

As a whole, we see from this table that OxGauss compares very well to Gauss 3.5 in terms of speed. Indeed, while Ox is in general 4% faster than OxGauss and Gauss 3.5 (see the last row of the table for the total time spent to run the 14 tests), the difference between OxGauss and Gauss 3.5 is very small (OxGauss is found to be about 0.5% faster than Gauss 3.5).⁵

⁵While a comparison with a more recent version of Gauss is of potential interest, we do not investigate this issue since the speed comparison is not the aim of the paper.

3 Graphs using GnuDraw

While most graphical features of Gauss are recognized by OxGauss via Ox Professional, Oxconsole has no graphs support.

To fix this problem, Charles Bos proposes GnuDraw, an Ox package that allows the creation of GnuPlot (see <http://www.gnuplot.info>) graphics from Ox. The package is free of charge and is downloadable from his homepage <http://www.tinbergen.nl/~cbos/>, together with the GnuPlot software.

When using Ox 3.30, GnuPlot can be called automatically from within Ox. Usage of GnuDraw is intended to be simple. See Cribari-Neto and Zarkos (2003) for a comprehensive overview of the GnuDraw package.

Interestingly, as OxGauss implements just a layer over Ox, it is possible to instruct the underlying Gauss to call GnuDraw routines instead of the OxDraw routines. The program `gnuGauss.prg` implements this.

```

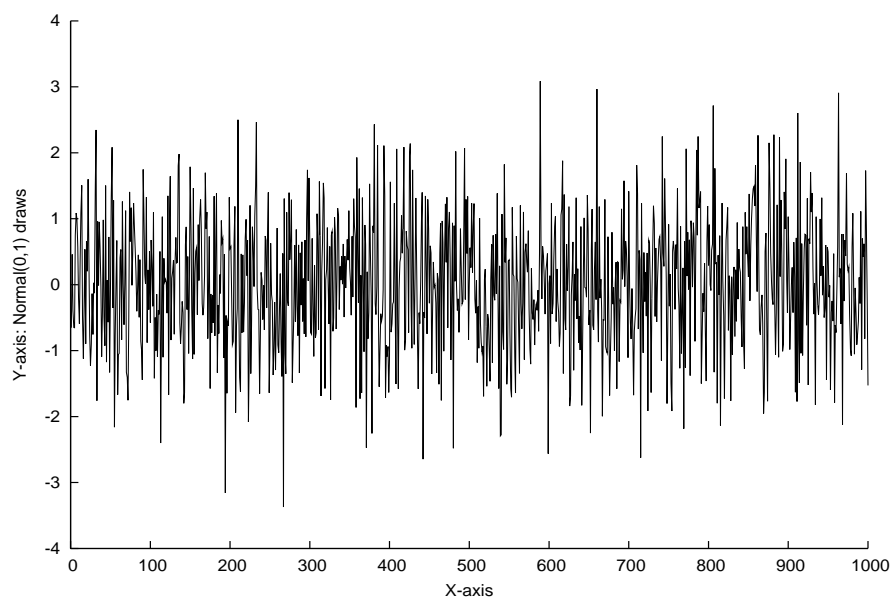
library pgraph_gnu;
x=seqa(1,1,1000);
y=rndn(1000,1);
xlabel("X-axis");
ylabel("Y-axis: Normal(0,1) draws");
_ptek="gnugauss.plb";
call xy(x, y);
end;

```

gnuGauss.prg

The Gauss library **pgraph** has to be replaced by **pgraph_gnu**. Note that exactly the same functionality is implemented as with OxGauss itself. A difference with (the present version of) the OxGauss version of pgraph is that it is now possible to save the graph using the global variable **_ptek**.

Figure 1 shows the graph obtained by running *gnugauss.prg* with OxGauss.



4 Some larger projects using M@ximize

The main drawback of OxGauss is that it is not suited to run Gauss codes that make use of commercial packages. For instance OxGauss reports an error message if the Gauss code requires one of the well known optimizers Cml, Maxlik, or Optmum. This makes OxGauss useless in many situations.

To overcome this problem, we propose a set of three procedures that make the link between the optimizers of Ox 3.3 and the optimizers of Gauss.⁶ The package, called M@ximize 1.0, is open source and freely available on the web at the following address: <http://www.core.ucl.ac.be/~laurent>. To install the package, unzip the main file *into* the Ox directory.

The objective of this section is to give several concrete examples, and address some of the issues that can be encountered. As explained above, the main reason for using OxGauss is probably to replicate the results obtained in a research paper. To test OxGauss and M@ximize 1.0 in a real-life situation, we have downloaded from the internet a huge number of Gauss codes.⁷ Here is a list of five web sites that we have visited:

James Hamilton: <http://weber.ucsd.edu/~jhamilto/>

Rolf Tschernig: <http://www.personeel.unimaas.nl/r.tschernig>

Bruce Hansen: <http://www.ssc.wisc.edu/~bhansen>

Chang-Jin Kim: <http://www.econ.washington.edu/user/cnelson/SSMARKOV.htm>

Luc Bauwens: <http://www.core.ucl.ac.be/econometrics/bauwens.htm>

From these web sites we have found both the data and the Gauss codes used in a collection of articles. In addition, we have used the codes provided by Kim and Nelson (1999) in their book on markov-switching models (chapters 3 to 11). Table 3 gives the list of papers we have replicated. Most of them rely on non-linear optimization technique and thus require one the three above mentioned optimizers of Gauss.

While most of the codes can be run in their present form some changes are sometimes needed. Here is a list of the most frequently encountered problems:

- *Convert data files.* For instance, when running example 22 in Table 2 gives the Invalid .FMT or .DAT file error message. The reason is that old style data sets (v89 *.dht/.dat*) must be converted to the new format (v96 *.dat*). The program to do this conversion is `ox/lib/dht2dat`. The conversion can be run from the command line as:

⁶Note that we do not translate the Gauss optimizers into Ox but we just link the options

⁷All the Gauss codes we have used are available from the M@ximize web site (see above).

Table 3 List of codes associated to papers .

1. HAMILTON, J. (1994): *State-Space Models*, in Handbook of Econometrics, Volume 4, 3039–3080, edited by R.F. Engle and D., McFadden, Amsterdam: North Holland.
2. HAMILTON, J. (1996): “The Daily Market for Federal Funds”, *Journal of Political Economy*, pp. 26–56.
3. HAMILTON, J. (1996): “Specification Testing in Markov-Switching Time-Series Models”, *Journal of Econometrics*, 70, 127–157.
4. HAMILTON, J., and C. ENGLE (1990): “Long Swings in the Exchange Rate: Are They in the Data and Do Markets Know It?”, *American Economic Review*, pp. 689–713.
5. HAMILTON, J., and O. JORDA (2002): “A Model for the Federal Funds Rate Target”, *Journal of Political Economy*, 110, 1135–1167.
6. HAMILTON, J., and G. LIN (1996): “Stock Market Volatility and the Business Cycle”, *Journal of Applied Econometrics*, 11, 573–593.
7. HAMILTON, J., and G. PEREZ-QUIROS (1996): “What Do the Leading Indicators Lead?”, *Journal of Business*, 69, 27–49.
8. HAMILTON, J., and R. SUSMEL (1994): “Autoregressive Conditional Heteroskedasticity and Changes in Regime”, *Journal of Econometrics*, 64, 307–333.
9. Yang, L. and R. Tschernig (1999): “Multivariate Bandwidth Selection for Local Linear Regression”, *Journal of the Royal Statistical Society, Series B*, 61, 793-815.
10. Hansen, B. (1992): “Tests for Parameter Instability in Regressions with I(1) Processes”, *Journal of Business and Economic Statistics*, 10, 321–335.
11. Hansen, B. (1992): “Testing for Parameter Instability in Linear Models”, *Journal of Policy Modeling*, 14, 517–533.
12. Hansen, B. (1992): “The likelihood Ratio Test under Non-standard Conditions: Testing the Markov Switching Model of GNP”, *Journal of Applied Econometrics*, 7, S61-S82.
13. Hansen, B. (1994): “Autoregressive Conditional Density Estimation”, *International Economic Review*, 35, 705-730.
14. Hansen, B. (1996): “Inference when a Nuisance Parameter is not Identified under the Null Hypothesis”, *Econometrica*, 64, 413-430.
15. Hansen, B. and A. Gregory (1996): “Residual-based Tests for Cointegration in Models with Regime Shifts”, *Journal of Econometrics*, 70, 99-126.
16. Hansen, B. (1997): “Approximate Asymptotic p-values for Structural Change Tests”, *Journal of Business and Economic Statistics*, 15, 60-67.
17. Hansen, B. (1997): “Inference in TAR Models”, *Studies in Nonlinear Dynamics and Econometrics*, 2, 1-14.
18. Hansen, B. (1999): “Testing for Linearity”, *Journal of Economic Surveys*, 13, 551-576.
19. Hansen, B. (2000): “Sample Splitting and Threshold Estimation”, *Econometrica*, 68, 575-603.
20. Hansen, B. (2000): “Testing for Structural Change in Conditional Models”, *Journal of Econometrics*, 97, 93-115.
21. Hansen, B. and M. Caner (2000): “Threshold Autoregression with a Unit Root”, *Econometrica*, 69, 1555-1596.
22. Hansen, B., D. Cox and E. Jimenez: “How Responsive are Private Transfers to Income? Evidence from a Laissez-faire Economy”, forthcoming in *Journal of the Public Economics*.
23. Hansen, B. and B. Seo (2002): “Testing for Threshold Cointegration”, *Journal of Econometrics*, 110, 293-318.
24. Hansen, B. (2001): “The New Econometrics of Structural Change: Dating Changes in U.S. Labor Productivity”, *Journal of Economic Perspectives*, 15, 117-128.
25. Hansen, B.: “Recounts from Undervotes: Evidence from the 2000 Presidential Election”, forthcoming in *Journal of the American Statistical Association*.
26. Kim, C.-J. and C. Nelson (1999): *State-Space Models with Regime Switching: Classical and Gibbs-Sampling Approaches with Applications*, The MIT Press.
27. Bauwens, L. M. Lubrano (1998): Bayesian Inference on GARCH models using the Gibbs Sampler, *The Econometrics Journal*, 1, C23-C46.

```
oxl lib/dht2dat old_datafile.dht new_datafile.dat
```

Alternatively, the data files can be converted to the new format through GiveWin by loading first the *.dht* file and second saving the file into the new format.

- *No extension.* To launch a Gauss code using OxEdit, the file needs an extension. It is common to use the extension *.src*.
- *Interactive mode.* Examples 1, 6, 13 and 27 use the Gauss function **cons**. Then the user received a message like "Do you wish to continue (y or n)?". Depending of the result the program takes one direction or the another. In other words, the program enters in an interactive mode. In such a case the program has to be launched using "Ox interactive", i.e. Oxli.exe under Windows instead of Oxl.exe.⁸

As a way of illustration, let us consider the Gauss code written by Bruce Hansen for his *Journal of the American Statistical Association* paper "Recounts from Undervotes: Evidence from the 2000 Presidential Election" (number 25 in Table 3).

Let us consider one Gauss code, i.e. *Absentee.prg*. The code is made of 468 lines of code (including the 200 lines of the file *beta.prc*). The program uses 2 libraries: *optmum* and *pgraph*. The code is given below.

⁸When using OxEdit to run the Gauss code, an additional shortcut has to be created. To do that, click on the menu VIEW/PREFERENCES/ADD/REMOVE MODULES. Then clone the OxGauss shortcut and in the Command line change Oxl.exe by Oxli.exe.

Absentee.prg

```

library optmum,pgraph;
/*****
ABSENTEE.PRG

This is a Gauss program.
It replicates the empirical work reported in my paper
"Recounts from Undervotes: Evidence from the 2001 Presidential Election"
September, 2001
Revised: April, 2002

The program calls the companion file "beta.prc",
which should be in the same directory. It loads the data file
"absentee.txt", and creates an output file "absentee.out".

This program could be modified to estimate datasets using the
multivariate BetaLogit likelihood.

Bruce Hansen
bhansen@ssc.wisc.edu
http://www.ssc.wisc.edu/~bhansen/

*****/
cls;
#include beta.prc;
output file=absentee.out reset;
"*****";
"";
output off;
/*****Data*****/
n=282;
load dat[n,6]=absentee.txt;
/*****/
pn=2;           @ number of models @
y1=dat[.,3 4];   @ first model - dependent variables @
y2=dat[.,5 6];
y2=y2.*(y2.>0);
ballots=y1+y2;
title1="Gore Percentage in Machine-Readable Vote";
title2="Gore Percentage in Recount";
xc=ones(n,1);x=xc;
let xcnames = "Const" ;xcnames=xcnames;pnames=xcnames;

let xselect1 = 0;
let xselect2 = 0;

let pselect1 = 0 ;
let pselect2 = 0 ;

let ipselect1 = 0;
let ipselect2 = 1;
/*****/
@ Estimation @
optset;
_opgdprc=&G_Like_Beta;
_ophsprc=&H_Like_Beta;
_opalgr=5;
_opshess=1;

```

```

p=zeros(n,pn);
e=zeros(n,pn);
rt=zeros(pn,1);
j=1; do while j<=pn;
    nj = ftocv(j,1,0);
    _y=varget("y" $+ nj);
    _w=((_y[.,1]+_y[.,2]).>0);
    xselect=varget("xselect" $+ nj);
    if sumc(xselect)>0; xs=x[.,xselect]; else; xs=0; endif;
    _x=xp(xs,p,j);
    phat=sumc(_y[.,1].*_w)/sumc(sumc(_y.*_w));
    theta0=zeros(cols(_x),1)|(1/phat/4);
    theta0[1]=ilogit(phat);
    theta,f,gr,ret=optmum(&Like_Beta,theta0);
    g=V_G_Like_Beta(theta,_y,_x);
    gr=sumc(g)';
    h=H_Like_Beta(theta);
    p[.,j] = pr_hat(_y,_x,theta);
    e[.,j]=(_y[.,1]/(_y[.,1]+_y[.,2]+(1-_w))
        -logit(_x*theta[1:cols(_x)]))*_w;
    sj = varput(phat,"phat" $+ nj);
    sj = varput(_x,"x" $+ nj);
    sj = varput(xs,"xs" $+ nj);
    sj = varput(_w,"_w" $+ nj);
    sj = varput(theta,"theta" $+ nj);
    sj = varput(f,"f" $+ nj);
    sj = varput(g,"g" $+ nj);
    sj = varput(gr,"gr" $+ nj);
    sj = varput(h,"h" $+ nj);
    rt[j]=rows(theta);
j=j+1;endo;
/*****/
@ Covariance Matrix @
r=sumc(rt);
h=zeros(r,r);
g=varget("g" $+ ftocv(1,1,0));
crt=0|cumsumc(rt);
j1=1; do while j1<=pn;
    rj1=rt[j1];
    nj = ftocv(j1,1,0);
    if j1>1; g=g~varget("g" $+ nj); endif;
    h[crt[j1]+1:crt[j1]+rj1,crt[j1]+1:crt[j1]+rj1]=varget("h" $+ nj);
    theta=varget("theta" $+ nj);
    j2=1; do while j2<=rj1;
        thetaj2=theta[j2];
        ax=abs(thetaj2);
        delta=((ax.>1).*ax + (ax.<=1)).*(1e-6);
        theta[j2]=thetaj2+delta;
        pj=p;
        xs=xp(varget("xs" $+ nj),pj,j1);
        pj[.,j1] = pr_hat(varget("y" $+ nj),xs,theta);
        j3=(j1+1); do while j3<=pn;
            nj3 = ftocv(j3,1,0);
            y=varget("y" $+ nj3);
            xs=xp(varget("xs" $+ nj3),pj,j3);
            _w=varget("_w" $+ nj3);
            thetaj3=varget("theta" $+ nj3);
            pj[.,j3] = pr_hat(y,xs,thetaj3);
            h[crt[j1]+j2,crt[j3]+1:crt[j3+1]]=
                (sumc(V_G_Like_Beta(thetaj3,y,xs))'-varget("gr" $+ nj3))./delta;
            j3=j3+1;endo;

```

```

theta[j2]=thetaj2;
  j2=j2+1;endo;
j1=j1+1;endo;
v=inv(h);
v=v'moment(g,0)*v;
/*****
@ Print Results @
format 16,8;
let fmt1[1,3]="*.*1f" 14 8;
let fmt2[1,3]="*.*1f" 14 6;
let fmt3[1,3]="*.*1f" 8 3;
let fmt4[1,3]="*.*1f" 8 8;
output on;
rr=0;
j=1; do while j<=pn;
  nj = ftocv(j,1,0);
  tit=varget("title" $+ nj);
  y=varget("y" $+ nj);
  _w=varget("_w" $+ nj);
  theta=varget("theta" $+ nj);
  rj=rows(theta);
  ij=seqa(rr+1,1,rj);
  vj=v[ij,ij];
  sj = varput(vj,"v" $+ nj);
  rr=rr+rj;
  xselect=varget("xselect" $+ nj);
  pselect=varget("pselect" $+ nj);
  ipselect=varget("ipselect" $+ nj);
  names=xcnames;
  if sumc(xselect)>0;
    names=names|xnames[xselect];
  endif;
  if sumc(pselect)>0;
    names=names|pnames[pselect];
  endif;
  if ipselect==1;
    names=names|xcnames;
  endif;
  s=theta[rj];
  phat=varget("phat" $+ nj);
  sig=sqrt(phat.*(1-phat)./(1+s));
  th=theta|sig;
  se=sqrt(diag(vj));
  se=se|(se[rj]*sig/(2*(1+s)));
  $tit;
  "Number of Precincts " sumc(_w);
  "Totals " sumc(y)';
  "Percentage " phat;
  "";
  pr=printfm("Variable"~"Estimate"~"St Error",0~0~0,fmt1);"";
  pr=printfm((names|"Scale"|"SD")~th~se,0~1~1,fmt2); "";
  "Negative Log Likelihood " varget("f" $+ nj);
  "";"";
  "*****";
j=j+1;endo;
output off;

/*****
@ Plot Graphs @
npoints=199;
xpoints=seqa(1/(npoints+1),1/(npoints+1),npoints);

```

```

graphset;
_pcolor=15;
_pdate="";
_plwidth=3;
j1=1; do while j1<=pn;
  nj1 = ftocv(j1,1,0);
  ipselect=varget("ipselect" $+ nj1);
  if sumc(ipselect) /= 0;
    pselect=varget("pselect" $+ nj1);
    theta=varget("theta" $+ nj1);
    _w=varget("_w" $+ nj1);
    xselect=varget("xselect" $+ nj1);
    p1=ilogit(xpoints);
    if pselect /= 0;
      p2=sumc(ln(p[.,pselect]).*_w.*ballots)./sumc(_w.*ballots);
    endif;
    v=varget("v" $+ nj1);
    v=v[1:rows(v)-1,1:rows(v)-1];
    tt= 1;
    tn=1;
    z1=ones(npoints,1);
    if sumc(xselect)>0;
      cx=rows(xselect);
      tt = tt|(sega(tn+1,1,cx));
      tn=tn+cx;
      z1=z1~ones(npoints,cx);
    endif;
    if sumc(pselect)>0;
      tt = tt|(sega(tn+1,1,rows(pselect)));
      tn=tn+rows(pselect);
      z1=z1~(ones(npoints,1)*(p2'));
    endif;
    tt=tt|(tn+1);
    thetal=theta[tt[.,1]];
    v1=v[tt[.,1],tt[.,1]];
    z1=z1~p1;
    zt=z1';
    mul=logit(z1*thetal);
    sel=sqrt(mul.*(1-mul).*sumc((v1*zt).*zt));
    a1=mul-sel*1.96;    b1=mul+sel*1.96;
    let _pltype = 6 3 3 4;
    xlabel("p");
    ylabel("E(q|p)");

    ytics(0,1,.1,10);
    tit="" $+ "Figure 4L" $+ varget("title" $+ nj1)
    $+ "Absentee Precincts";
    title(tit);
    xy(xpoints,mul~a1~b1~xpoints);
  endif;
j1=j1+1;endo;

```

```

/*****
@ Procedures @
proc xp(xs,p,j);          @ generates covariate X matrix @
local pselect,ipselect,xx,xselect;
  pselect=varget("pselect" $+ ftocv(j,1,0));
  xselect=varget("xselect" $+ ftocv(j,1,0));
  ipselect=varget("ipselect" $+ ftocv(j,1,0));
  xx=xc;
  if sumc(xselect)>0;
    xx=xx~xs;
  endif;
  if sumc(pselect)>0;
    xx=xx~ln(p[.,pselect]);
  endif;
  if ipselect==1;
    xx=xx~(ilogit(p[.,1]).*xc);
  endif;
retp(xx);
endp;

fn pr_hat(y,x,theta)=(logit(x*theta[1:cols(x)]).*theta[rows(theta)]
                    +y[.,1])./(y[.,1]+y[.,2]+theta[rows(theta)]);
fn expt(x)=exp(x.*(x.>(-700)).*(x.<700)-(x.<=(-700))*700+(x.>700)*700);
fn logit(u)=1./(1+exp(-u));
fn ilogit(y)=-ln((1./y)-1);

```

Here is the output obtained with OxGauss under GiveWin:

Ox version 3.30 (Windows) (C) J.A. Doornik, 1994-2003

Value of objective function: 322.30298

Strong convergence using analytical first derivatives and hessian

Gore Percentage in Machine-Readable Vote

Number of Precincts	238.00000000	
Totals	40696.00000000	52146.00000000
Percentage	0.43833610	

Variable	Estimate	St Error
Const	-0.14079028	0.04187395
Scale	12.19074575	1.41620233
SD	0.13661777	0.00733387

Negative Log Likelihood 59221.49838094

Gore Percentage in Recount

Number of Precincts	90.00000000	
Totals	222.00000000	296.00000000
Percentage	0.42857143	

Variable	Estimate	St Error
Const	0.07851050	0.10908096

Const	1.05415020	0.20402046
Scale	88.30611647	142.66769468
SD	0.05236631	0.04182794

Negative Log Likelihood 322.30297854

Here is the output obtained with Gauss 3.5:

Gore Percentage in Machine-Readable Vote

Number of Precincts	238.00000	
Totals	40696.000	52146.000
Percentage	0.43833610	

Variable	Estimate	St Error
Const	-0.140782	0.041872
Scale	12.179387	1.413274
SD	0.136677	0.007328

Negative Log Likelihood 59221.498

Gore Percentage in Recount

Number of Precincts	90.000000	
Totals	222.00000	296.00000
Percentage	0.42857143	

Variable	Estimate	St Error
Const	0.078522	0.109059
Const	1.054151	0.203970
Scale	88.184856	141.949558
SD	0.052402	0.041702

Negative Log Likelihood 322.30275

Figures 2 and 3 show respectively the graphs obtained by running *Absentee.prg* with OxGauss under GiveWin and Gauss 3.5.

5 Conclusion

This paper presents a review and a discussion of OxGauss, an application that enables the user to run a wide range of Gauss programs/codes under Ox without needing to have Gauss installed on his/her machine. One main drawback of OxGauss is that it is of little use once the purpose is to execute a program that requires one of the three well know Gauss application modules Cml, Maxlik or Optmum. In this paper we propose a set of additional procedures that contribute to bridge the gap between Ox and the above mentioned optimizers. The effectiveness of the procedures is illustrated by

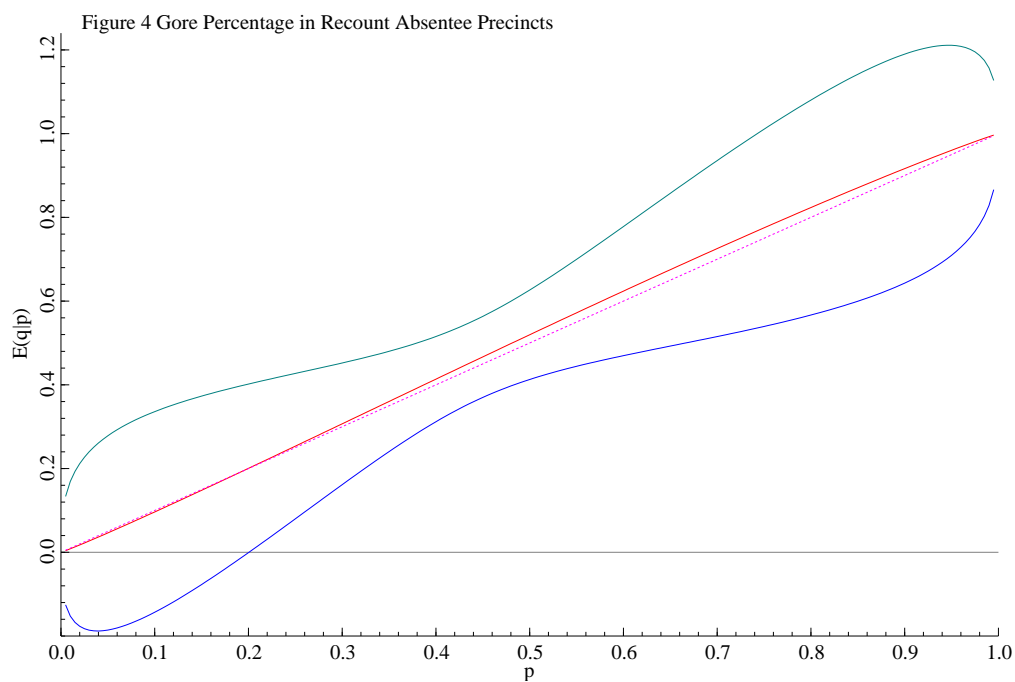


Figure 2 Figure produced by the GAUSS code *Absentee.prg* with OxGauss under GiveWin.

revisiting a large number of Gauss codes that are freely available on the internet and that use Gauss application modules that require numerical optimization. We revisit no less than 26 papers published in international reviews and show that the results can be replicated using the original codes and data provided by the authors.

Figure 4
Gore Percentage in Recount
Absentee Precincts

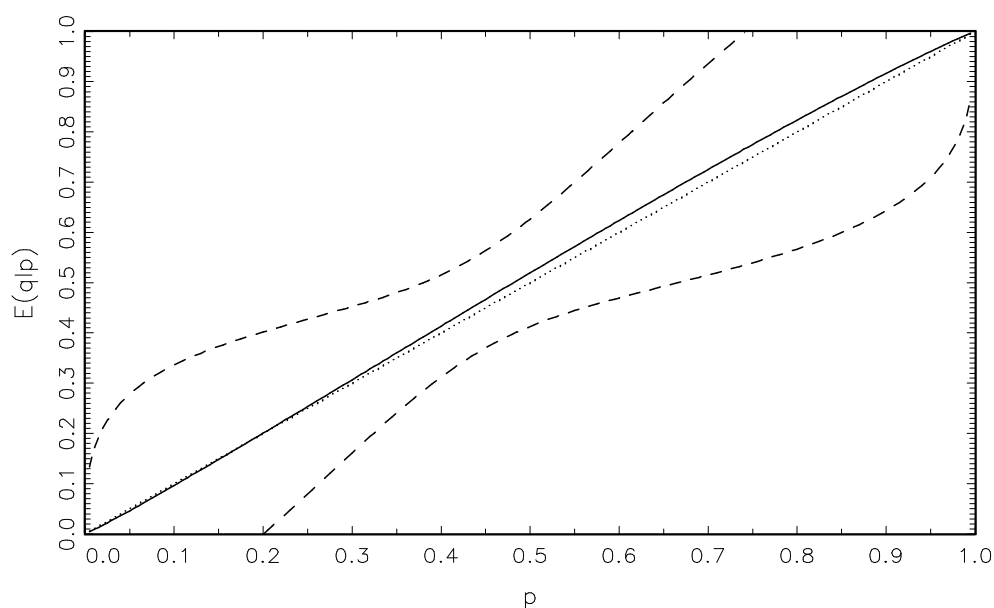


Figure 3 Figure produced by the GAUSS code *Absentee.prg* under Gauss 3.5.

References

- CRIBARI-NETO, F. (1997): "Econometric Programming Environments: Gauss, Ox and S-Plus," *Journal of Applied Econometrics*, 12, 77–89.
- CRIBARI-NETO, F., and S. ZARKOS (2003): "Econometric and Statistical Computing Using Ox," *Computational Economics*, 21, 277–295.
- FRANSES, P., and D. VAN DIJK (2000): *Non-Linear Series Models in Empirical Finance*. Cambridge University Press.
- KIM, C.-J., and C. NELSON (1999): *State-Space Models with Regime Switching: Classical and Gibbs-Sampling Approaches with Applications*. The MIT Press.
- LAURENT, S., and J.-P. PETERS (2002): "G@RCH 2.2 : An Ox Package for Estimating and Forecasting Various ARCH Models," *Journal of Economic Surveys*, 16, 447–485.

Table A1: Precompiled functions supported by OxGauss

_fcmptol	delete	lib library	output	save
abs	det	library	outwidth n	saveall
arccos	diagrv	line	pdfn	screen
arcsin	disable	ln	plot x,y	scroll
arctan	dlibrary	load x	plotsym n	shell
arctan2	dllcall	loadf f	pqqwin	show
atan	enable	loadk k	presn n	sin
atan2	end/stop	loadm x	print	sinh
cdfchic	erf	loadp p	printdos str	sqrt
cdfchii	errorlog str	loads s	rank	system
cdffc	exp	locate m,n	replay	tan
cdfn	eye	log	rerun	tanh
cdfnc	feq	lowmat	rev	toeplitz
cdfni	floor	lprint	rndcon c	trace new
cdftc	fmod	lpwidth n	rndmod m	trap new
ceil	format	lshow	rndmult a	trunc
cols	gamma	meanc	rndn	use gcgfile
cos	graph	median	rndseed seed	vcx
cosh	hsec	msym str	rndu	vech
create	inv	new	round	xpnd
datalist	invpd	ones	rows	zeros
debug	ismiss	open	run filename	

STEINHAUS, S. (2003): "Comparison of Mathematical Programs for Data Analysis (4rd ed),"

<http://www.scientificweb.de/ncrunch/>.

VITON, P. (2001): "Running GAUSS programs Under Ox3," [http://facweb.arch.ohio-](http://facweb.arch.ohio-state.edu/pviton/support/oxgauss)

[state.edu/pviton/support/oxgauss](http://facweb.arch.ohio-state.edu/pviton/support/oxgauss).

Table A2: Open source functions supported by OxGauss

balance	corrvc	etstr	keyw	polymult	seekr	upper
band	corrxx	exctsmpl	lag1	polyroot	selif	utrisol
bandchol	counts	exec	lagn	printfm	seqa	vals
bandcholsol	countwts	export	lncdfbvn	printfmt	seqm	vcm
bandltsol	crossprd	exportf	lncdfn	prodc	setcnvrt	vec
bandrv	crout	fcheckerr	lncdfn2	putf	setdif	vecr
bandsolpd	croutp	fclearerr	lncdfnc	qnewton	setvmode	vget
base10	csrtype	fflush	lnfact	qprog	shiftr	wait
besselj	cumprodc	fft	lnpdfmvn	qqr	sleep	waitc
bessely	cumsumc	ffti	lnpdfn	qqre	solpd	writer
cdfbeta	cvtos	fftn	loadd	qqrep	sortc	xpnd
cdfbvn	date	fge	lower	qr	sortcc	
cdfchinc	datestr	fgets	lowmat1	qre	sorthc	
cdffnc	datestring	fgetsa	ltrisol	qrep	sorthcc	
cdfgam	datestrymd	fgetsat	lu	qrsol	sortind	
cdfn2	dayinyr	fgetst	lusol	qrtsol	sortindc	
cdftci	delif	fgt	maxc	qtyr	sortmc	
cdftnc	design	files	maxindc	qtyre	sqpsolve	
cdftvn	detl	fle	maxvec	qtyrep	stof	
cdir	dfft	flt	mbesselei	quantile	stop	
changedir	dffti	fne	mbesselei0	qyr	strindx	
chol	dfree	fopen	mbesselei1	qyre	strlen	
choldn	diag	formatcv	mbesseli	qyrep	strput	
cholsol	dos	formatnv	mbesseli0	rankindx	strrindx	
cholup	dotfeq	fputs	mbesseli1	readr	strsect	
chrs	dotfge	fputst	meanc	real	submat	
close	dotfgt	fseek	minc	recode	subscat	
closeall	dotfle	fstrerror	minindc	recserar	substute	
cls	dotflt	ftell	miss	recsercp	sumc	
cmadd	dotfne	ftocv	missex	recserrc	svd	
cmcplx	dstat	ftos	missrv	rfft	svd1	
cmcplx2	dummy	gammaii	moment	rffti	svd2	
cmdiv	dummybr	gausset	ndpchk	rfftip	svdcusv	
cmemult	dummydn	getf	ndpclex	rfftn	svds	
cmimag	eig	getname	ndpcntrl	rfftnp	svdusv	
cminv	eigh	gradp	null1	rfftp	sysstate	
cmmult	eighv	hasimag	ols	rndbeta	system	
cmreal	eigr	hessp	olsqr	rndgam	tab	
cmsoln	eigr2	imag	olsqr2	rndnb	tempname	
cmsub	eigrs	import	orth	rndns	time	
cmtrans	eigrs2	indcv	packr	rndp	timestr	
code	eigv	indexcat	parse	rndus	token	
color	end	indices	pause	rndvm	trapchk	
colsf	envget	indices2	pi	rotater	trim	
con	eof	indnv	pinv	rowsf	trimr	
cond	eqsolve	intrsect	polychar	rref	type	
cons	erfc	invswp	polyeval	save	union	
conv	error	iscplx	polyint	saved	uniqindx	
coreleft	etdays	iscplx2	polymake	scalerr	unique	
corrmm	ethsec	key	polymat	scalmiss	upmat	

Table A3: Functions not supported by OxGauss (under Ox 3.3)

cdfbvn2	filesa	makevars	sortd	vnamecv
cdfbvn2e	getnr	medit	spline1d	vput
cdfmvn	getpath	mergeby	spline2d	vread
complex	header	mergevar	stdc	vtypecv
conj	hess	momentd	tocart	
csrcol	importf	nametype	topolar	
csrlin	intgrat2	nextn	typecv	
editm	intgrat3	nextnevn	typef	
eigcg	intquad1	null	varget	
eigcg2	intquad2	optn	vargetl	
eigch	intquad3	optnevn	varput	
eigch2	intrleav	quantiled	varputl	
fftm	intsimp	schtoc	vartype	
fftmi	lncdfbvn2	schur	vartypef	
fileinfo	lncdfmvn	setvars	vlist	