Bridging the Gap Between Ox and Gauss using OxGauss

Sébastien Laurent[1] and Jean-Pierre Urbain[2]

October 2003

Abstract

The purpose of this paper is to review and discuss the key improvements brought to OxGauss. Without having to install Gauss on his or her machine, the OxGauss user can run under Ox a wide range of Gauss programs and codes. Even with the *console* Ox version (free for academics), Gauss codes can either be called from Ox programs or run and executed on their own. While the new OxGauss version is very powerful in most circumstances, it is of little use once the purpose is to execute programs that attempt to solve optimization problems using Cml, Maxlik or Optmum. In this paper we propose a set of additional procedures that contribute to bridge the gap between Ox and three well-known Gauss application modules: Cml, Maxlik or Optmum.

The effectiveness of our procedures is illustrated by revisiting a large number of freely available Gauss codes in which numerical optimization relies on the above Gauss application modules. The Gauss codes include many programs dealing with non-linear models such as the Markov regime-switching models 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.

[1]CeReFim (Université de Namur) and CORE (Université catholique de Louvain).
E-mail: Sebastien.Laurent@fundp.ac.be. Correspondence to CeReFim, 8 rempart de la vièrge, B5000 Namur, Belgium. Phone: +32 (0) 81 724869. Fax: +32 (0) 81 724840.

[2]Department 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

Important new technical advances have been made in theoretical econometrics over the past few years. Due to the ever increasing use of computer intensive modelling techniques and estimation methods, applied econometricians are more and more facing the dire need to write programs with the aim of implementing these recently developed techniques. While some pre-packaged routines are often made available in standard econometric software like RATS or TSP, most of the latest technical advances are not so rapidly introduced in these softwares. For example, the estimation of various non-linear time series models requires high computing skills from the researcher. Real world applications of computer intensive techniques like bootstrap inferences, indirect inferences, non-parametric and semi-parametric analyzes also require efficient programming. In all theses cases, researchers will have to put their shoulders to the wheel. As Cribari-Neto and Zarkos (2003) pointed out in their review of Ox, programming forces researchers to think more deeply into the issue that is being investigated. However, it may be rather inefficient to programme complex procedures or estimation techniques that have already been programmed, used and checked by other researchers for the sole purpose of applying or replicating the 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, often with the aim of 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, Gauss[1] has probably been one of the most, if not the most, popular programming environments in econometrics for the last two decades, along with S-Plus, MATLAB and now Ox.[2,3] The many codes available through several

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

[2]There 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*).

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

Gauss archives give the applied researcher a unique and important opportunity to implement procedures that are otherwise often demanding to programme. However, the researcher who is willing to replicate or to apply these freely available techniques should already be a Gauss user or should at least have a direct access to a legally registered version of Gauss. Moreover, the price of many software packages (including Gauss) has increased substantially over the last few years, making the acquisition of these packages very costly to small research groups or, more importantly, to researchers located in third world countries where financial resources are rather scare.

Besides the whole GNU project, several statisticians and econometricians have made a number of softwares freely available, at least to academics and students. These include Easyreg, ECTS, Vista (Visual Statistics System), GRETEL (Gnu Regression, Econometrics and Time-series Library), or some low-cost alternatives or clowns of Matlab (O-Matrix, Octave) and S-Plus (R). Although these softwares are very useful for empirical research given the wide range of tools offered, they still do not give the opportunity to benefit from the impressive Gauss code archive.

For all these reasons, OxGauss fulfils a genuine need in that it provides the researcher with a free and rather simple solution to run Gauss codes. The advantage of OxGauss is that, even with the *console* version (free for academics), Gauss codes can be called from Ox programs, or can be run and executed on their own. OxGauss has in this sense similar possibilities than O-matrix that can run most Matlab codes; but the crucial difference is that OxGauss comes freely with the console version of Ox.

While the new version of OxGauss (provided with Ox 3.3) is in many situations very powerful, it becomes of little use at least in two situations. First, it is important at this stage to emphasize that, in terms of compatibility, OxGauss is currently not designed to run (or call) Gauss procedures making use of new features specific to Gauss 4.0 and 5.0. Indeed, OxGauss only supports Gauss versions 3.2 and 3.5. Second, OxGauss is useless 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 convertable to OxGauss. As we've seen, one important case is when it uses Gauss's Maxlik procedure; or more generally, other extra-cost Gauss add-ons. Even in the Maxlik case it may be possible to substitute Ox's own optimization routine; if anyone knows how to do this, I'd 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 map Maxlik and its output generally to domax. Again, if anyone does this and would like to explain it, I'd be glad to include the information here."*

This paper proposes to fix this second limitation of OxGauss. In particular, we propose a set of additional procedures (gathered in the M@ximize package) that contribute to bridge the gap between Ox and the above mentioned optimizers. The effectiveness of the procedures is illustrated by revisiting various 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, STAR models and various GARCH-type models. In order to be transparent, all the Ox and Gauss codes used in the applications are available on our web site (see below). 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.

The 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 optimizators. Finally, Section 5 concludes.

2 OxGauss

OxGauss is an integral part of Ox 3.3. It is 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 goal of the analysis and on the 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 number of changes to these codes. This is beneficial to both the Ox user and the writer of the Gauss codes as it increases the visibility and hence the 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 attractive for the non-Gauss and potentially even non-Ox users in that it makes the replication of published work (theoretical MC simulations, empirical examples) possible 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 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.

In order to illustrate how Gauss codes can be called from OX, 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 purpose of the header file is simply to allow the declaration of the functions, constants and external variables to be known wherever it is required. The reason is that, to avoid a compilation error in Ox, any function or global variable has to be explicitly declared before its use. In our example, the header file (*Gaussprocs.h*) 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
}
```

Additional procedures can be added in *Gaussprocs.src* but the header file has to be modified accordingly.[4] 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 so, 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::`.

[4] Arguments 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, the user simply has to enter `oxl GarchEstim.ox`. Alternatively, it can be launched from OxEdit. 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 OxEdit is used, the user should execute the `Preferences/Add Predefined Modules` menu and select Ox. Ox and Gauss programs can then be run 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`.

2.2 Running Gauss codes

The second potentially attractive feature of OxGauss is to enable the user to directly run a wide range of Gauss programs under Ox.

As an example, we consider the Gauss package *Mixed Logit Estimation Routine for Panel Data* of Kenneth Train, David Revelt and Paul Ruud. The archive file *train0299.zip* (available at `http://elsa.berkeley.edu/Software/abstracts/train0296.html`) contains seven files including the code file `mxlp.g` and the data. This program has been written by Kenneth Train and used by this author in a collection of papers (see the web site above for more details) dealing with mixed logit models.[5]

In order to save space, we do not report the 1396 lines of code of the main file `mxlp.g`. This program can be run on the command line by entering `oxl -g mxlp.g`. Alternatively, it can be launched from OxEdit (`Modules/OxGauss` menu) or within GiveWin by using the menu `Modules/Start OxGauss`. Note that while the previous versions of OxGauss required a few modifications on the codes,[6] the program is now almost fully compatible with the new version. The only problem is that the program estimates the model by maximum likelihood, giving the opportunity to the user to chose the maximization routine domax of Paul Ruud or the commercial package maxlik. Launching the program could lead to the following error message:

```
C:\...path...\mxlp.g (1372): maxlik file not found
C:\...path...\mxlp.g (1373): maxlik.ext include file not found
```

To solve this problem one can either comment out lines 421 to 451 concerning the add-on maxlik (and use the domax procedure, i.e. the default option $OPTIM = 1$ in the program) or install the M@ximize package presented in Section 4. Table 1 below reports the estimated parameters obtained using Gauss 3.2 and OxGauss. As expected, they are very similar, if not identical (the only difference is detected after the sixth decimal of the standard errors).

Table 1 Estimated parameters produced by *mxlp.q* using OxGauss and Gauss 3.2 and the maximization routine domax of Paul Ruud.

Parameters	Gauss 3.2		OxGauss	
1.00000000	-0.90144883	(0.09161600)	-0.90144883	(0.09161599)
2.00000000	-0.21389101	(0.06638235)	-0.21389101	(0.06638234)
3.00000000	0.39170476	(0.06460276)	0.39170476	(0.06460276)
4.00000000	2.39484807	(0.32913048)	2.39484807	(0.32913048)
5.00000000	2.93704953	(0.44486784)	2.93704953	(0.44486785)
6.00000000	1.54510319	(0.19789114)	1.54510319	(0.19789114)
7.00000000	1.99077587	(0.36298403)	1.99077587	(0.36298404)
8.00000000	-8.68309604	(0.84790562)	-8.68309604	(0.84790555)
9.00000000	4.58749482	(0.52699408)	4.58749482	(0.52699402)
10.00000000	-8.73028891	(0.80958935)	-8.73028891	(0.80958927)
11.00000000	4.92021932	(0.65490251)	4.92021932	(0.65490246)

Robust standard errors are reported in parentheses.

[5] Mixed logit (also called random-parameters logit) generalizes standard logit by allowing the parameter associated with each observed variable (e.g., its coefficient) to vary randomly across units (e.g. individuals or customers).

[6] Philip Viton mentioned on his web site about 6 operations to make before succeeding to run the code without compilation error.

2.3 Understanding OxGauss

When an OxGauss program is run, it automatically includes the `\include\oxgauss.ox` file. This itself imports the required files:[7]

/include/oxgauss.ox

```
#define OX_GAUSS
#import <g2ox>
#import <gauss::oxgauss>
```

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

A nice feature of OxGauss is its transparency since most of the codes that link Gauss functions to Ox are gathered in the file `\include\g2ox.ox`. 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.

part of g2ox.ox

```
seqa(const start, const inc, const m)
{
    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)';
}
```

From this example, it is clear from that OxGauss does not translates the Gauss code in to Ox but makes the link between the Gauss function (here `seqa`) and its Ox counterpart (here `range`)

Recall that, in terms of compatibility, OxGauss is currently not designed to run (or call) Gauss procedures making use of new features specific to Gauss 4.0 and 5.0. While this can be seen as a

[7] For ease of presentation, the filename is printed in the upper right corner of the window.

drawback from the current version of OxGauss, one should emphasize again the open source character of OxGauss that implies that the users are free to provide and add further procedures that exploit specific Gauss 4.0 or Gauss 5.0 features. For this reason, when comparing OxGauss and Gauss, we will only consider Gauss 3.2 and 3.5 to ensure maximum compatibility even if we know that this will bias the conclusions that can be drawn from our comparisons. OxGauss being by definition a "work in constant progress", compatibility with specific Gauss 4 and 5 features is desirable.

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) like `sin` in Table A1 and open source functions (like `seqa`, see above) in Table A2. Adding adding all the functions leads to a total of 420 functions recognized by OxGauss. Table A3 in the appendix gives a list of 64 Gauss functions not supported by the current version of Ox (or about 15%). It is important to note that "Function name"() unsupported will be printed whenever the user calls or uses a Gauss function that is currently not supported by OxGauss. For instance, there is no equivalent of the Gauss function `intgrat2` (for the computation of double integrals) in Ox 3.3. For this reason, the corresponding procedure in `OxGauss.ox` just reports the error message `intgrat2() unsupported` (see below).

part of g2ox.ox

```
intgrat2(const f, const xl, const gl)
{
    __printunsup__("intgrat2");
    return .NaN;
}
```

If such a function becomes available in a next version of Ox, mapping `ingrat2` to the corresponding function in Ox will be a child's play!

2.4 Speed Comparison

As pointed out by Cribari-Neto (1997), *the main strength of Ox is its speed*, although Gauss performs quite well too and its speed performance are not far behind those of Ox. A recent and 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 just implements a layer on Ox, OxGauss is expected to be slower than Ox. But one may wonder how slower is it and how it really compares to Gauss in terms of speed? To answer these two questions we consider the Benchmark tests proposed by Stefan Steinhaus (edition 3). Note that since the functions *intquad2* and *intquad3* (double and triple integration of functions) are not available in Ox 3.3, the corresponding tests have been discarded which leads a total of 14 points of comparison. To perform the speed comparison, we did first execute the Ox benchmark program `Benchox2.ox` with 5 replications of each test on a Pentium III 450 Mhz and 526 MB RAM running under Windows 98 with Windows versions of the programs. We also conducted the same experiment with the Gauss benchmark program `Benchga2.prg` using OxGauss, Gauss 3.5. The results are reported in Table 2.

Table 2 Speed Comparison (timing in seconds) between Ox 3.3, OxGauss and Gauss 3.5.

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	4.380	4.437
Creation of 1000x1000 cross-product matrix:	8.953	12.817	12.777
Calculation of 500000 fibonacci numbers:	1.377	1.380	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	0.950	0.790
Linear regression over a 1000x1000 random matrix:	28.563	28.597	28.713

Benchmark programs run (5 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 faster than OxGauss and Gauss 3.5 (when taking the 14 experiments we see that Ox is about 4% faster than OxGauss and Gauss 3.5), the difference between OxGauss and Gauss 3.5 is very small (OxGauss is found to be on average about 0.5% faster than Gauss 3.5). [8]

[8]While 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. Note that we have done the same experiment with Gauss 3.2.29 and the result is that Gauss 3.5 is a much improved release in terms of speed.

3 Graphs using GnuDraw

While most graphical features of Gauss are recognized by OxGauss via Ox Professional for Windows (through GiveWin), Oxconsole has no graphs support. Nevertheless, the user of the console version can rely on the Ox package GnuDraw developed by Charles Bos 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. A nice feature of this package is its platform independence (unlike the current version of GiveWin which is only available for Windows) which provide non-Windows users (e.g. Unix, Linux, Solaris, Sun, etc.) an efficient graph support. 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.[9] 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.

gnuGauss.prg

```
library pgraph;

x=seqa(1,1,1000);
y=rndn(1000,1);
xlabel("X-axis");
ylabel("Y-axis: Normal(0,1) draws");
call xy(x, y);

end;
```

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

4 Some larger projects using M@ximize

The main drawback of OxGauss until now is that it was not suited to run Gauss codes that make use of commercial applications models. For instance OxGauss reports an error message if the Gauss code requires one of the optimizators from the modules Cml, Maxlik, or Optmum (see Section 2.2). This makes OxGauss useless in many situations of practical interest. To overcome this problem, we propose a set of three procedures that make the link between the optimizators of Ox 3.3 and the optimizators of Gauss. The package, called M@ximize 1.0, is open source and freely available from the authors on the web at the following address: `http:\\www.core.ucl.ac.be\~laurent`.

[9] More information about GnuDraw can be found in the help file *gnudraw.html*.

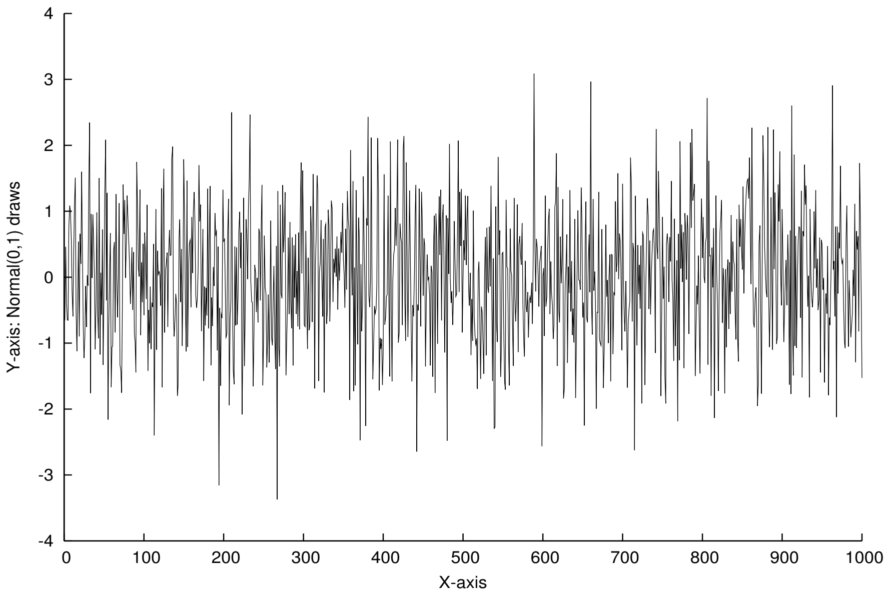

Figure 1 Graph produced by the program gnuGauss.prg under OxGauss (console version of Ox 3.3 and the GnuDraw package).

It is important to note that the purpose of M@ximize is not translate the Gauss optimizators into Ox as we just link the options; nor to clone the optimizators. Notice that in the first release of the package (M@ximize 1.0) not all the functions of these optimizators are available, although the most important options of Cml, Maxlik, and Optmum are taken into account. Instead of presenting all the features of the package, we will illustrate its usefulness through a number of concrete examples.

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. Here is a list of five web sites that we have visited from which both the data and the Gauss codes can be retrieved:
James Hamilton: `http://weber.ucsd.edu/˜jhamilto/`
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`
Rolf Tschering: `http://www.personeel.unimaas.nl/r.tschernig/`

We have also 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 that we have replicated. Most of these papers rely quite heavily on non-linear optimization techniques and thus require one the three above mentioned optimizators of Gauss.

While most of the codes can be run in their present form, some marginal changes in the GAUSS codes are sometimes needed. The most frequently encountered problems are:

- *Converting 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 Gauss data sets (v89 *.dht/.dat*) must be converted to the new Gauss format (v96 *.dat*). The program to do this conversion is ox/lib/dht2dat. The conversion can be run from the command line as:

  ```
  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.
- *Absence of 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, 9 and 13 use the Gauss function `cons` that requests an input from the keyboard (console) and puts it into a string. The typical use of this function is to generate 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.[10]

As illustration, we consider the Gauss package written by Rolf Tschernig accompanying the paper of Yang and Tschernig (1999) published in the *Journal of the Royal Statistical Society, Series B* (number 27 in Table 3). We focus on the example file provided by the author, i.e. *multband.tes* that estimates the asymptotic optimal vector bandwidth for simulated bivariate non-linear regression models. This file is made up of about 190 lines of Gauss code and includes three libraries, i.e. Optmum, pgraph and multband (a library provided by the author) as well as a set of three dll files. To use the package under Gauss, one has first to install the library multband by first copying the file *multband.lcg* into the subdirectory `./lib` of Gauss and second the files *multband.src* (about 2900 lines of code) and *multband.dec* (declarations of global variables) into the subdirectory `./src`. Finally, to complete the installation, one has to copy the three dll files *locling.dll, density.dll* and *loccubg.dll* into the subdirectory `./dlib`. Importantly, to use the package under OxGauss, one has to follow the same

[10] When using OxEdit to run the Gauss code, an additional shortcut has to be created. The simple solution is to 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.

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.	Bauwens, L. M. Lubrano (1998): Bayesian Inference on GARCH models using the Gibbs Sampler, *The Econometrics Journal*, 1, C23-C46.
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.	Yang, L. and R. Tschernig (1999): "Multivariate Bandwidth Selection for Local Linear Regression", *Journal of the Royal Statistical Society, Series B*, 61, 793-815.

instructions and copy the files into the existing subdirectories `./OxGauss/lib`, `./OxGauss/src` and `./OxGauss/dlib`.

Now the example file *multband.tes* can be executed.[11] The outputs obtained with OxGauss (left) and Gauss 3.2 (right) are reported below. Once again, we see that even with this complex program that uses dll files and an external library, the program does not report a compilation error and gives very similar results.

[11] Notice that this example file simulates a sequence of 250 observations. To make the comparison between Gauss and OxGauss possible we have changed the original code that now always uses (load) the same random numbers.

```
Ox version 3.30 (Windows) (C) J.A. Doornik, 1994-2003
hdrot_ll: chosen block: 2.0000000 1.0000000
Results from bandrot.g
h_ROT
     0.051626512
     0.051626512
hd_ROT
     0.044616931
     0.064029894
B_hat          0.23183482
Cm_hat
       5535.4077        477.33950
       477.33950        1305.0202
C_hat           7795.1069
hdrot_ll: chosen block: 2.0000000 1.0000000
hcdrotlp: chosen block: 2.0000000 1.0000000
hcdrotlp: Blamu
       78617.709       -3421.7567
      -4919.6541        80.592526

Results from bandpi.g
h_PI
     0.076176271
     0.076176271
hd_PI
     0.082851089
     0.064612911
Bd_hat         0.37280023
hd_ROT
     0.044616931
     0.064029894
C_hat           1227.6312
hC_ROT         0.18391889
Cm_hat
       458.95790        18.900462
       18.900462        1240.7565
hCd_ROT
      0.16024143       0.17575527
      0.17575527       0.37881573
```

```
hdrot_ll: chosen block: 2.0000000 1.0000000
Results from bandrot.g
h_ROT
     0.051626512
     0.051626512
hd_ROT
     0.044616919
     0.064029901
B_hat          0.23183482
Cm_hat
       5535.4077        477.33950
       477.33950        1305.0202
C_hat           7795.1069
hdrot_ll: chosen block: 2.0000000 1.0000000
hcdrotlp: chosen block: 2.0000000 1.0000000
hcdrotlp: Blamu
       78617.709       -3421.7567
      -4919.6541        80.592526

Results from bandpi.g
h_PI
     0.076176271
     0.076176271
hd_PI
     0.082851007
     0.064612978
Bd_hat         0.37280024
hd_ROT
     0.044616919
     0.064029901
C_hat           1227.6312
hC_ROT         0.18391889
Cm_hat
       458.95789        18.900464
       18.900464        1240.7565
hCd_ROT
      0.16024143       0.17575527
      0.17575527       0.37881573
```

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.

It is important to emphasize that OxGauss is potentially useful both for Gauss and Ox users. On the one hand Gauss codes can efficiently be called under Ox. This means that Ox programmers willing to use existing Gauss procedures do not have to translate the procedures into Ox but can call the Gauss code directly under Ox. On the other hand, OxGauss can be used to run the Gauss code(s) under Ox and hence to replicate the results of a paper for which Gauss code is made available by the author(s). The effectiveness of OxGauss 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 (26 papers published in international reviews and the procedures related to a book, see Table 3). Importantly in all cases the programs were found to be fully compatible with OxGauss in the sense that no change was required (or very minor changes) on the original code and that the results were almost identical.

To conclude, we believe that OxGauss could bring the Gauss and Ox user communities closer. We could even hope that Gauss users who already share their Gauss codes would start testing the compatibility with OxGauss, make the changes to ensure compatibility if necessary and indicate that their code is "OxGauss compliant".

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.

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-state.edu/pviton/support/oxgauss.

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	pqgwin	show
atan	enable	loadk k	prcsn 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	

Table A2: Open source functions supported by OxGauss

balance	corrvc	etstr	keyw	polymult	seekr	upper
band	corrx	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	eigrg	hessp	olsqr	rndgam	tab	
cmsoln	eigrg2	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	iscplxf	polymake	scalerr	unique	
corrm	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	