

# Ox programming and Bayes

ISBA Software review  
Charles Bos, 5 March 2005

## 1 Introduction

In a recent course on Ox programming the question came up: ‘Why do you want to program in the first place?’. A wrong answer to the question was that one would do it for the fun of it; better might run like

To get to the results we need, in a fashion that is controllable, where we are free to implement the newest and greatest, and where we can be ‘reasonably’ sure of the answers

Clearly, there are multiple ways to get such results, and one of those ways is to program the necessary methods in the Ox programming language. For me, this language has been my programming tool of choice for many years already, and in this review I’ll give an overview of its possibilities, advantages over some other languages, and the way it can help in a Bayesian setting as well.

## 2 Ox and its possibilities

Ox (Doornik 2001) in itself is a matrix programming language with built-in capabilities of using the object-oriented approach. As a matrix programming language, it can be compared to languages as Gauss, Matlab, S-Plus and R. Compared to these other languages, Ox is a relatively young language, with origins dating back about ten years. It grew within a community of econometricians, at first, and is spreading around the last 5 years among a wider audience.

The language-syntax displays its origins: Ox started as a shell around C, allowing for quick matrix computations, without having to care for the problem of memory management. Later on, the goal of Ox became bigger, to provide a full and flexible environment in which every conceivable computational problem could in principle be implemented, with bigger ease of programming than reverting to C or Fortran, without relinquishing all of the speed advantages of those languages (see also below).

With respect to the ease of programming, Ox reached this goal clearly. The syntax of the language is strict, clear, and displays similarities to C which will help a programmer to make the switch without much of a problem. The object-oriented programming capabilities are implemented as a subset of possibilities in C++, in the sense that there is sufficient flexibility to inherit from existing classes, but without introducing unnecessary complexity in the syntax.

Ox comes with an extensive HTML manual of all available functions, which is a great help for both beginning and more advanced programmers, if only to check the order in which certain functions expect their parameters. Apart from the included help file, there is an active mailing list of Ox-users.

The combination of the clear syntax, the possibility to write object-oriented code, and an active community of users, are helpful in the respect that packages can easily be shared

among users. The background of Ox in the econometric research community implies that many statistical and econometrical algorithms are implemented in the language itself, or have been made available by others in the form of convenient plug-in packages, enlarging the possibilities of Ox. For instance, a simple line of code like

```
#include <packages/ssfpack/ssfpack.h>
```

includes code concerning state space models (Koopman, Shephard, and Doornik 1999), allowing for the filtering, simulation, estimation and the like of unobserved component models following the theory by Durbin and Koopman (2001).

Using packages of well-known econometricians, allows you to be reasonably sure of the answers; the clean syntax helps in writing good code which is relatively easy to check for bugs, such that also in that respect the programming you do can be ‘controllable’.

### 3 Ox and speed

Even though computers are getting faster, for Bayesians the speed of their programs still is a bottleneck. When I started research on the hedging of currency risk (Bos, Mahieu, and Van Dijk 2000), needing to run through time series with several thousands of observations, for a range of different models, it was the speed advantage of Ox which made me switch to it. Not only is the language itself among the fastest higher level programming languages, the aforementioned SsfPack provides all the Kalman filtering routines for state space (and their special case, ARIMA) models written in C-code. All time-consuming operations of the package are run in C, after which the results are seamlessly passed on to Ox.

This connection between C and Ox came in very handy as well for those models which did not fit exactly in the state space framework: For models like the stochastic volatility or GARCH models, the loops that took most time were easily programmed in C, performing all bookkeeping, input and output in Ox.

And another speed-related advantage: Ox comes in versions for a range of computing environments. The programs for the hedging research were actually written on a Windows desktop, while I ran the programs either on a Sun Solaris server, or on a Linux machine, whichever was available and quicker. This flexibility of using the same codebase on different machines has always been of great convenience to me.

Even without adding in task-specific C-code Ox is a very quick computing engine. Steinhilber (2004) provides a series of comparisons of several mathematical programming environments, in which Ox comes out among the top performers with respect to speed.

### 4 MC2Pack: MCMC sampling in Ox

When working with Bayesian sampling algorithms, one quickly finds himself rewriting the same series of bookkeeping routines. Gathering samples, checking convergence, making graphs, even the sampling algorithms themselves are the same for different models. When I was working on the algorithm of Adaptive Polar Sampling (Bauwens, Bos, Van Dijk, and Van Oest 2004), meant to be more robust against multimodal posterior densities than standard algorithms, I needed a better solution for quickly comparing the performance of different models and algorithms. Out of this need evolved the MC2Pack package.

Listing 1: Minimal sampling program

```

#include <oxstd.h>
#include <packages/mc2pack/mc2pack.h>

AvgLnPostStack(const vP, const adLnPost, const adDum, const adDum)
{
    // Specification of the posterior density
}

main()
{
    decl mcmc, vMu, mS2, mTheta, vW;

    vMu= ...; mS2= ...; // Initial estimates of location and scale
    mcmc= new MC2Pack();
    mcmc.SetMethod(MC_MH); // or MC_APS, MC_APIS, MC_IS, MC_GG
    mcmc.SetCandidate(MC_CSTUD, {1, 4}); // or a different t(df) or
    // user-provided density
    // Only for clarity: Parameter names can be given
    mcmc.SetParNames({"Air Flow", "Water Temp", "Acid Conc", "Sigma", "k", "Alpha"});
    mcmc.SetSample(10000); // Set sample size
    mcmc.SetPosterior(AvgLnPostStack); // Posterior density
    mcmc.SetOutput("excl/mcstack"); // Output goes here

    // Possibly set location and scale parameters for candidate, and
    // number of observations
    mcmc.SetLocationScale(&vMu, &mS2, iT, FALSE);

    mcmc.Simulate(); // And simulate

    mcmc.GetDraws(&mTheta, &vW);
    mcmc.Marglik(MC_MLKERN, meanr(mTheta)); // Marginal likelihood
    delete mcmc;
}

```

To make the collection of a Bayesian sample of parameters easier, the MC2Pack package, available from <http://www.tinbergen.nl/~cbos/software/mc2pack.html> can be of help. This package asks the user to specify a posterior function in a standard format, and essentially can sample from it using a tiny program of the format of Listing 1.

The listing only gives a first impression of the convenient object-oriented fashion in which a project in Ox can be set up. The present version of the package implements methods as the importance, Metropolis-Hastings, (Griddy) Gibbs and Adaptive Polar sampling algorithms, but it could be expanded for inclusion of further methods relatively easy.

Apart from sampling algorithms, the package also includes the marginal likelihood calculation methods which were compared in Bos (2002). These range from kernel-based methods, the LaPlace approximation, a method based on the harmonic mean or on a prior sample, brute-force numerical integration, and also the method of Chib (1995) to get the marginal likelihood from the output of the Gibbs sampler.

## 5 Concluding remarks

Ox is a clean and quick language by itself. Combined with its packages, it can be a convenient solution for writing Bayesian samplers. In previous reviews in the ISBA bulletin, the controversy between writing code in C/Fortran or Matlab/S-Plus seems eternal. The combination of the speed of C and the power of a convenient matrix programming language as Ox can be

a perfect alternative.

## References

- Bauwens, L., C. S. Bos, H. K. Van Dijk, and R. D. Van Oest (2004). Adaptive radial-based direction sampling: Some flexible and robust Monte Carlo integration methods. *Journal of Econometrics* 123(2), 201–225.
- Bos, C. S. (2002). A comparison of marginal likelihood computation methods. In W. Härdle and B. Ronz (Eds.), *COMPSTAT 2002 – Proceedings of Computational Statistics*, pp. 111–117.
- Bos, C. S., R. J. Mahieu, and H. K. Van Dijk (2000). Daily exchange rate behaviour and hedging of currency risk. *Journal of Applied Econometrics* 15(6), 671–696.
- Chib, S. (1995). Marginal likelihood from the Gibbs output. *Journal of the American Statistical Association* 90(432), 1313–1321.
- Doornik, J. A. (2001). *Object-Oriented Matrix Programming using Ox*. London: Timberlake Consultants Ltd. See <http://www.doornik.com/>.
- Durbin, J. and S. J. Koopman (2001). *Time Series Analysis by State Space Methods*. Oxford: Oxford University Press.
- Koopman, S. J., N. Shephard, and J. A. Doornik (1999). Statistical algorithms for models in state space using SsfPack 2.2. *Econometrics Journal* 2, 107–160.
- Steinhaus, S. (2004). Comparison of mathematical programs for data analysis. <http://www.scientificweb.com/ncrunch/>, version 4.42.