Advanced Programming in Quantitative Economics

Introduction, structure, and advanced programming techniques

Charles S. Bos

VU University Amsterdam
Tinbergen Institute

c.s.bos@vu.nl

15 – 19 August 2011, Aarhus, Denmark

Outline

Day 4 - Afternoon

14.15 [Seminar]

16.00L Added capabilities

- Graphics packages
- SsfPack/Arfima and others

17.00 End

Include

Enlarging the capabilities of ox beyond `oxstd.h` capabilities: Either

```
#include <oxprob.h>
```

(to include the mentioned file literally within the program at that point, and will be compiled in), or

```
#import <maximize>
```

(to import the code when needed; pre-compiled code is used when available)

Ox-provided packages

Package	Purpose
oxprob.h	Extra probability densities
oxfloat.h	Definition of constants
oxdraw.h	Graphics capabilities (∗)
arma.h	ARMA filters and generators
quadpack.h	Univeriate numerical integration
maximize	Optimization using Gauss-Newton or Simplex methods (∗)
maxsqp	Maximize non-linear function with sequential quadratic programming
solvenle	Solve a system of non-linear equations
solveqp	Solve a quadratic program with restrictions
database	General class for creating a database
modelbase	General class for building a model
simulation	General class for simulation exercise

User-provided packages

Package	Author	Purpose
arfima	Doornik, Ooms	Long memory modelling
dcm	Eklof, Weeks	Discrete choice models
dpd	Doornik, Arellano, Bond	Dynamic Panel Data models
financialnr	Ødegaard	Financial numerical recipes
gnudraw.h	Bos	Alternative graphing capabilities
maxsa.h	Bos	Simulated Annealing
mc2pack.h	Bos	Markov chain Monte Carlo estimation
msvar	Krolzig	Markov switching
oxutils.h	Bos	Some convenient utilities (∗)
oxdbi	Bruche	A database independent abstraction layer for Ox
ssfpack.h	Koopman, Shephard, Doornik	State space models
...	...and many others	
m@ximize	Laurent, Urbain	Use CML optimisation in OxGauss
oxgauss	Doornik	Run Gauss code through Ox

- Packages reside either in `ox-home/packages`, or in a local packages folder.
- After including the package, the package is supposed to work seamlessly with Ox
- Easy and clean way of communicating research

A package: oxutils

What does 'seamless' mean?

Standard situation: What is the size of a matrix I'm using?

```
main()
{
  ...
  print (rows(mX)|columns(mX));
}
```

How often would you use this code while debugging?

A package: oxutils II

Alternative: Use a package with some extra functions, not previously available

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

main()
{
  ...
  print (size(mX));
}
```

Check manual

<ox-home>\packages\oxutils\doc\oxutils.html

Other routines I use plenty:

`info`	Measure time an iteration takes, time until end of program
`TrackRoutine`	Routine to profile your program
`printtex`	Replacement for `print`, outputting in LaTeX format
`ReadArg`	Read arguments from command line
`setseed`	Reset the random seed, psuedo-randomly

A package: oxutils III

(From tomorrow's slides on speed)

Use `TrackTime("concat")` to profile a piece of code, get a report using `TrackReport()`

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

main()
{
  decl iN, iK, mX, j;

  iN= 1000;     // Size of matrix
  iK= 100;

  TrackTime("concat");
  mX= <>;
  for (j= 0; j < iN; ++j)
    mX|= rann(1, iK);

  TrackTime("predefined");
  mX= zeros(iN, iK);
  for (j= 0; j < iN; ++j)
    mX[j][]= rann(1, iK);
  TrackTime(-1);

  TrackReport();
}
```

Output:

```
Ox Professional version 6.00 (Linux_64/MT)
Time spent in routines
concat                2.42    0.99
predefined            0.02    0.01
Total:  2.44
```

A package: SsfPack

Model in State Space:

$$\alpha_{t+1} = d + T\alpha_t + \eta_t$$
$$y_t = c + Z\alpha_t + \epsilon_t$$
$$(\eta_t', \epsilon_t')' \sim \mathcal{N}\begin{pmatrix} HH' & HG' \\ GH' & GG' \end{pmatrix}$$

- Flexible structure for all linear Gaussian time series models
- Incorporates ARMA models, Exponentially Weighted Moving Average, regression models
- Allows for distinction of level, trend, seasonal components
- Can easily handle time series with missing observations
- Likelihood expressed analytically (through *recursion* formula)

SsfPack: Recursions

Recursion formula:

- Kalman filter (and related routines)
- Cumbersome to program in robust and general way

$\Rightarrow$ SsfPack (S.J. Koopman, N. Shephard, J. Doornik)

SsfPack: LLN

Some notation:

$$\begin{pmatrix} \alpha_{t+1} \\ y_t \end{pmatrix} = \delta + \Phi\alpha_t + u_t \qquad\qquad u_t \sim \mathcal{N}(0, \Omega)$$

Simple Local Level with noise or Random Walk with Noise model:

$$\delta \equiv \begin{pmatrix} 0 \\ 0 \end{pmatrix} \qquad \Phi = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \qquad \Omega = \begin{pmatrix} .01 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\alpha_{t+1} = \alpha_t + \eta_t \qquad \eta_t \sim \mathcal{N}(0, .01) \qquad \text{Random walk}$$

$$y_t = \alpha_t + \epsilon_t \qquad \epsilon_t \sim \mathcal{N}(0, 1) \qquad \text{with noise}$$

SsfPack: Likelihood

Define $a_{t+1} = E(\alpha_{t+1}|Y_t)$, with corresponding $P_{t+1} = \text{cov}(\alpha_{t+1}|Y_t)$. Then, observing y_{t+1} provides extra info on $\alpha_{t+1} \Rightarrow$ Filter...

$$v_t = y_t - Za_t \qquad F_t = ZP_tZ' + GG'$$

$$K_t = (TP_tZ' + HG')F_t^{-1}$$

$$a_{t+1} = Ta_t + K_tv_t \qquad P_{t+1} = TP_tT' + HH' - K_tF_tK_t'$$

Prediction error $v_t|Y_t \sim \mathcal{N}(0, F_t)$

$$\log \mathcal{L}(Y_n;\theta) = \sum_{t=1}^{n} \log p(y_t|y_1,..,y_{t-1};\theta)$$

$$= -\frac{n}{2}\log(2\pi) - \frac{1}{2}\sum_t(\log|F_t| + v_t'F_t^{-1}v_t)$$

Hassle to compute likelihood, recursion, and slow...

SsfPack: Package

Alternative: Use package...

```
main()
{
  decl iN, vP, vAlpha, vY, mPhi, mOmega, mSigma, ir, dLnPdf, dVar;

  iN= 10000;
  vP= <.1; 1>;
  vAlpha= cumulate(vP[0]*rann(iN, 1))';     // Generate state
  vY= vAlpha + vP[1] * rann(1, iN);         // Generate data

  mPhi= <1; 1>;
  mOmega= diag(vP);                         // Place variances
  mSigma= <-1; 0>;
  ir= SsfLik(&dLnPdf, &dVar, vY, mPhi, mOmega, mSigma);
}
```

What is this `SsfLik`?

1. Internal Ox function, with heading in `ssfpack.h` (compare `oxdraw.h`)?
2. Ox function, defined in `ssfpack.h`
3. Other type of function, reference in `ssfpack.h`?

SsfPack: Load those files

Let's take a peek...

Listing 1: .../ssfpack.h

```
extern "packages/ssfpack/ssfpack,FnSsfLik"
    SsfLik(const adLik, const adVar, const mY, const mTZ, const mHG, ...);
```

(check `extern` statement)

Apparently

- Routine is internally called `FnSsfLik`
- Heading is defined here in Ox terms
- Code is declared *externally*
- ... in some .dll/.so file
- $\Rightarrow$ C-code included for Ox functionality...

SsfPack: Summary

SsfPack

- provides easy access to wealth of routines concerning State Space models
- gives seamless integration into Ox
- codes routines in optimized C-code
- stems from some of the leading researchers in the field