

Advanced Programming in Quantitative Economics

Introduction, structure, and advanced programming techniques

Charles S. Bos

VU University Amsterdam

`c.s.bos@vu.nl`

20 – 24 August 2012, Aarhus, Denmark

Tutorial Day 1 - Morning

11.00 Implementation of first program

1. Get acquainted with help system
2. Print a matrix, get started (`main`, `print`)
3. Print successively rows of a matrix (`for-loop`)
4. Implement back-substitution, to solve $\mathbf{U} \mathbf{x} = \mathbf{b}$

12.30 Lunch

Practicalities

- ▶ Create your own directory (with your name) on the network drive
- ▶ In your directory, create a subdirectory for each practical, for me this would be cbos/p1a, for afternoon I would use cbos/p1b
- ▶ Save your work in these subdirectories, such that TAs can help/take a look over your shoulder
- ▶ Feel free to ask

Exercise 1: Get acquainted

- ▶ Open up OxEEdit, run `myfirst.ox`
- ▶ Open up the help-file (press F1 in OxEEdit)
- ▶ Leaf through the 'function summary' in the help file: What types of functions are available?
- ▶ Search for the `print` and `println` statements. What is the difference?

Exercise 2: Print a matrix

Write an Ox program which

- ▶ contains all necessary explanations
- ▶ declares a matrix and a vector, giving them the values

$$A = \begin{pmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 0 & -3 \end{pmatrix}, \quad b = \begin{pmatrix} 16 \\ -6 \\ -9 \\ -3 \end{pmatrix}$$

- ▶ prints them, with output on screen as to which is which
- ▶ prints the maximum element of A, and the minimum of b.
- ▶ you save as `exer2print.ox`.

Exercise 3: Backsubstitution

Solve the system $Ax = b$ for the matrices you defined before. As a hint, the way to solve it was

$$x_n = b_n / a_{nn}, \quad x_i = \left(b_i - \sum_{j>i} a_{ij}x_j \right) / a_{ii}, \quad i = n-1, \dots, 1$$

Think about it before you begin: It might be easier to first define

$$s = \sum_{j>i} a_{ij}x_j$$

and for all x_n, \dots, x_1 use the same formula for solving.

For this purpose, maybe start with a simple `exer3for.ox` where you show you can count backwards using a for-loop.

Initialise `x` as a vector of the correct size of zeros (see `zeros()`).

Afterwards, write `exer3bs.ox`, showing the solution for `x`. How can you/the program check that your solution is correct?

Exercise 4: BS function

As an extra, take your old program and push the backsubstitution into a function. What would the inputs and outputs of the function be?

Save the result as `exer4bsfunc.ox`.

Tutorial Day 1 - Afternoon

14.30 Targets:

1. Learn some syntax
2. *Use* the syntax
3. Get a simulation running

16.00 End for today

Learning syntax

Four sources for now:

- ▶ Introduction to Ox (included in help system as PDF)
- ▶ Syntax sheets (see webpage, PDF)
- ▶ Language tutorial (included in help system as HTML)
- ▶ Tutors

Check out those sources first, find your way through.

Spend first \pm half hour on one of those sources, such that you know most important syntax.

Exercise OLSGenS

Take the model

$$y = \mathbf{X}\beta + \epsilon \qquad \epsilon \sim \mathcal{N}(0, \sigma^2)$$

with $n = 20$ observations, $\beta = [1; 2; 3]$, $\sigma = 0.25$ and $\mathbf{X} = [1 \ u_1 \ u_2]$ where $u_i \sim U(0, 1)$.

1. Write a program which creates the \mathbf{X} matrix; print it, and make sure it is what you want it to be. Save the program as `olsgens0.ox`.
2. Generate data y from the model. Is the mean of y roughly what you expect it to be? Save the program as `olsgens1.ox`.

OLSGenS 2

3. Estimate b using OLS. Is the estimate decent? Save as `olsgens2.ox`.
4. Add a loop, such that you generate $S = 1000$ samples of y successively, and for each estimate and store b . Print as output the mean and variance of the estimated b 's. Save the program with the name `olsgens3.ox`.
5. Compare the results you get with the theoretical covariance matrix of $\Sigma = \sigma^2(X'X)^{-1}$. Does it all still make sense? Save as `olsgens4.ox`.

Tomorrow

Try to get along with the exercise; leave it in your personal directory for us to check; leave a `olsgen.txt` next to it with questions to ask, if you have any.

Mind you: Course is to get practice, not to do everything 'perfect' at the first try.

Tutorial Day 2 - Morning

10.30P Tutorial

- ▶ Addresses
- ▶ Minimal blocks

12.00 Lunch

Exercise: Addresses

Create a (or multiple small) program(s) with a `main` and a function:

1. Pass an integer to the function, return the square.
2. Pass an integer to the function, pass the square back through an address.
3. Pass a string, e.g. `sX= "Aargus";` to the function. Can you change only the "g" to a "h"? (Maybe first try without the function: How would you change an element of a string, directly within main?)
4. Pass the array `aX= {"Aargus", 5, <2.4, 4.6>};` to the function, change the 5 to a 7, the 4.6 to its square, and the "g" to a "h".

Ensure you *fully* understand the address thing here... Talk to the tutor if not.

Exercise: OlsGen and Sim with functions

Target of this exercise is to set up a program for a slightly larger task. The task itself is not hard, but the idea is to do it in a structured, extensible way.

Target:

- ▶ Start with a set of regressors, X (e.g. take $X = [1 \ u_1 \ u_2]$ with $u_i \sim U(0, 1)$), and vector of parameters β (e.g. $\beta = [1; 2; 3]$). Assume we use $n = 20$ observations for this exercise.
- ▶ Repetitively, say S times, generate n observations from $y = X\beta + \sigma\epsilon$,
- ▶ For each iteration, estimate and save for later use the parameter estimates $\hat{\beta}$ and residual standard deviation s ,

$$s^2 = \frac{1}{n - k} \sum e_i^2, \quad e = y - X\hat{\beta}$$

- ▶ After the computations, provide interesting output.

For the exercise, start e.g. with $S = 1000$, using $\sigma = 2$.

Exercise: OlsGen and Sim II

1. Analyse the exercise: What variables do I need for initial settings; what separate tasks do I have; hence, what routines could I use; what are inputs and outputs to those routines; what is the final output. *Write, on paper, an indication of the plan for your program!*
2. Start the programming, but in steps: First write `olssim0.ox`, containing only the outline of the program including the headings of the routines, then `olssim1.ox` which does the initialisation, when it works move to `olssim2.ox` which forgets about the loop, estimates just once the model and prints the outcome, then a third step in `olssim3.ox` where you add the loop and collect results (maybe start with $S = 5$), etc.
3. Check the output: Can you print means and standard deviations of $\hat{\beta}$, s ? Are those values 'expected'?

Exercise: OlsGen and Sim III

In each iteration, estimate both

$$b = \hat{\beta} = (X'X)^{-1}X'y$$

$$s^2 = \frac{1}{n-k} e'e$$

$$e = y - X\hat{\beta}$$

and save b and s , e.g. in a matrix `mOLS`:

$$\text{mOLS} = \begin{pmatrix} b_1 & b_2 & b_3 & s \\ b_1 & b_2 & b_3 & s \\ \vdots & \vdots & \vdots & \\ b_1 & b_2 & b_3 & s \end{pmatrix} \begin{array}{l} \text{for replication 1} \\ \text{for replication 2} \\ \vdots \\ \text{for replication } M. \end{array}$$

Exercise: Output

Output could be in the form of text or possibly of graphs. Some exemplifying code (check manual for explanation!):

Listing 1: olsprint.ox

```
mMS= <1, 2, 3, 4; .5, .7, .6, .8>;
print ("%c", {"b1", "b2", "b3", "s"},           // Column names
      "%r", {"mean", "std"},                     // Row names
      "%cf", {"%8.3f", "%8.3f", "%8.3f", "%8.3f"}, // Column formats
      mMS);                                     // Matrix to print
```

Tutorial Day 2 - Afternoon

14.30P Tutorial

- ▶ Exercises HB
- ▶ Data duration model

16.00 End

Rows or columns

Does it make a difference to use rows or columns, for the speed of your program? Find this out...

- ▶ Make a program which fills a $n \times n$ matrix with random numbers, e.g. with $n = 1000$
- ▶ Do this either
 1. at once, using `rann(iN, iN);`
 2. filling `mX` row by row, `mX[i] [] = rann(1, iN);`
 3. filling `mX` column by column, `mX[] [i] = rann(iN, 1);`
- ▶ Measure the time each of these takes. Use the `dTime= timer();` and `println (timespan(dTime));` to measure.
- ▶ If your computer is too quick, repeat the filling say $S = 10$ or $S = 1000$ times, or more, until you see which one works better. Or increase n ?

Precision/smallest number in Ox

What is the smallest number you can create which is still larger than zero?

1. Start of with a number of choice
2. Check if it is different from 0
3. Divide it by two
4. and repeat from 2

Report the last number which you found different from zero. Also report the number of times you divided by 2.

For this exercise you might want to use a construction like

```
do
{
    // Do something}
}
while (dX != 0);           // Test whether dX is equal to zero
```

Accuracy in addition

Define three numbers

$$a = 0.1234567 \times 10^0 \quad b = 0.4711325 \times 10^4 \quad c = -b$$

and compute the outcomes of

$$a + b + c$$

$$a + (b + c)$$

$$(a + b) + c$$

Is there a difference?

Afterwards, do the same thing but with 10^{40} instead of 10^4 : Do you now find a difference? Can you find the number of significant digits With what k , for 10^k , does the result seem correct?

Memory use

Does declaring new memory, or using local variables, take time?
Investigate this by

1. Writing a loop which S times creates a random matrix of size $n \times n$ of random numbers, with n, S sufficiently large, and time it
2. Then do the same, but each time in the loop reset the matrix to a scalar 0, before assigning the big matrix to it.
3. Then do the same, but call a function which locally declares a matrix `mX` and assigns the random numbers to it.

Any difference? (Hint: I cannot find much of a difference myself)

Duration modelling

The duration model is heavily used e.g. to model the duration of unemployment spells. It also provides a convenient workhorse during this course as it

- ▶ contains relatively few parameters
- ▶ does have restrictions on the parameter space
- ▶ can be estimated using a loglikelihood approach
- ▶ allow for easy extension from the regression framework

See a.o. Lancaster, 'The Econometric Analysis of Transition Data' (1990) for details.

Here we use a simplified version of the model, assuming all data is observed.

Duration: The (simplified) model

Durations y_i are assumed to be distributed according to

$$y \sim \text{Weib}(\alpha, \lambda) \quad f(y; \alpha, \lambda) = \alpha \lambda^\alpha y^{\alpha-1} \exp(-\lambda^\alpha y^\alpha)$$

Dependence on personal characteristics can be introduced by taking

$$\lambda_i \equiv \exp(X_i \beta)$$

$$y_i \sim \text{Weib}(\alpha, \lambda_i)$$

Duration: Simulation

Write a program which generates $N = 1000$ durations y from the Weibull model, with $\beta = (1 \ 1)'$, $X = [1 \ N(0, 1)]$, $\alpha = 1.5$.

Some remarks:

- ▶ Think about the status of $y, X, \lambda, \alpha, \beta$: Which is parameter, which is 'fixed data', 'derived data' etc?
- ▶ Work *in matrices* as far as possible.
- ▶ Work in routines: In what steps can you generate this data? What should you retain?
- ▶ If you generate $E \sim \text{Exp}(\lambda_e = 1)$, then $Y = E^{1/\alpha} / \lambda_w \sim \text{Weib}(\alpha, \lambda_w)$. Use this relation, and the element-by-element division and power operators `./` and `.^` to obtain a sample from the requested Weibull.
- ▶ `ranexp` does not work immediately. You'll miss a line `#include <oxprob.h>` at the beginning. What is the logic?

Duration: Output

In later tutorials, we will need data from this model. Therefore,

- ▶ Save a data set `data/genrdur.fmt` containing for each individual the duration y and the explanatory variables X .
- ▶ Get some output on the y 's and X 's.

Don't try to do all at once: First check that you can generate e.g. from the Weibull for fixed λ , then generate separate λ 's, etc...

Tutorial Day 3 - Morning

10.30P Tutorial

- ▶ Loglikelihood duration model
- ▶ Standard deviations
- ▶ Analytical score

12.00 Lunch

Duration: The (simplified) model

As a reminder: Durations y_i are assumed to be distributed according to

$$y \sim \text{Weib}(\alpha, \lambda) \quad f(y; \alpha, \lambda) = \alpha \lambda^\alpha y^{\alpha-1} \exp(-\lambda^\alpha y^\alpha)$$

Dependence on personal characteristics can be introduced by taking

$$\begin{aligned} \lambda_i &\equiv \exp(X_i \beta) \\ y_i &\sim \text{Weib}(\alpha, \lambda_i) \end{aligned}$$

Make sure you have data available, e.g. in `data/genrdur.fmt`, or download a file from the web.

Duration: Optimisation

Central to optimisation is the log likelihood function. In this case, it would read

$$\begin{aligned}\lambda_i &\equiv \exp(X_i\beta) \\ \log \mathcal{L}(y; \theta) &= \sum_i (\log \alpha + \alpha \log \lambda_i + (\alpha - 1) \log y_i - \lambda_i^\alpha y_i^\alpha) \\ &= N \log \alpha + \alpha \sum_i \log \lambda_i + (\alpha - 1) \sum_i \log y_i - \sum_i (\lambda_i y_i)^\alpha\end{aligned}$$

Work on this in steps...

Duration: Steps

Perform, in steps, for instance

1. Get the outline of your loglikelihood function. Call it from main, with a valid vector of parameters, and set the likelihood value equal to the average of your y 's.
2. Extract the vector of parameters, into β and α . Print them separately from the loglikelihood function.
3. Check the value of α . If negative, maybe return a zero?
4. Construct a vector of λ 's. Does this work?
5. Construct full loglikelihood function. Does the value seem 'logical'?
6. Run `MaxBFGS()`. What return value `ir` do you get, what does it mean? What is `vP`?

Duration: Standard errors

For the standard errors, you had to find

$$\Sigma(\hat{\theta}) = -H(\hat{\theta})^{-1}$$

$$H(\hat{\theta}) = \frac{\delta^2 l(Y; \theta)}{\delta \theta \delta \theta'} \Big|_{\theta = \hat{\theta}}$$

Some standard code could look like

```
if (Num2Derivative(AvgLnLiklRegr, avP[0], &mH))
  mS2= invertgen(-mH, 30)/iN,
  avS[0]= sqrt(diagonal(mS2)');
```

7. Get the standard errors with it. How do they change if you only use the first 10 observations?
8. Beautify the output: Get a nice print with the maximum likelihood you find, the type of convergence, the parameters, standard errors and t -values

Duration: Analytical score

As an extra: Work out the analytical score for the model

9. Find it on paper
10. In the program, add the necessary code (only compute it if asked for it, if `avScore` is an address!)
11. Check it, contrasting your analytical score to `Num1Derivative`
12. Evaluate the number of function evaluations needed when using numerical scores, and when using analytical scores (hint: Define an extra global variable, `s_iEval`)

Tutorial Day 3 - Afternoon

14.30P Implementing covariance estimation

- ▶ Duration model restricting α
- ▶ Covariance of parameters

16.00 End

18.00 Course dinner at 'Sct. Oluf'

Duration: Restrictions

Continue with the earlier exercise.

Add the restrictions that

1. $\alpha > 0$
2. $0 < \beta_0 < 2$ (as an extra, not really implied by model...)

Duration: Schedule

Remember steps:

- ▶ Write `TransPar`, `TransBackPar`
- ▶ Test them against each other
- ▶ Implement within full program
- ▶ Adapt standard errors, Delta method using `NumJacobian`
- ▶ Compare outcome with earlier program

Tutorial Day 4 - Morning

10.30P Estimating a duration model

- ▶ Setting up a class
- ▶ Moving the likelihood

12.00 Lunch

14.15 Empty slot?

16.00L Added capabilities

- ▶ Graphics packages
- ▶ SsfPack/Arfima and others

17.00 End

Duration: Working in a class

Target: Estimate your duration model using a class

Possible starting point: `lists/class/ols.ox`,
`lists/class/olsclass.ox`

Possible steps:

1. Adapt your data generating program, such that it saves a `.in7` file. Check manual, one option is

```
savemat("data/durgen.in7", vY~mX, {"Y", "X1", "X2"});
```

2. Prepare `include/durclass.ox`, with an almost empty class deriving from `Modelbase`
3. Prepare `durmain.ox`, including
`#include <include/durclass.ox>`, and declare a new package. Do you get the correct package name on the output?
4. Read the data into the class, select X and Y variables (see `lists/class/ols.ox`)

Duration: Class II

5. Check if you can print the data from the class, in an almost empty `DoEstimation(vP)`
6. Maybe add the `InitPar` member, for finding initial parameters. Place them using `SetPar(vP)` (which sets the parameter count as well)
7. Prepare a `GetParNames()`, which should return an array of strings with the parameter names
8. Add a member `AvgLnLiklDur(...)`, see if you can call it once
9. Use `MaxBFGS` from `DoEstimation`, put the results in place
10. Add the `Covar` member, computing the covariance matrix of the parameters
11. Remember setting the loglikelihood value `m_dLogLik` in place

Tutorial Day 5 - Morning

10.30P Selected exercises. Choice of

- ▶ Reading HF data
- ▶ Implement HF Autoregressive Duration Model
- ▶ Using graphing package
- ▶ AR(p) estimation with/without ARFIMA package

12.00 Lunch

IBM Tick data

High frequency data displays strange characteristics.

As an exercise, investigate the fraction of trades which show small price changes.

Take the data file `ibm0908tick.csv` (Source: WRDS) with the prices , and see if you can recreate the following table:

| ΔP | Price differences | | | | | | | | |
|------------|-------------------|-------|-------|--------|--------|--------|-------|-------|-------|
| | -0.04 | -0.03 | -0.02 | -0.01 | 0.00 | 0.01 | 0.02 | 0.03 | 0.04 |
| n | 17697 | 32244 | 57774 | 127748 | 637664 | 127676 | 58104 | 32223 | 17801 |
| % | 1.50 | 2.73 | 4.90 | 10.82 | 54.03 | 10.82 | 4.92 | 2.73 | 1.51 |

Also get me a plot of the prices during the first 45 minutes of trading on the first day of the sample.

AR Conditional Duration models

The file mmm9912-adur.txt contains 1680 adjusted intraday trading durations of 3M stock in December 1999. There are 39 10-minute intervals in a trading day. The URL for the data file: <http://gsbwww.uchicago.edu/fac/ruey.tsay/teaching/fts2/>

- (a). Build a duration model for the adjusted series using an Exponential duration model.
- (b). Check your duration model implementation using simulated data

Autoregressive Conditional Duration ACD(1,1) model:

$$x_i = \psi_i \varepsilon_i$$

$$E(\varepsilon_i) \equiv 1, \quad E(x_i) = \psi_i$$

$$\psi_i = \omega_0 + \gamma_1 x_{i-1} + \omega_1 \psi_{i-1} \varepsilon \sim \text{Exp}(1)$$

ACD models, deriving moments, likelihood and tests

unconditional expectation:

$$E(x_i) = \frac{\omega_0}{1 - (\gamma_1 + \omega_1)}$$

Log likelihood: (Jacobian term, Heij et al. §1.2, (1.10), (1.19))

$$x = \psi \varepsilon, \quad f(x) = f(\varepsilon) \left| \frac{d\varepsilon}{dx} \right| = f\left(\frac{x}{\psi}\right) \left| \frac{1}{\psi} \right|$$

Parameter restrictions:

For specific distributions of ε : Derive unconditional variance of $x_i \Rightarrow$ interpret/restrict parameters during estimation/simulation, e.g. use $\omega_0 > 0$, $0 < \gamma_1 + \omega_1 < 1$.

Diagnostic checks:

“normalized innovations” $\hat{\varepsilon}_i = x_i / \hat{\psi}_i$. QQ-plot, no correlation in $\hat{\varepsilon}_i$, SACF, Ljung Box statistic.

ACD hints

See also Tsay (2005), Analysis of Financial Time Series.

- ▶ What is θ , your vector of parameters?
- ▶ Notice the recursion you will need in the likelihood function?
- ▶ Can you generate data from the model?
- ▶ Think of the parameter restrictions
- ▶ Check whether your $\hat{\varepsilon}_i$ are indeed uncorrelated, and distributed like an exponential.

Exercise: Selecting and Plotting

- ▶ Study the file `fxukjpd.m` using OxEEdit, such that you understand what is in it
- ▶ Define a matrix

```
mD= <1991, 5, 15;  
      2003, 7, 12>;
```
- ▶ Write an Ox program with a routine that takes as input a string with the filename and the above matrix, and returns both a vector of dates and the matrix with exchange rates, between the dates in `mD`.
- ▶ Try out different options for selecting the data with combinations of the commands `dayofcalendar()`, `selectifr()`, `vecrindex()`
- ▶ Add a function taking the exchange rate returns, printing mean and standard deviation of the returns.
- ▶ Can you also select only those days within this period where the DEM/USD exchange rate is increasing by more than 1%?

Constant variances?

One can wonder if the variance of the FX returns over time are constant. To find out, compute the variance $s_i^2(k)$ over k returns r_{i-k+1}, \dots, r_i ; $i = k - 1, \dots, T - 1$.

One way to check this could be to compute

$$C_i = \sum_{j=0}^i r_j^2 \qquad s_i^2(k) = \frac{1}{k}(C_i - C_{i-k})$$

Search functions to cumulate things by yourself: You do not need *any* loop for this exercise.

As output, plot $s_i^2(k)$ against a time index, using $k = 22$ lags.

Note: Above formula assumes that the mean return is zero. What would you have to change if the mean return is not zero?

Plotting

Create a plotting function, taking as input the exchange rates and an array with the names of the exchanges rates, which

- ▶ makes four plots in one window, containing
 1. The exchange rates against a time-index (simple) or against time (more difficult)
 2. A density plot of the returns on the UK exchange rate, with a normal approximation
 3. A crossplots of the returns of the UK vs JP
 4. A QQ-plot of the UK returns against the Student- t density with 4 degrees of freedom (The variance of the student- t density is $\nu/(\nu - 2)$).
- ▶ saves the graph as an EPS file, and shows the graph on screen.

NB: As always: Try one plot at a time...