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

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 $\mathbf{U}\, x = \mathbf{b}$

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 OxEdit, run `myfirst.ox`
- Open up the help-file (press F1 in OxEdit)
- 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

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

- 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

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

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

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

and for all $x_n, \ldots, x_1$ 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`.