

Initial exercise Elim

Charles S. Bos

September 8, 2009

1 Introduction

Soon you will be following the course on *Computer Programming in Econometrics*. In order to get a bit of a headstart, you'll find attached a first complete program, written in the Ox Doornik (2006) programming language.

During the course, we will study the concepts in detail. In order to streamline the discussions, you are asked to prepare yourself studying the attached program.

2 Preparation

Section 3 contains the listing of the program. Read this program through, don't use a computer at all at this stage (maybe use a pocket calculator if you really want to).

Ask yourself e.g. the following questions:

1. Where would execution of the program start?
2. What lines are comments, which are code?
3. What is the system in the naming of the variables?
4. After line 130, the value of `mC` is

$$\mathbf{mC} = \begin{pmatrix} 10 & -7 & -4 & 28 \\ -7 & 59 & 18 & -145 \\ -4 & 18 & 58 & 56 \end{pmatrix}$$

What would its value be after line 137?

5. Matrices are indexed throughout the program. How does this work? Where does the index start?
6. The program prints some final solution. In standard econometrics, what is the problem that is solved? What would you have written on line 5?
7. This same program could have been written in some 35 lines of code (of which roughly half initialisation of `vY` and `mX`). What would be possible advantages of the present, rather extensive program, using 135 lines instead?

Think about these questions before the first class; you are not supposed to answer them all precisely and correctly, that should be easy at the end of the course. You could write for yourself some basic answers, or list your doubt where you don't see the answer. During the course, tick-off the doubts that are solved, or raise the questions with the instructors.

3 Program elim.ox

```
1  /*
2  **  elim.ox
3  **
4  **  Purpose:
5  **    ???
6  **
7  **  Date:
8  **    23/7/09
9  **
10 **  Author:
11 **    Charles Bos
12 */
13 #include <oxstd.h>
14
15 /*
16 **  TransRow(const amC, const i, const j)
17 **
18 **  Purpose:
19 **    Clean row j of element in column i
20 **
21 **  Inputs:
22 **    amC    address of existing iK x iM matrix
23 **    i      integer, number of pivot element
24 **    j      integer, number of row to sweep
25 **
26 **  Output:
27 **    amC    address of iK x iM matrix, with a zero at location j,i
28 **           as a result of the sweeping
29 **
30 **  Return value:
31 **    ir     TRUE if all well
32 */
33 TransRow(const amC, const i, const j)
34 {
35     decl dF;
36
37     if (amC[0][i][i] == 0)
38         return FALSE;
39     // Find factor multiplying row i
40     dF= amC[0][j][i] / amC[0][i][i];
41
42     // Subtract dF times row i from row j
43     amC[0][j][i:]-= dF * amC[0][i][i:];
44
45     return TRUE;
46 }
```

```

47
48 /*
49 ** ElimColumn(const amC, const i)
50 **
51 ** Purpose:
52 **   Eliminate the elements of column i, below the pivot
53 **
54 ** Inputs:
55 **   amC   address of existing iK x iM matrix
56 **   i     integer, number of pivot element
57 **
58 ** Output:
59 **   amC   address of iK x iM matrix, with zeros below location i,i
60 **         as a result of the sweeping
61 **
62 ** Return value:
63 **   ir    TRUE if all well
64 **/
65 ElimColumn(const amC, const i)
66 {
67   decl ir, j, iK;
68
69   ir= TRUE;
70   iK= rows(amC[0]);
71   for (j= i+1; j < iK; ++j)
72     ir= ir && TransRow(amC, i, j);
73
74   return ir;
75 }
76
77 /*
78 ** ElimGauss(const amC)
79 **
80 ** Purpose:
81 **   Eliminate the lower diagonal of the matrix
82 **
83 ** Inputs:
84 **   amC   address of existing iK x iM matrix
85 **
86 ** Output:
87 **   amC   address of iK x iM matrix, with zero at lower diagonal,
88 **         as a result of the sweeping
89 **
90 ** Return value:
91 **   ir    TRUE if all well
92 **/
93 ElimGauss(const amC)
94 {
95   decl iK, ir, i;
96
97   iK= rows(amC[0]);
98   ir= TRUE;
99   for (i= 0; i < iK-1; ++i)
100    {
101      println ("Starting iteration ", i);
102      ir= ir && ElimColumn(amC, i);
103      println ("resulting in ", amC[0]);

```

```

104     }
105
106     return ir;
107 }
108
109 main()
110 {
111     decl mX, vY, mA, vB, mC, ir, iN, vX;
112
113     // Inputs
114     mX= < 1, 1, 3;
115           1, -1, -3;
116           1, -4, -1;
117           1, 1, -1;
118           1, 0, 2;
119           1, 1, -2;
120           1, 2, 3;
121           1, 1, -2;
122           1, -5, 1;
123           1, -3, -4>;
124     vY= <4; -2; 12; -4; 5;
125         -6; 1; -6; 19; 1>;
126
127     // Transform inputs
128     mA= mX'mX;
129     vB= mX'vY;
130     mC= mA~vB;
131
132     print ("Initial matrix [A | b]: ", mC);
133
134     // Eliminate the mC matrix, resulting in [ mU | vC ]
135     ir= ElimGauss(&mC);
136     println ("ElimGauss returns ", ir ? "TRUE" : "FALSE",
137             " with mC= ", mC);
138 }

```

References

Doornik, J. A. (2006). *Ox: An Object-Oriented Matrix Programming Language*. London: Timberlake Consultants Ltd.