

Initial exercise E0

Charles S. Bos

v. 2020

1 Introduction

You are set to follow the course *Principles in Programming in Econometrics*. In order to get a bit of a headstart, you'll find attached a first complete program, written in the Python programming language.

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

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 129, the value of `mC` is

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

What would its value be after line 135? What would you have written on line 7, instead of the '???'?

5. Matrices are indexed throughout the program. How does this work? Where does the index start?
6. There is something special with the numerics. Where do you encounter the numerical values 0, 1 and 2? Is there a difference in the region of the program where you encounter those numbers? Why?

7. This same program could have been written in some 40 lines of code (of which roughly half initialisation of `vY` and `mX`). What would be possible advantages of the present, rather extensive program, using 142 lines instead?
8. (*More difficult*) At line 130, the matrix `mC` is changed. Why does this happen, what would have gone (hopelessly) wrong otherwise?

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 `e0_elim.py`

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  """
4  e0_elim.py
5
6  Purpose:
7      ???
8
9  Date:
10     2018/8/28
11
12 @author: cbs310
13 """
14 #####
15 ### Imports
16 import numpy as np
17
18 #####
19 ### br= ElimElement(mC, i, j)
20 def ElimElement(mC, i, j):
21     """
22     Purpose:
23     Eliminate one element [i,j] of a matrix, subtracting multiples
24     of row j from row i
25
26     Inputs:
27     mC      iK x iK+iY matrix
28     i       integer, number of row to eliminate
29     j       integer, number of row with pivot
30
31     Outputs:
32     mC      iK x iK+iY matrix, with 0 created in location [i,j]

```

```

33
34     Return value:
35     br    boolean, True if all went well
36     """
37     if mC[j,j]== 0:
38         return False
39
40     # Find factor multiplying row j
41     dF= mC[i,j] / mC[j,j]
42
43     # Subtract dF times row j from row i
44     mC[i,j:] = mC[i,j:] - dF*mC[j,j:]
45
46     return True
47
48 #####
49 ### br= ElimColumn(mC)
50 def ElimColumn(mC, j):
51     """
52     Purpose:
53     Eliminate one column [:,j] of a matrix, creating zeros below
54     the pivot at [j,j]
55
56     Inputs:
57     mC    iK x iK+iY matrix
58     j     integer, number of row with pivot
59
60     Outputs:
61     mC    iK x iK+iY matrix, with 0 created below [j,j]
62
63     Return value:
64     br    boolean, True if all went well
65     """
66     br= True
67     iK= np.size(mC, 0)
68     for i in range(j+1, iK):
69         # print ("Starting row ", i)
70         br= br and ElimElement(mC, i, j)
71         # print ("resulting in mC= \n", mC)
72
73     return br
74
75 #####
76 ### br= ElimGauss(mC)
77 def ElimGauss(mC):
78     """
79     Purpose:
80     Eliminate a matrix, creating zeros at lower triangular
81

```

```

82     Inputs:
83     mC      iK x iK+iY matrix
84
85     Outputs:
86     mC      iK x iK+iY matrix, with 0 created below main diagonal
87
88     Return value:
89     br      boolean, True if all went well
90     """
91     iK= np.size(mC, 0)
92     br= True
93     for j in range(iK):
94         print ("Starting iteration ", j)
95         br= br and ElimColumn(mC, j)
96         print ("resulting in mC= \n", mC)
97     return br
98
99     #####
100    ### main
101    def main():
102        # Magic numbers
103        mX= [ [1,   1,   3],
104              [1,  -1,  -3],
105              [1,  -4,  -1],
106              [1,   1,  -1],
107              [1,   0,   2],
108              [1,   1,  -2],
109              [1,   2,   3],
110              [1,   1,  -2],
111              [1,  -5,   1],
112              [1,  -3,  -4] ]
113        vY= [ 6,  -1, 10,  -3,  4,
114             -5,  1, -5, 19,  2]
115
116        # Transform inputs to matrices of floats
117        mX= np.array(mX)
118        iN= np.size(vY)
119        vY= np.array(vY).reshape(iN, 1)
120
121        # Prepare A= X'X, b= X'y, C= [A, b]
122        mA= mX.T@mX
123        vB= mX.T@vY
124        mC= np.hstack((mA, vB))
125        mC= mC.astype(float)
126
127        print ("Initial matrix [A | b]: \n", mC);
128
129        # Eliminate the mC matrix, resulting in [ mU | vC ]
130        ir= ElimGauss(mC)

```

```

131     print ("ElimGauss returns ir= ", ir,
132            " with mC= \n", mC)
133
134     #####
135     ### start main
136     if __name__ == "__main__":
137         main()

```