Initial exercise E0

Charles Bos

v. 2022

1 Introduction

You are set to follow the course *Principles of 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. To help in understanding those concepts, first try to form yourself an idea: Why would the program be written as it is? What parts are meaningful? What would you have done the same/different/never in a lifetime?

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 the following questions, and answer them on a piece of scratch paper/a file in wordpad:

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 127, the value of `mC` is

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

   What would its value be after line 132? 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 139 lines instead?

8. *(More difficult)* At line 127, the matrix `mC` is changed. Why does this happen, what would have gone (hopelessly) wrong otherwise?

Think about these questions *before* watching the main set of videos; 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, 2021/8/4
11
12 Author:
13     Charles Bos
14 """
15 ###############################################################
16 ### Imports
17 import numpy as np
18
19 ###############################################################
20 ### br= ElimElement(mC, i, j)
21 def ElimElement(mC, i, j):
22     """
23     Purpose:
24       Eliminate one element [i,j] of a matrix, subtracting multiples
25       of row j from row i
26
27     Inputs:
28       mC     iK x iK+iY matrix
```

```
29        i      integer, number of row to eliminate
30        j      integer, number of row with pivot
31
32      Outputs:
33        mC     iK x iK+iY matrix, with 0 created in location [i,j]
34
35      Return value:
36        br     boolean, True if all went well
37      """
38      if mC[j,j]== 0:
39          return False
40
41      # Find factor multiplying row j
42      dF= mC[i,j] / mC[j,j]
43
44      # Subtract dF times row j from row i
45      mC[i,j:]= mC[i,j:] - dF*mC[j,j:]
46
47      return True
48
49  ##############################################################
50  ### br= ElimColumn(mC)
51  def ElimColumn(mC, j):
52      """
53      Purpose:
54        Eliminate one column [:,j] of a matrix, creating zeros below
55        the pivot at [j,j]
56
57      Inputs:
58        mC     iK x iK+iY matrix
59        j      integer, number of row with pivot
60
61      Outputs:
62        mC     iK x iK+iY matrix, with 0 created below [j,j]
63
64      Return value:
65        br     boolean, True if all went well
66      """
67      br= True
68      iK= np.size(mC, 0)
69      for i in range(j+1, iK):
70          # print ('Starting row ', i)
71          br= br and ElimElement(mC, i, j)
72          # print ('resulting in mC= \n', mC)
73
74      return br
75
76  ##############################################################
77  ### br= ElimGauss(mC)
```

```
78 def ElimGauss(mC):
79     """
80     Purpose:
81       Eliminate a matrix, creating zeros at lower triangular
82 
83     Inputs:
84       mC    iK x iK+iY matrix
85 
86     Outputs:
87       mC    iK x iK+iY matrix, with 0 created below main diagonal
88 
89     Return value:
90       br    boolean, True if all went well
91     """
92     iK= np.size(mC, 0)
93     br= True
94     for j in range(iK):
95         print ('Starting iteration ', j)
96         br= br and ElimColumn(mC, j)
97         print ('resulting in mC= \n', mC)
98 
99     return br
100 
101 ################################################################
102 ### main
103 def main():
104     # Magic numbers
105     mX= [ [1,   1,   3],
106           [1,  -1,  -3],
107           [1,  -4,  -1],
108           [1,   1,  -1],
109           [1,   0,   2],
110           [1,   1,  -2],
111           [1,   2,   3],
112           [1,   1,  -2],
113           [1,  -5,   1],
114           [1,  -3,  -4] ]
115     vY= [ 6,  -1,  10,  -3,   4,
116          -5,   1,  -5,  19,   2]
117 
118     # Transform inputs to matrices of floats
119     mX= np.array(mX)
120     iN= np.size(vY)
121     vY= np.array(vY).reshape(iN, 1)
122 
123     # Prepare A= X'X, b= X'y, C= [A, b]
124     mA= mX.T@mX
125     vB= mX.T@vY
126     mC= np.hstack((mA, vB))
```

```
127     mC= mC.astype(float)
128
129     print ('Initial matrix [A | b]: \n', mC);
130
131     # Eliminate the mC matrix, resulting in [ mU | vC ]
132     ir= ElimGauss(mC)
133     print ('ElimGauss returns ir= ', ir,
134            ' with mC= \n', mC)
135
136 ################################################################
137 ### start main
138 if __name__ == "__main__":
139     main()
```