Autumn 2023

EC2B1

Macroeconomics II - Assignment 8

For week 9 classes

Guidelines

This is an excellent preparation for the course project which will ask you to do very similar programming exercises.

- The program for question 1 is not that difficult and no template to get started is given. Such templates are provided for questions 2 and 3. Those can be found in zip files on the Moodle assignment page.
- Python resources are given on the “Moodle AT Support" page.
- And a video to get you started *with the absolute basics* can be found in the course Echo recordings.
- Doing all three questions in one week is a lot of work, especially if you are a beginner. The most important thing is that you get started and get a feel of the kind of programming that will be required for the course project. And since there is no regular homework this week, we give you a bit of time for this.

Section A
(Growth Accounting)

Question 1

- Use the discrete time version of Weil.
- Assume that $\delta = 0.1$, $\alpha = 0.3$, $\gamma = 0.2$, $A = L = 1$, and $n = g = 0$.

(a) Calculate the steady state value for per capita capital, k.

(b)
- Consider a one-time temporary increase in A of 1%. Write a Python program to calculate the time path for per capita capital k and output, y.
- Consider a permanent increase in A of 1%. Write a Python program to calculate the time path for per capita capital k and output, y.
- For the last two questions, use growth accounting to determine which part of the changes in output are due to technology and which are due to changes in capital. With the answers already obtained, this is just a bit of extra code.
- What does this exercise teach you about any causal inferences you can draw about growth accounting.

Section B
(Balanced Growth)

The purpose of section B is to understand convergence to and being at the balanced growth path.

Question 2

For those who are beginners, we have provided a template in which you only have to fill in just a few steps. And the *.py template includes several comments on Python programming (the Jupyter notebook has less educational Python comments, but includes economic formulas in the code).

This question continues question 4 of homework #5.

Suppose that

$$\dot{A} = L_A^{\lambda} A^{\phi},$$

where $\dot{A} = dA/dt$. Note that we have set the value of θ equal to 1. We assume that L_A grows at rate n.
The term dt is a measurement unit that is infinitesimally small. That is something that we cannot put on a computer. But we can approximate it. For example, we can set $dt = 0.01$. We start time, t, at $t = 1$. Thus, time evolves as $[1, 1.01, 1.02, \cdots]$.

(a) Calculate and plot time paths for the growth rate $\dot{A}/A$ and the levels L_A and A for the following cases.

- $n = 0.02$, $\lambda = 1$, $\phi = 0$, $L_A(1) = 1$, and $A(1) = 1/n$.
- $n = 0.02$, $\lambda = 1$, $\phi = 0$, $L_A(1) = 1$, and $A(1) = 0.1/n$.
- $n = 0.02$, $\lambda = 1$, $\phi = 0$, $L_A(1) = 1$, and $A(1) = 50/n$.

- $n = 0.02$, $\lambda = \phi = 1/2$, and $L_A(1) = 1$ and $A(1) = (1/n)^2$.
- $n = 0.02$, $\lambda = \phi = 1/2$, and $L_A(1) = 1$ and $A(1) = 0.1(1/n)^2$.
- $n = 0.02$, $\lambda = \phi = 1/2$, and $L_A(1) = 1$ and $A(1) = 50(1/n)^2$.

(b) What are the checks you can perform to convince yourself that your program is correct?

(c) Can you intuitively and clearly explain the behavior of A and its growth rate for the different initial conditions?

(d) There is a striking difference between the case with $\lambda = 1$ and $\phi = 0$ and the case with $\lambda = \phi = 1/2$. What is that difference and what is the explanation?

(e) **A somewhat harder exercise.** Suppose that

$$\dot{A} = L_A^{\lambda} A^{\phi}, \tag{1}$$

where $\dot{A} = dA/dt$. Note that we have set the value of θ equal to 1. We assume that L_A grows at rate n equal to 0.02, $\phi = 0$, and $\lambda = 1$.

Suppose the economy is on a balanced growth path. Suddenly λ drops to 0.9. We know that this will lead to a reduction in the steady state growth rate of A. But it is possible that the growth rate (i) first increases, (ii) stays the same on impact and then falls, and (iii) falls on impact. Explain. Hint: This is not a deep question and no complicated calculations or algebra are needed. But feel free to check your answer with the program you wrote for part C of this assignment.

Section C

(Fitting Trends)

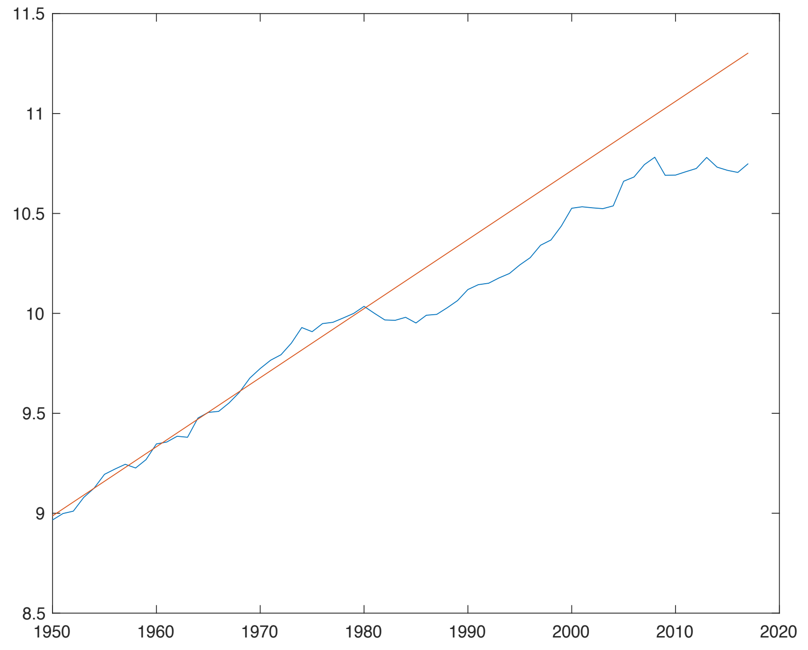
The purpose of section C is to calculate long-term growth paths. Recall that we abstract from business cycle fluctuations in the theories you study in EC2B1. But the data have both a long-term growth component and a business cycle component. In this exercise, you learn how to calculate the trend component. That is, that part of the data that growth theories try to explain.

Question 3

Purpose. These types of Python exercises will not only teach you transferable programming skills. They will also help you to start thinking about economic questions by looking at data, a skill that will be useful in many jobs and research projects. The material of this question is not examinable, but it will help you to become a more mature economist/thinker which could help you with other types of questions and challenges and specifically the course project.

Background and motivation. Especially after a (severe) economic downturn, it is common practice to compare an economy's GDP development with the counterfactual under which the economy would have continued to grow *like it did on average before the downturn*. In particular, there is typically a lot of interest on whether the economy get's back to the old trend path or not. If it does, then there is no permanent level effect. But it is also possible that the economy will never get back to the old growth path. And it is even possible that the downturn has a long-lasting impact on the *growth* rate.

An example is given in the figure above which considers Dutch GDP per capita. Its purpose is to answer the question *how realized GDP per capita post 1980 compares to the associated trend values when the trend would have continued to grow post 1980 the way it did before 1980.* It plots GDP per



capita for the Netherlands and a trend line based on behavior of GDP per capita up to 1980. By construction, the fitted trend line captures the trend during the first three decades well because those data were used to “fit" the trend line. It also shows that actual GDP per capita fell far short of this trend path in subsequent decades. One could, of course, reestimate the trend line using all data, not just the first three decades. But that would not answer the question the figure is supposed to answer.

The question arises how to estimate the trend growth path. Let $\widehat{y}_t$ denote the trend value of per capita GDP. A common way to proceed is to assume that $\widehat{y}_t$ follows *exponential* growth with a constant growth rate. That is,

$$\widehat{y}_t = \widehat{y}_0 e^{gt}, \tag{2}$$

where $\widehat{y}_t$ is the trend value of y_t and g is the growth rate. With the following regression equation we can estimate the growth rate.

$$\ln(y_t) = a + bt + u_t. \tag{3}$$

If you use this regression equation, then b would be the estimate for the growth rate g.

A somewhat richer specification would be

$$\ln(y_t) = a + b_1 t + b_2 t^2 + u_t. \tag{4}$$

Now the trend growth rate would no longer be constant. It would be given by

$$\ln\left(\frac{y_t}{y_{t-1}}\right) = a + b_1 t + b_2 t^2 - (a + b_1(t-1) + b_2(t-1)^2) = b_1 + 2b_2 t - b_2. \tag{5}$$

But are we sure that exponential growth (possibly modified with higher-order terms) is the best approach? A paper by Thomas Philippon has argued that for TFP an *additive* approach with constant increments may provide a better approach. If this would be true for GDP per capita, then this would imply the following regression model (for the linear specification):

$$y_t = a + bt + u_t. \tag{6}$$

Note that the dependent variable is the level of GDP per capita, NOT the log. Both specifications predict that GDP per capita will continue to grow (as long as $b > 0$), but the specification in equation (6) implies that the *growth* rate would decrease to zero. However, one note of caution. Philippon only shows that for available data sets, which are finite, an additive model may at times provide a better representation. That does not mean that the additive model will be the better one if we use this model into the very far future, i.e., when $t \to \infty$.

Out of sample evaluation. So the main idea is that you "fit" different trend models over a particular sample and then investigate whether actual GDP per capita continues to behave according to this trend or not. What this question teaches you is that there isn't one way to model the trend and you should be aware of this.

A note about the programs. On Moodle, you will find the Python program called *GDP_trend.py*. This is a template for the main program. This is a standard Python program. If you prefer to use the Jupyter notebook, then you should use *GDP_trend.ipynb* which has also been provided. Instead of you writing a program from scratch, you just have to fill in the key bits. Moreover, we provide you with instructions in this template. So we start easy! A key part of the exercise is to run a regression like you have learned in EC1C1. This is done in a separate program which is then used in the main program. The name of this regression program is called *get_regression_coefs.py*. If this file is stored in the same directory, then the main program will recognize it. *get_regression_coefs.py* is already complete.

What you should do specifically.

- The file *pwt100.xls* contains data from the Penn World Tables. Decide which country you want to study. You should first check the Excel file yourself and do a first check on whether there are no problems with the data series. For example, are enough data available? In the main program *GDP_trend.py* you will find instructions on how to select your chosen country and the variables you need to construct GDP per capita.
- Write a program that does the following.
  - Make a plot of the raw data. You should NEVER do any empirical work without having carefully looked at the raw data. You should not continue if something looks weird.
  - Fit both the linear exponential and the linear additive model using data up to 2006, that is, before the start of the financial crisis. You will have to use some judgement on when to start your sample but use at least 20 years. In general a longer sample is better when estimating coefficients. However, if the economy you are interested in underwent big changes in the decades before 2006, then a shorter sample may be better. That is, you have to decide which decades before 2006 will be more representative as a benchmark.
  - When you have obtained your estimates of the regression coefficients, then evaluate the behavior of GDP per capita for the post-2006 period relative to the estimated benchmark. Do NOT reestimate the model. After all, the objective is to see how actual post-2006 behavior compares with the counterfactual when the economy would have experienced trend growth equal to that of the pre-2007 period.
  - Does your answer to the last question depend on the particular model to characterize the trend?
  - Feel free to also try quadratic specifications.

- Suppose that GDP per capita post 2006 is closer to a higher-order trend specifciation than a linear one. Does that mean that the linear specification is necessarily worse?

If you are up for a REALLY big programming challenge and want to learn more econometrics.

It is fine to use the provided *get_regression_coefs.py* which is complete. And you can also use this for the course project. However, some of you may want to know what this program does and challenge yourself with an additional programming exercise. If you do, then we get you started with *get_regression_coefs_incomplete.py*. Of course, when you get stuck, you can look at *get_regression_coefs.py* to see what the program should look like. To understand *get_regression_coefs.py*, you will have to learn a little bit more econometrics. The reason is the following. In EC1C1 you learned that a regression is a minimization problem. So that would require using a minimization routine which is a bit tricky and software like STATA does not do that. There is a different approach that does NOT require minimization, but gives you the exact same regression results. Understanding that requires a bit more econometrics and relies matrix algebra. It is perfectly fine to ignore all this for now and simply use *get_regression_coefs.py*. But if you are up for a challenge, then we spell out below how it is done.

Programming it yourself. Here we will focus on a generic regression specification with only a constant and one regressor, but everything goes through for specifications with more regressors. Thus, we have

$$y_t = a + bx_t + u_t = \begin{bmatrix}1 \; x_t\end{bmatrix}\begin{bmatrix}a\\ b\end{bmatrix} + u_t. \tag{7}$$

How to find good values for a and b? One possible choice is to let the estimates for a and b be the values that give the *best fit*. This is what you learned in EC1C1. There is an alternative motivation which actually leads to the exact same numerical outcomes. The alternative motivation starts with the following two assumptions.

1. Assumption: The expectation of u_t is equal to zero. This is a sensible assumption when the regression equation also has a constant.
2. Assumption: The expectation of $u_t x_t$ is also equal to zero. Since the mean of u_t equals zero this means that u_t and x_t are not correlated with each other. If u_t and x_t are correlated, then the estimate of b will not just capture the "pure" effect of x_t on y_t but also the impact on y_t of the bits in u_t that are correlated with x_t.

The estimates that we get based on these assumptions are identical to the ones based on the minimization approach. So you will get a good fit whether these assumptions are true or not. But if these assumptions are true, then we do not only get a good fit. We also get that b measures the true impact of x_t on y_t.[1]

Okay, let's see how we can get estimates based on these two assumptions. When we pre-multiply both sides of our regression equation with $\begin{bmatrix}1\\ x_t\end{bmatrix}$, then we get

[1]The following simple example makes this clear. Suppose that the true model is as follows.

$$y_t = a + bx_t + dz_t + u_t^*. \tag{8}$$

So the error term in the original regression equation, u_t, is equal to $dz_t + u_t^*$. If x_t and z_t are correlated then u_t and x_t are correlated. This means that the estimate of b will not just capture the true impact of x_t on y_t but also capture the effect of z_t on y_t. An extreme example would be the case when $b = 0$. The estimate of b will not be equal to zero since x_t will be an (imperfect) measure of the variable that does affect y_t, namely z_t.

$$\begin{bmatrix} 1 \\ x_t \end{bmatrix} y_t = \begin{bmatrix} 1 \\ x_t \end{bmatrix} \begin{bmatrix} 1 \; x_t \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} + \begin{bmatrix} 1 \\ x_t \end{bmatrix} u_t. \tag{9}$$

When we take expectation of both sides and use our regression assumptions, then we get

$$\mathbb{E}\left[\begin{bmatrix} 1 \\ x_t \end{bmatrix} y_t\right] = \mathbb{E}\left[\begin{bmatrix} 1 \\ x_t \end{bmatrix} \begin{bmatrix} 1 \; x_t \end{bmatrix}\right] \begin{bmatrix} a \\ b \end{bmatrix} \tag{10}$$

or

$$\begin{bmatrix} \mathbb{E}[y_t] \\ \mathbb{E}[x_t y_t] \end{bmatrix} = \begin{bmatrix} \mathbb{E}[1] & \mathbb{E}[x_t] \\ \mathbb{E}[x_t] & \mathbb{E}[x_t x_t] \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \tag{11}$$

Is it key that you understand that this is a system of two equations in the two unknowns, a and b. The idea behind regression analysis is to replace the expectations by their sample means. That is,

$$\begin{bmatrix} \Sigma y_t/T \\ \Sigma x_t y_t/T \end{bmatrix} = \begin{bmatrix} \Sigma 1/T & \Sigma x_t/T \\ \Sigma x_t/T & \Sigma x_t x_t/T \end{bmatrix} \begin{bmatrix} \widehat{a} \\ \widehat{b} \end{bmatrix}. \tag{12}$$

Note that as soon as the population moments are replaced by their sample counterparts, that the true coefficients a and b are replaced by the associated estimates, $\widehat{a}$ and $\widehat{b}$. Multiplying both sides with T gives[2]

$$\begin{bmatrix} \Sigma y_t \\ \Sigma x_t y_t \end{bmatrix} = \begin{bmatrix} \Sigma 1 & \Sigma x_t \\ \Sigma x_t & \Sigma x_t x_t \end{bmatrix} \begin{bmatrix} \widehat{a} \\ \widehat{b} \end{bmatrix}. \tag{13}$$

Here is the beautiful part. With matrix algebra, things are going to be very easy to program. Let Y denote the $(T \times 1)$ vector with the T observations for y_t, and let X be the following matrix

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & X_3 \\ \vdots & \vdots \\ 1 & x_t T \end{bmatrix}.$$

Then we have

$$X'Y = \begin{bmatrix} \Sigma y_t \\ \Sigma x_t y_t \end{bmatrix} \tag{14}$$

and

$$X'X = \begin{bmatrix} \Sigma 1 & \Sigma x_t \\ \Sigma x_t & \Sigma x_t x_t \end{bmatrix} \begin{bmatrix} \widehat{a} \\ \widehat{b} \end{bmatrix} \tag{15}$$

Combining gives

[2]These two conditions are the first-order conditions of the minimization approach. This is the reason why this approach gives the exact same answers as the one based on getting the best fit with a minimization routine.

$$X'Y = X'X \begin{bmatrix} \widehat{a} \\ \widehat{b} \end{bmatrix} \tag{16}$$

from which we get

$$\begin{bmatrix} \widehat{a} \\ \widehat{b} \end{bmatrix} = \left(X'X\right)^{-1} X'Y. \tag{17}$$

The formulas are the same when we have more regressors. For example, when the regression model is

$$y_t = a + b_1 t + b_2 t^2, \tag{18}$$

then the X matrix is

$$X = \begin{bmatrix} 1 & 1 & 1^2 \\ 1 & 2 & 2^2 \\ \vdots & \vdots & \vdots \\ 1 & T & T^2 \end{bmatrix} \tag{19}$$

END OF PAPER