LSE Macroeconomics Summer Program
Part II: Heterogeneous Agents
Instructor: Wouter J. Den Haan

Thursday Assignment

1 Objective

This assignment deals with frontier research for two reasons. First, it deals with *learnable* sunspots in RBC-type models. The lack of such cases is referred to as the stability puzzle in the sunspot literature.[1] In this assignment, we focus on a recent example of a model in which the sunspot *is* learnable. Second, we investigate sunspots using *nonlinear* approximation methods. To the best of my knowledge, sunspots have only been analyzed using linear approximations and never before with *nonlinear* approximation methods. This is true both for the analysis of sunspots themselves and for the question whether sunspots are learnable.

2 Model

The model is from McGough, Meng, and Xue (2011) (MMX from now on) and the notation used jere follows this paper closely. The model is quite close to a standard RBC model.

Household problem. The first-order conditions for the household are given by

$$\begin{aligned} C_t^{-\nu} &= \mathrm{E}_t\left[\rho\left(1-\delta+R_{t+1}\right)C_{t+1}^{-\nu}\right], && (1)\\ \Lambda_H H_t^{\chi} &= W_t C_t^{-\nu}, && (2)\\ C_t+K_{t+1} &= W_t H_t + R_t K_t + (1-\delta)K_t. && (3) \end{aligned}$$

The standard notation is used, except note that ρ (not β) is the discount factor. These are standard first-order conditions.

Firm problem. The firm's first-order conditions are given by

$$\begin{aligned} R_t &= a A_t K_t^{a-1} H_t^{b}, && (4)\\ W_t &= b A_t K_t^{a} H_t^{b-1}. && (5) \end{aligned}$$

[1]Stability refers to the mapping from the perceived law of motion to the actual law of motion, the T mapping, and stability of this mapping at the rational expectations equilibrium (with or without sunspots) implies learnability. If the T mapping is nonlinear, then the focus is on *local* stability of the T mapping and learnability starting with beliefs that are *close* to the rational expectations equilibrium. The T mapping is usually nonlinear, even when the underlying model is linear.

Productivity. The model differs from the prototype RBC model in the specification of productivity, A_t. Here, A_t is endogenous and given by

$$A_t = \Lambda_A \overline{K}_t^{\alpha-a} \overline{H}_t^{\beta-b}, \tag{6}$$

where a bar over a variable indicates that it is an aggregate variable and thus taken as given by the agents. Λ_A is a scalings coefficient.

Equilibrium. Combining the equations above results in the following system:

$$\begin{aligned} C_t^{-\nu} &= \mathrm{E}_t\left[\rho\left(1-\delta+a\Lambda_A K_{t+1}^{\alpha-1} H_{t+1}^{\beta}\right) C_{t+1}^{-\nu}\right], && (7)\\ \Lambda_H H_t^{\chi} &= b\Lambda_A K_t^{\alpha} H_t^{\beta-1} C_t^{-\nu}, && (8)\\ C_t + K_{t+1} &= \Lambda_A K_t^{\alpha} H_t^{\beta} + (1-\delta) K_t. && (9) \end{aligned}$$

If $a = \alpha$ and $b = \beta$, then this would correspond to a standard RBC model. If these conditions do not hold, then there are externalities.

Steady state. The steady state values can be solved from

$$\begin{aligned} 1 &= \rho\left(1-\delta+a\Lambda_A K_{ss}^{\alpha-1} H_{ss}^{\beta}\right), && (10)\\ \Lambda_H H_{ss}^{\chi} &= b\Lambda_A K_{ss}^{\alpha} H_{ss}^{\beta-1} C_{ss}^{-\nu}, && (11)\\ C_{ss} &= \Lambda_H K_{ss}^{\alpha} H_{ss}^{\beta} - \delta K_{ss}. && (12) \end{aligned}$$

This is a nonlinear system. Although not that difficult to solve numerically, there is an easy trick to solve for the steady state *without* modifying the dynamic properties of the system. In particular, we set $K_{ss} = H_{ss} = 1$. To make the Euler equation hold for this outcome, we adjust Λ_A. The budget constraint gives C_{ss}. To make the intratemporal first-order condition hold, we adjust Λ_H. It is important to understand that you can only use this trick by adjusting coefficients that are scalings coefficients like Λ_A and Λ_H.[2] If you adjust other coefficients, then the dynamic properties of the system are likely to change.

Linearized system. After substituting out H_t, we are left with a system of two equations in C_t and K_{t+1}. The linearized version is given by

$$\begin{aligned} \widetilde{c}_t &= b_k \widetilde{k}_{t+1} + b_c \mathrm{E}_t\left[\widetilde{c}_{t+1}\right], && (13)\\ \widetilde{k}_{t+1} &= d_k \widetilde{k}_t + d_c \widetilde{c}_t. && (14) \end{aligned}$$

A lower-case variable with a tilde indicates log-deviation from steady state. The last equation can also be written as

$$\widetilde{c}_t = b_k \widetilde{k}_{t+1} + b_c\left(\widetilde{c}_{t+1} - \zeta_{t+1}\right). \tag{15}$$

[2] That is, changing λ_A is like a change in the unit in which the amount of commodities are measured and changing λ_H is like a change in the unit in which time is measured.

According to the model, ζ_{t+1} is a prediction error. If there are exogenous random variables in the system, then innovations to these would affect ζ_{t+1}. Sunspot variables can also affect the forecast error. Since there are no exogenous random variables in the system, non-zero value of ζ_{t+1} can only be due to sunspots.

3 Sunspots

If the model does not have to satisfy the condition that $a = \alpha$ and $b = \beta$, then it is easy to find sunspots. The solution, including the sunspot, can be represented as

$$\begin{aligned} \begin{pmatrix} \widetilde{k}_{t+1} \\ \widetilde{c}_{t+1} \end{pmatrix} &= J \begin{pmatrix} \widetilde{k}_t \\ \widetilde{c}_t \end{pmatrix} + \begin{pmatrix} 0 \\ \zeta_{t+1} \end{pmatrix}, \\ J &= \begin{pmatrix} 1 & 0 \\ b_k & b_c \end{pmatrix}^{-1} \begin{pmatrix} d_k & d_c \\ 0 & 1 \end{pmatrix}, \end{aligned}$$

where ζ_{t+1} is the sun spot variable.

Note that getting the policy function is trivial. You simply solve the system forward. You don't need any perturbation analysis at all. For more info see the slides, but remember that both eigenvalues of J have to have a modulus less than 1 for this representation to be a non-explosive solution.

4 Learnable sunspots

The literature typically focuses on positive externalities for capital. The insight of the MMX paper is that learnable sunspots do exist, but only for *negative* externalities, i.e., when $\alpha < a$. One interpretation of a negative externality is that higher volume lowers productivity because of competition.

In this assignment, we ask the question whether a sunspot solution can be learned in a nonlinear approximation of the model as well. It is important to be clear about the timing when analyzing models with learning. The timing worked out explicitly by MMX is one in which agents use $\widetilde{k}_{t-1}$ and $\widetilde{c}_{t-1}$ to make forecasts. Here we start with this timing but also use the more typical timing in which both $\widetilde{k}_t$ and $\widetilde{c}_{t-1}$ are used.[3]

What can be learned. Recall that you cannot learn which sunspot to learn. That is, the model builder has to decide which sunspot to put in. This is true in a linear and in a nonlinear setting.

In the linearized setting, you also cannot learn the importance of the sunspot. That is, the initial value of the coefficient measuring the impact of the sunspot is fixed to an arbitrary value and the only question is whether these initial

[3]Recall that K_t is the beginning-of-period capital stock, so is a predetermined period-t variable.

beliefs are confirmed. We will investigate whether this limitation also holds in the nonlinear framework.

Adaptive learning in nonlinear setings. One more comment before we describe the algorithm. As in MMX we focus on adaptive learning. That is, we specify a perceived law of motion and then determine the actual law of motion of this economy based on population moments.[4] This derived actual law of motion is then used to update the perceived law of motion. In the linearized world, such population moments are trivial to obtain and one can easily derive the mapping from the perceived to the actual law of motion analytically. Here we will use a long regression to obtain the actual law of motion.

5 The algorithm

The algorithm is a relatively straightforward implementation of parameterized expectations (PEA). The key step of PEA is to replace the conditional expectation in the Euler equation with a nonlinear function of the period t state variables, $h(S_t;\eta)$, where η are the coefficients of $h(\cdot)$. To explain the details of the algorithm we set

$$\begin{aligned} h(S_t;\eta) &= \exp\left\{\eta_0 + \eta_k \widetilde{k}_{t-1} + \eta_c \widetilde{c}_{t-1} + \eta_\zeta \zeta_t\right\} \qquad (16) \\ &\approx \mathrm{E}_t\left[\rho\left(1-\delta + a\Lambda_A K_{t+1}^{\alpha-1} H_{t+1}^{b}\right) C_{t+1}^{-\nu}\right]. \qquad (17) \end{aligned}$$

This can be viewed as a first-order approximation. The attractive part of this functional form is that it corresponds with the log-linear formulation of MMS. To be more precise, if we set

$$\begin{aligned} \eta_0 &= -\nu \ln C_{ss}, \\ \eta_k &= -\nu J_{21}, \\ \eta_c &= -\nu J_{22}, \\ \eta_\zeta &= -\nu, \end{aligned}$$

then we get exactly the log-linear sunspot solution of MMX for the case when the sunspot coefficient is equal to 1.

5.1 Steps of the algorithm

The algorithm consists of an iterative procedure. It starts with an initial guess for the values of η, η^1, where the vector η consists of η_0, η_k, and η_c and not η_ζ. The solution from the log-linearized system can be used as the initial condition. The algorithm then iterates on the following system:

[4] In contrast, with recursive learning you use sample moments that are updated each period. This is probably the more interesting case, but it is better to start with adaptive learning which is easier to implement.

1. Given a value for η^i, generate a long time series for C_t, K_{t+1}, and H_t using

$$\begin{aligned}
C_t^{-\nu} &= \exp\left\{\eta_0^i + \eta_k^i \widetilde{k}_{t-1} + \eta_c^i \widetilde{c}_{t-1} + \eta_\zeta \zeta_t\right\}, && (18)\\
\Lambda_H H_t^{\chi} &= b\Lambda_A K_t^{\alpha} H_t^{\beta-1} C_t^{-\nu}, && (19)\\
C_t + K_{t+1} &= \Lambda_A K_t^{\alpha} H_t^{\beta} + (1-\delta) K_t, && (20)\\
& \text{where} && (21)\\
\widetilde{k}_t &= \ln\left(\frac{K_t}{K_{ss}}\right), && (22)\\
\widetilde{c}_t &= \ln\left(\frac{C_t}{C_{ss}}\right). && (23)
\end{aligned}$$

   η_ζ does not have an i superscript because—as mentioned above—it is not identified and thus kept fixed. Let

$$Y_{t+1} = \rho\left(1 - \delta + a\Lambda_A K_{t+1}^{\alpha-1} H_{t+1}^{\beta}\right) C_{t+1}^{-\nu}. \qquad (24)$$

2. $h(S_t; \eta)$ is supposed to be an approximation to the conditional expectation. So it makes sense to get coefficients η from the following regression problem:

$$\begin{aligned}
&\widehat{\eta}_0, \widehat{\eta}_k, \widehat{\eta}_c \\
&= \qquad\qquad (25)\\
&\arg\max_{\eta_o,\eta_k,\eta_c} \sum_{t=1}^{T} \left(Y_{t+1} - \exp\left\{\eta_0 + \eta_k \widetilde{k}_{t-1} + \eta_c \widetilde{c}_{t-1} + \eta_\zeta \zeta_t\right\}\right).
\end{aligned}$$

   Again note that η_ζ is not estimated. The $\widehat{\eta}$ coefficients characterize the actual law of motion.

3. The vector of coefficients characterizing the perceived law of motion in the next iteration is given by

$$\eta^{i+1} = (1-\omega)\,\widehat{\eta} + \omega \eta^i, \qquad (26)$$

   where ω is a dampening factor.

Comment about the regression. Note that the coefficients are obtained using NLLS. It is not possible to take the logs of both Y_{t+1} and $h(\cdot)$ and turn it into a linear regression, because the corresponding error terms would not have the necessary properties.

5.2 Higher-order version

If we would let $h(\cdot)$ be a higher-order approximation, then the algorithm would be exactly the same. Similarly, the algorithm is the same if we move away from the MMX choice for S_t and let S_t consists of the more natural elements, namely K_t, C_{t-1}, and ζ_t.

5.3 Alternative functional form

Consider the following alternative functional form for $h(\cdot)$:

$$h(\widetilde{k}_{t-1}, \widetilde{c}_{t-1}, \zeta_t) = h_S(\widetilde{k}_{t-1}, \widetilde{c}_{t-1}) + \eta^* \sin\left(h_\zeta\left(\widetilde{k}_{t-1}, \widetilde{c}_{t-1}, \zeta_t\right)\right). \tag{27}$$

To avoid clutter we explain the idea using a linear approximation. As above, the idea of the algorithm easily extends to higher-order approximation. In particular, let

$$h(\widetilde{k}_{t-1}, \widetilde{c}_{t-1}, \zeta_t) = \exp\left\{\eta_0 + \eta_k \widetilde{k}_{t-1} + \eta_c \widetilde{c}_{t-1}\right\} + \eta^* \sin\left(\frac{\eta_\zeta \pi}{2}\zeta_t\right). \tag{28}$$

The parameter η^* is kept fixed and not pinned down by either the model or the learning of agents. In this particular example, the second part of the approximation does not depend on $\widetilde{k}_{t-1}$ and $\widetilde{c}_{t-1}$. The additional sunspot part is then not correlated with the other explanatory variables. In practice you would have to check whether this is simplification is accurate or not, but a key aspect of functional analysis is to limit terms to include and $\widetilde{k}_{t-1}$ and $\widetilde{c}_{t-1}$ are already included in the first part of the approximation.

In contrast to the analysis above, η_ζ is a free parameter. To understand the role of η^* and η_ζ consider the case in which ζ_t can take on two values, -1 and 1, both with equal probability. The maximum absolute value of the sinus is equal to 1, so η^* measures the maximum possible impact of the sunspot. The parameter η_ζ determines whether this maximum impact will be attained. The attractive feature of this case with discrete support is that it is easy to see when this maximum impact is attained. This occurs when η_ζ is equal to 1. If η_ζ is equal to 0, then the sunspot has no impact at all. In the assignment, you are asked to investigate to which value η_ζ converges.

6 Exercise #1

6.1 Overview

In this exercise we derive a solution using the first-order approximation given in equation (16). The functional form implies a law of motion for consumption that has the exact same functional form as the one used in MMX.

Nevertheless there are important differences. First, although the law of motion for consumption is restricted to be a log-linear approximation, the law of motion for capital and labor are derived using the true nonlinear equations of the model. Second, the coefficients of the approximation are derived using a projection method that is based on simulated data, not on the derivatives around the steady state.

For this exercise you need the following programs:

1. the main program: `sunspotlearning_linear1.m`
2. function to define RSS: `rssfcn.m`

The structure of `sunspotlearning_linear1.m` is as follows:

1. set parameter values algorithm
2. set parameter values model & steady state
3. derive linearlized solution
4. generate sunspot shocks and time series with linearized solution
5. generate solution with PEA

6.2 For you to do

1. In terms of programming there is little for you to do. The only thing needed to do is to complete the part that generates the time series in part 5 (indicated with XYZ in the program).
2. Run the program using the log-linearized solution of MMX as the initial value. You'll see that the PEA algorithm wants to move away from this solution, but not very much. Make some plots to understand that the PEA and the log-linearized solution are quite close to each other, but do not exactly coincide.
3. Now increase the standard deviation of the sunspot shock to 0.03 and check again how much difference there is between the two approximations.

7 Exercise #2

7.1 Overview

In this exercise, we will use expressions of the form given in equation (27) and we will check for accuracy.

For this exercise you need the following programs:

1. the main program: `sunspotlearning_sinus3.m`
2. function to define RSS: `rssfcn_sinus.m`
3. programs to do (numerical) integration: `numi.m`, `hernodes.m`, and `numi_case1.m`
4. programs to calculate expression inside conditional expectation for either a shock or a quadrature node: `RHSrealization*.m`.

The structure of the main program hasn't changed except that two modules have been added to check for accuracy. This program considers 4 cases:

1. first-order approximation for non-sunspot; first-order approximation for sunspot part of approximation; MMX timing; discrete shocks

2. same but shocks have continuous support
3. same as 2 but standard timing
4. higher-order approximation for non-sunspot; higher-order approximaiton for sunspot part; standard timing.

7.2 For you to do.

1. set:
   (a) version to `case1`
   (b) initial value to `linear`
   (c) accuracylinear to `no`
   (d) accuracyprojection to `no`
2. Set the initial value for η_ζ (beginning of part 6) to 0.9.
3. Run the program. To which value does the sunspot coefficient converge? (Convergence is not that strict here so you can run programs a bit quicker, but without time constraint you would want to tighten the convergence criterion)
4. Reset the initial value for η_ζ to 0.3. What does the solution converge to? Does it still make sense to run regressions at the point where it is converging to?
5. Run accuracychecks by setting the accuracyoptions equal to `yes`. The program calculates dynamic Euler equation tests. These differ from standard Euler equation tests in two aspects. First, it calculates errors on a simulated time path. Second, and more importantly, it takes the accurately calculated values to update the state variables so that it can check whether errors are systematic and accumulate. The graphs plot the time path according to the approximation and the one implied when conditional expectations are calculated explicitly.
6. Is any of the two solutions accurate?
7. Move to case 2. Set the initial value for η_ζ (beginning of part 6) to 0.9 and find a solution.
8. Make a plot of the series generated using the log-linearized solution and the first-order PEA solution. Do they look as if they are approximations to the same sunspot REE? They do not. Does this necessarily mean that at least one of them is not accurate?
9. Again do accuracy tests.

10. Save generated series for consumption, investment, and output to a file for later use.

11. Move to case 3. In nonlinear problems it is important to get good initial conditions. We could try the log-linearized values again, but these are not quite appropriate anymore since we have a different explanatory variable, K_t instead of K_{t-1}. Think of a way to use your solution to case 2 to get good initial conditions for case 3. Hint: don't do algebra but think of a simple thing to do on the computer. To use these in the program set initialvalues to `linear` and at the beginning of part 6 under case 3 use these initial values.

12. Again do accuracy tests.

13. Save generated series for consumption, investment, and output to a file for later use.

14. Finally, we are going to look at higher-order approximations. In particular, we will use

$$h(\widetilde{k}_{t-1}, \widetilde{c}_{t-1}, \zeta_t) = \exp\left\{\begin{array}{c} \eta_0 + \eta_k \widetilde{k}_{t-1} + \eta_c \widetilde{c}_{t-1} \\ +\eta_{k^2}\left(\widetilde{k}_{t-1}\right)^2 \\ +\eta_{c^2}\left(\widetilde{c}_{t-1}\right)^2 \\ +\eta_{ck}\widetilde{c}_{t-1}\widetilde{k}_{t-1} \end{array}\right\} + \eta^* \sin\left(\begin{array}{c} \eta_\zeta \zeta_t \\ +\eta_{\zeta^2}\zeta_t^2 \\ +\eta_{\zeta k}\zeta_t \widetilde{k}_{t-1} \end{array}\right)$$

    Use the procedure outlined in 11 to get initial conditions.

15. Get a solution and do accuracy tests. Do the accuracy tests indicate that higher-order terms are necessary?

16. Compare the time paths generated using case 2, case 3, case 4 and the log-linearized approximation. Again address the question whether these are approximations to the same REE.