Dynare Summer School - Day 2 Assignments

Wouter J. DEN HAAN

July 4, 2008

1 Preliminaries & Overview

The idea of this assignment is to use Dynare to obtain the policy functions and to program the rest yourself. Some of the things can be done by Dynare but they are easy to program yourself and this way it will hopefully become more clear what you are doing

1.1 How to simulate yourself

I like to use the set of policy coefficients *exactly* the way the way Dynare reports them on the screen. I rewrote the file disp_dr.m so that it write this matrix with coefficients to the matlab data file dynarerocks.mat (in the same directory as your *.mod file). To get the coefficients back into memory you only have to type

load dynarerocks

This matlab files dynarerocks contains both the name "decision" and its content. With the load command you put both the name and its content into memory. If you want to use this option you have to replace the file disp_dr.m with the file provided.

For modelc.mod the matrix decision looks like

3.6373	0	1.0132
0.9815	0.0000	0.3940
0.0681	0.9500	0.3379
0.0717	1.0000	0.3557

If you put the following commands in a Matlab file you can now simulate a data set

```
dynare modelc                    % this solve the model
load dynare rocks                % reloads policy coefficients decision back into memory
iiicap=1;                        % iiicap is column in decision corresponding to capital
iiicons=3;                       % same for consumption
T=1000;                          % number of observations
cap = zeros(T,1);                % initializes capital values as a column vector
prod = zeros(T,1);               % same for productivity
shocks = randn(T,1);             % generates
cap(1)=decision(1,iiicap);       % sets initial capital value to steady state value
prod(2)=0                        % sets initial prod value used to steady state value
for t = 2:T                      % start the key do loop
prod(t)= ...
rho*prod(t-1)+sig*shocks(t)
cap(t)=decision(1,iiicap)+decision(2,iiicap)*(cap(t-1)-decision(1,iiicap)) ...
        decision(3,iiicap)*prod(t-1) + decision(4,iiicap)*shocks(t)
end
plot(exp(cap))
```

Note that dynare does keep the parameter values in memory, which is why I could use them after running Dynare without redefining them.

Simulating solutions for second-order pertubation. Be careful in simulating series with the second-order perturbation. In particular,

1. The constant of the policy rule is the first coefficient. The second is not part of the policy rule (but gives the gap with the steady state).

2. Recall that the explanatory variables are in deviation of the steady state. For first-order that is simply the first coefficient. For second-order that is the first coefficient minus the second coefficient.

1.2 Looping

Suppose you want to do the exercise above twice for different values of risk aversion, nu. Then you can do the following.

1. Replace the command "nu = 3" with

   load wouterrocks

   set_param_value('nu',nu);

   The matlab file wouterrocks has both the name "nu" and its value. Of course, you have to make sure that you store the right value of nu into the file wouterrocks before running Dynare.

2. In your Matlab program loop over the different values of nu. In each iteration, first save the current value of nu to the file wouterrocks with

   "save wouterrocks nu

   and *then* run Dynare

3. To make comparison easy you may want to use the same shocks in each iteration. So either generate the shocks outside the loop or reset the seed in each loop using the command "randn('state',20070417)"

4. If you want to keep results generated in an iteration you have to save it (I think)

1.3 Overview

1. The first exercise looks at the mean of generated series for different Dynare solutions. Getting the mean right is actually not such an easy exercise. Note that we are often also not that interested in the mean. Usually we are interested in second-order properties of the data and these are easier to get.

2. This exercise performs accuracy tests for the growth model of modelc.mod.

3. This exercise focuses on the matching model. It does two things

   (a) calculates IRFs for positive and negative shocks at different points in the state space

   (b) calculates accuracy tests again

2 Assignment 1

This assignment is about the neoclassical growth model, programmed in modelc.mod. The objective is to calculate the mean capital and consumption level for different parameter values and for different solution procedures. Note that we are interested in the mean of the *level* of capital and consumption, not in the mean of the logs.

Preliminaries. Set sig equal to 0.1. This is quite high but this way you get some action.

Exercise. Calculate the mean of capital and consumption by simulating yourself after you have used Dynare to solve for the coefficients of the policy function. Do it for the following settings.

1. Vary the coefficient of relative risk aversion from 0 to 15. (You can use the loop structure described above and possibly you may want to put in a pause statement to look at the results step by step)

2. Calculate these means for the following algorithms:

   (a) log-linear (as in modelc.mod)

   (b) linear in levels

   (c) 2nd-order in logs (as in modelc.mod)

   (d) 2nd-order in levels

Exercise. If things are going quickly you may want to redo the exercise if you substitute out consumption from the Dynare program and calculate consumption from the exact budget constraint in your simulation program.

1. Does doing this affect the policy function for capital?
2. Does it affect your answers for the mean of capital?
3. Does it affect the means for consumption?

3 Accuracy

The goal of this exercise is to check the accuracy of the neoclassical growth model programmed in modelc.mod. But if you already have some Dynare experience, I encourage you to assess the accuracy of your own favourite model.

Start with an easy case and then make it more challenging. Start with a case for which it should not be too difficult to get an accurate answer. In particular, start with a low value for sig (say 0.007) and a reasonable value for risk aversion (say 1). After that you can "stretch" the model and see how far you can get by increasing these two parameters

3.1 Euler-equation errors

To keep life simple you may just want to start with the case in which the innovations can take on only two values, namely 1 and -1, both with probability 1/2. But try to push yourself and also do the advanced test using Gaussian quadrature.

Grid. To calculate Euler-equation errors you have to construct a fine grid. To save time your grid doesn't have to be very fine but you should pick sensible boundaries. To determine sensible you should simulate your series. If your shocks can take on only two values you can still use the randn command. Simply set the shock equal to 1 if the value

is positive and equal to 1 if the value is non-positive. A long simulation gives you sensible bounds

What to report? Report the maximum and the average. Where do you find the largest errors? Did the simulated values ever get close to these combinations of the state variables?

3.2 DHM statistic

Simulate your data and calculate the DHM statistic. For simplicity you can simply use $h(z_t) = 1$. Get a new draw and repeat (up to 100 times). Do you roughly get the right number of rejctions? Do you reject more often if you increase the sample size used?

3.3 Dynamic Euler-equation errors

After all this you should be ready to calculate dynamic Euler equation errors. This boils down to calculate two series for each variable (here capital and consumption). One series is simply the one that comes from your policy function. The second is the one implied by the conditional expectation.

4 Matching model

The objective of this model is twofold. The first is to investigate the possibilities of non-linearities in more detail. Second to check for accuracy in a more serious model.

4.1 Non-linearities

Use the program dynv1simple.mod to get some idea about the relevant range for the state variables capital and employment. A relevant range for productivity can be calculated analytically. You can use this range for the exercises below.

Versions Rewrite the program so it is linear in levels instead of linear in loglevels. Now we have four versions (linear vs log-linear and first-order vs second-order).

1. Plot the different policy functions and see whether there are differences between the alternative programs. Given that there are three state variables you have to be efficient in looking at them. If you plot them as a function of capital you can keep the value of productivity fixed and use three different values for employment (say an average value, a value that is high according to the simulation, and a value that is low according to the simulation).

2. Calculate IRFs yourself. Use the simulation to choose several interesting and plausible initial conditions. Plot IRFs of output, employment, and vacancies and investigate whether they depend on initial conditions, on the magnitude of the shock, and whether the sign of the shock matters. A good way to plot IRFs is often to scale them with the IRF of produtivity (the exogenous driving process). If the model has no magnification then this relative IRF would not respond. If the size of the shock does not matter then this relative IRF would not be affected by the size of the shock (whereas the unscaled IRF of course is virtually always affected by the size of the shock).

3. Investigate second-order properties and in particular the standard deviation of employment, vacancies, and output relative to the standard deviation of productivity. Do they differ for the different versions of the program? Do they differ for different values of the standard deviation of the innovations? If you are ambitious you can try to calculate conditional standard deviations and see whether they vary over the cycle. That is, does risk increase in a downturn?

4.2 Accuracy

Repeat the accuracy exercises described above for this model.