Incorporating Dynare into Matlab programs; Comparing solutions for different structural parameter values and different types of solutions in one Matlab program

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

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

The coefficients are now stored in the matrix decision. If you want to use this option you have to replace the file disp_dr.m with the file provided.

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 random shocks
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.

Looping
Suppose you want to do the exercise above twice for different values of alpha. Then you can do the following.

1. Replace the command "alpha = 0.36" with

load wouterrocks;
set_param_value('alpha',alpha);

2. In your Matlab program loop over the different values of alpha. Save the current value of alpha to the file wouterrocks with "save wouterrocks alpha" before you 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. Save the results of each iteration (different value for alpha) and compare the different generated series

Part A: 1st & 2nd order – linear versus loglinear

Use the neoclassical growth model as programmed in the modelblevel.mod file. Set sig equal to 0.1. This is quite high but this way you get some action. The objective is to compare the following four solutions:
(a) log-linear
(b) linear in levels (as in modelblevel.mod)
(c) 2nd-order in logs
(d) 2nd-order in levels (as in modelblevel.mod)

First a *.mod file for the log-linear solution. Then write one matlab program that runs Dynare four times (for the four *.mod) files, each time generate a time series for capital and consumption, and finally make a plot for capital and consumption which contains the series according to the four different types of numerical solution.

Part B: IRFs and consumption smoothing

Write a Matlab program that loops over different values of risk aversion, runs Dynare, and then calculate the IRFs.