LSE Macroeconomics Summer Program

Part I: The Essentials

Instructor: Wouter J. Den Haan

Wednesday Assignment

New simulations parameterized expectations

1 Goal

The goal of this assignment is threefold. First, we show that simulations PEA is an algorithm that is quite easy to program. Second, we show the advantages of some of the proposals made by Maliar, Maliar, and Judd in their 2011 paper "One-node quadrature beats Monte Carlo: A generlized stochastic simulation algorithm". In particular, we show how results can be improved by using the (numerically approximated) conditional expectation, $\mathrm{E}_t[y_{t+1}]$ instead of the realization value, y_{t+1}, as the depending variable. The traditional procedure to do simulations PEA will be referred to as "non-quadrature simulations PEA" and the new version as "quadrature simulations PEA". Third, we show that calculating asset prices accurately is not as easy as calculating the real variables in typical business cycle models.

2 Model

We use the standard neoclassical growth model modfied so that agents can also invest in zero-coupon bonds. A zero-coupon bond with maturity j can be bought for $q_{j,t}$ and pays off 1 unit in period $t+j$.

Let $q_{0,t} = 1$. The first-order conditions of the agent can then be written as follows:

$$\begin{aligned}
c_t^{-\nu} &= \mathrm{E}_t\left[\beta c_{t+1}^{-\nu}\left(\alpha z_{t+1} k_t^{\alpha-1} + 1 - \delta\right)\right] \\
q_{j,t} c_t^{-\nu} &= \mathrm{E}_t\left[\beta q_{j-1,t+1} c_{t+1}^{-\nu}\right] \\
c_t + k_t + \sum_{j=1}^{J} q_{j,t} b_{j,t} &= z_t k_{t-1}^{\alpha} + (1-\delta) k_{t-1} + \sum_{j=0}^{J-1} q_{j,t} b_{j,t-1} \\
\ln z_t &= \rho \ln z_{t-1} + \varepsilon_t, \quad \varepsilon_t \sim N\left(0, \sigma^2\right)
\end{aligned}$$

Bonds are in zero net supply. This has the following two consequences for the equilibrium.

1. Most importantly, asset prices have no consequences for the real side of the economy. That is, one can first solve for the policy functions of c_t and k_t and separately for asset prices. If it is easy to obtain an accurate solution for the real side, then one can use a simple/cheap algorithm such as low-order perturbation for the real side and if necessary a more accurate algorithm for asset prices.
2. Here the real side of the economy is simply the standard neoclassical growth model.

3 Solving the real economy

3.1 Idea behind the algorithm

The idea of the algorithm can be summarized as follows.

- Start with a guess for $\mathrm{E}_t\left[\beta c_{t+1}^{-\nu}\left(\alpha z_{t+1} k_t^{\alpha-1} + 1 - \delta\right)\right]$. We use a polynomial in k_{t-1} and z_t with coefficients ψ. The program is written so you can use tensor-product polynomials of arbitrary order. Since most of the assignment is to focus on the stochastic outcome of the algorithm we focus on first order for simplicity. But feel free to go to higher-order approximations *after* you have everything running for the first-order version. In particular, we start with

$$\exp\left(\psi_0^0 + \psi_z^0 \ln z_t + \psi_k^0 \ln k_{t-1} + \psi_{zk}^0 \ln z_t \ln k_{t-1}\right)$$

as the approximation for the conditional expectation. The cross product shows up because we use tensor product polynomials.

- The ψ^j coefficients are updated using the following iterative scheme.

  1. Given a value ψ^j generate a time series $\{c_t, k_t, z_t\}_{t=1}^{T}$. Since z_t does not depend on ψ^j it can be generated outside the loop.
  2. Use observations from T_1 to T to find $\widehat{\psi}$, i.e., the value of ψ that is best for this simulation.
     (a) In the non-quad simulations version, you would focus on the following non-linear regression

     $$\beta c_{t+1}^{-\nu}\left(\alpha z_{t+1} k_t^{\alpha-1} + 1 - \delta\right) = \exp\left(\psi_0 + \psi_z \ln z_t + \psi_k \ln k_{t-1} + \psi_{zk} \ln z_t \ln k_{t-1}\right) + u_{t+1}$$

     (b) In the quad-simulations version, you would focus on the following non-linear regression

     $$\mathrm{E}_t\left[\beta c_{t+1}^{-\nu}\left(\alpha z_{t+1} k_t^{\alpha-1} + 1 - \delta\right)\right] = \exp\left(\psi_0 + \psi_z \ln z_t + \psi_k \ln k_{t-1} + \psi_{zk} \ln z_t \ln k_{t-1}\right)$$

     where the conditional expectation is calculated numerically (just as in yesterday's assignment). Since there is no error term in this equation, we are allowed to take the log on both sides which gives the following *linear* regression equation

     $$\ln \mathrm{E}_t\left[\beta c_{t+1}^{-\nu}\left(\alpha z_{t+1} k_t^{\alpha-1} + 1 - \delta\right)\right] = \psi_0 + \psi_z \ln z_t + \psi_k \ln k_{t-1} + \psi_{zk} \ln z_t \ln k_{t-1}$$

  3. Update ψ using

     $$\psi^{j+1} = \omega\widehat{\psi} + (1-\omega)\,\psi^j \text{ with } 0 < \omega \leq 1$$

3.2 Scaled versus non-scaled

In the last program of this assignment we will normalize (scale) the state variables. To make the programs similar the state variables have scaled versions here too (`ks_S` for `ks` and `zs_S` for `zs`), but here those are just the logs of the levels.

3.3 Details of the first program

The first mother program used is `"newpeaproi0.m"`. The places in the program where you have to complete something are indicated with XXX. The program has the following structure.

1. Set the parameter values of the model and the algorithm.
2. Simulate a time series for z_t.
3. Loop over the two options and read initial values. Regarding initial values the program allows you to either use a fixed set that is given (option=`'fresh'`) or—if you have run the program already and found a solution—to use the last obtained solution (option=`'previous'`). The first time you run the program, you obviously have to use `'fresh'`.
4. Perform the three steps described above for both versions.
5. Compare the two solutions. In section 5 of the program, the two solutions are plotted. It also plots an accurate solution (obtained with higher-order approximation, with a longer sample, and with more quadrature nodes). This sollution is stored in accurate.mat and read by the program.

The program uses the following external functions:

- `makepoly.m`: generates the polynomial terms given the specified order and the values for k_{-1} and z . If k_{-1} and z are vectors then you get a matrix with the polynomials terms of each observation in the corresponding row.

- `RHSrealization.m`: calculates $\beta c_{t+1}^{-\nu}\left(\alpha z_{t+1} k_t^{\alpha-1}+1-\delta\right)$ as a function of next period's shock ε_{t+1} for a given value of ψ.
- `numi.m`: calculates the expectation using Hermite Gaussian quadrature using as input a function (here RHSrealization) and a quadrature order.
- `rssfcn.m`: calculates the RSS, that is, for a given value for ψ, Y, and X it calculates

$$(Y-\exp(X\psi))'\left(Y-\exp(X\psi)\right)$$

3.4 What you have to do in newpeapro0.m

1. Give the expression for Y in the part of the program using non-quadrature.
2. Complete RHSrealization.m in the part of the program using quadrature simulation PEA.
3. Run the program. Be patient! It will need a couple hundred iterations to converge.
4. Look at the graphs to compare the two solutions. Are the differences in the generated series important?

4 Solve for asset prices

4.1 Idea behind the algorithm

Asset prices are given by

$$q_{j,t}=\mathrm{E}_t\left[\beta\left(\frac{c_{t+1}}{c_t}\right)^{-\nu}\right]$$

for $j=1$ and by

$$q_{j,t}=\mathrm{E}_t\left[\beta q_{j-1,t+1}\left(\frac{c_{t+1}}{c_t}\right)^{-\nu}\right]$$

for $j > 1$.

This means that one can get a solution for each of the bond prices in one step. It is not possible to solve for all J bond prices simultaneously, but given a solution for $q_{j-1,t}$ one can find the solution for q_j without iterating. It is important you understand the significance. Given that you get the solution in one step, it is easy to use a more expensive algorithm. For example, you can use a longer simulation, calculate $\mathrm{E}_t[\cdot]$ more accurately, and/or use a higher-order approximation.

This is good news because it is more difficult to solve accurately for asset prices than for real variables. The reason is the following. Focus on the expression for $q_{1,t}$. There are no exogenous variables directly on the right-hand side. Time variation in $q_{1,t}$ is completely driven by expected changes in an endogenous variable. With a direct influence of an exogenous variable it is easier to get at least an important part of the function of interest right. Getting the approximation for bond prices right using a simulation is difficult because consumption smoothing makes $(c_{t+1}/c_t)^{-\gamma}$ a persistent variable. In fact, it can display close to unit-root type behavior. And when variables are very persistent, then you need a long sample to get accurate regression results.

4.2 Particular exercise

A drawback of simulations PEA is that the the approximation obtained is stochastic. That is, the solution for ψ depends on the particular realization. This is less so for the version that calculates the conditional expectation explicitly, but even this version runs the regression at a random set of points. If T is large enough, then the sampling uncertainty should disappear but in practice we need to work with a finite T.

So the idea of the assignment is to generate B_iter samples and to calculate solutions for the bond prices in each of the B_iter samples. Given the solutions we investigate how different the B_iter solutions are. You will see that the differences are smaller when you calculate the conditional expectation numerically. In this part of the program we set T equal to 2,000, but feel free to experiment with other values.

4.3 Details of the first program

The file `newpeaproi0B.m` contains the part related to asset prices. This program assumes knowledge of everything calculated in `newpeaproi0.m` so don't clear anything out of memory or redefine variables. Alternatively, you may append `newpeaproi0B.m at the end of newpeaproi0.m` and use the `'previous'` option.

The structure of the program is as follows.

1. The main part loops over the B_iter different samples.

2. Simulate time series for c_t. Note that in all samples we use the same policy function to solve for c_t. That is, we do not recalculate the policy functions for the real side of the economy.

3. In each replication, we sequentially calculate the solutions for bond prices. Since we use a different set of shocks in each replication, we get a "somewhat" different outcome in each sample due to sampling uncertainty. The question is how much different and whether it matters which version of PEA we use.

4. Next, we calculate time series for yields and do some comparisons across the different samples.

4.4 What you have to do in newpeapro0B.m

1. Give the expression for Y in the part of the program using non-quadrature.

2. Complete RHSrealization_bond in the part of the program using quadrature simulation PEA.

3. Look at the graphs and absorb the results. Which of the two simulations PEA versions give more similar answers across different samples? Why do the results for higher maturities display more variation across samples for quadrature simulations PEA?

4.5 Output generated

The program calculates three graphs, one for each of the summary statistics considered (mean, standard deviation, and autocorrelation). The x-axis gives the number of the replication. Ideally you get flat lines, that is, sampling uncertainty does not matter and all replications give the same outcome. For each replication you get the outcome for the different maturities. The blue lines correspond to the non-quadrature version and the black lines to the quadrature version. Which display the least variation across replications?

5 Scaling and more convenenient formulation

The file `newpeaproi1.m` is an alternative mother program that also solves the real side of the economy. There are five differences with the approach above.

1. The Euler equation is written as

$$k_t = \mathrm{E}_t\left[k_t\beta\left(\frac{c_{t+1}}{c_t}\right)^{-\nu}\left(\alpha z_{t+1}k_t^{\alpha-1}+1-\delta\right)\right]$$

   Thus, the approximation to $\mathrm{E}_t\left[\cdot\right]$ directly gives the policy rule for one of the variables.

2. We use a regular polynomial instead of the exponential of a (regular) polynomial.

3. We normalize the variables. As mentioned in the slides, it is better to keep the scalings coefficients fixed (unless the model is so well behaved so you can set $\omega = 1$ in which case it doesn't matter). This means that the scaling for capital is approximate. (When you have solved the model you could resolve it with a better scaling). Anyway, the formula for the scaling is

$$\texttt{ks\_S} = \frac{\texttt{ks-ks\_mean}}{\texttt{ks\_std}}$$

and the same for productivity. For productivity we can choose fixed values for ks_mean and ks_std give the exact desired normalization.

4. We use a second-order approximation.

5. Note that the value for ω (`lrate`) is set to a much lower value. In general, I found this version of the algorithm to be less stable (but with the lower value of ω it does converge nicely).

5.1 What you have to do in newpeapro1.m

1. Use the `'fresh'` option and think of a way to come up with good initial conditions for ψ, `ks_mean`, and `ks_std`.. Think of what you have already done today!

2. Give the expression for Y in the part of the program using non-quadrature.

3. Complete RHSrealization_K in the part of the program using quadrature simulation PEA.

6 Additional exercises

Here we didn't check directly for accuracy, but simply looked at sensitivity. Having similar results across different samples is necessary. but not sufficient for accuracy. Check how the results change if you use a higher value for T (when you have more time to let the programs run). Which of the two simulations PEA versions changes more if you increase T? Also try higher-order approximations.