Solution Methods for Macroeconomic Models

Assignment 2

Petr Sedláček

1 Objective

In this assignment you are asked to solve a DSGE model using value function iteration. We'll focus on the "Brock-Mirman" version of the solution, which will allow us to compare our solution to the true non-linear policy rules.

2 Model

The model is the standard "neoclassical growth model" discussed in class. In particular, we assume

- a representative household maximizing expected life-time utility (derived from consumption only): $\max_{\{c_t,k_{t+1}\}_{t=0}^{\infty}} \mathbb{E}_0 \sum_t \beta^t \left(\frac{c_t^{1-\gamma}}{1-\gamma}\right)$
  - where γ is the coefficient of relative risk aversion
- the household owns the production technology which uses capital (think of labor as inelastically supplied and normalized to 1) as an input factor and is subject to productivity shocks: $y_t = z_t k_t^{\alpha}$
  - where $\alpha \in (0,1)$ and $\ln z_t = \rho \ln z_{t-1} + \epsilon_t$ is aggregate productivity, where $\epsilon_t \sim N(0,\sigma^2)$ and $\rho \in (0,1)$ is a persistence parameter and z_0 is given
- resources are spent on consumption and investment into accumulation of capital and, each period, a fraction (δ) of the capital stock depreciates: $c_t + k_{t+1} = y_t + (1-\delta)k_t$, with k_0 given

3 Assignment

3.1 Bellman equation

Given that we're focusing on value function iteration, we want to rewrite our model using the Belmman equation. You have this on the slides, just make sure you understand it.

After you've done so, have a look through `VFIMain.m` to make sure you understand the structure of the code. `VFIMain.m` first sets the model parameters, creates a grid for the two state variables and then proceeds to solve the model using value function iteration.

3.2 Setting up the algorithm

In the first part of `VFIMain.m` you should do the following:

- set the model parameters to the following values: $\alpha = 1/3$, $\beta = 1.03^{-1/4}$, $\gamma = 1$, $\delta = 1$, $\rho = 0.9$ and $\sigma = 0.01$. The parameter $\lambda \in (0, 1]$ is the speed at which the value function iteration algorithm updates guesses. Play around with this to see how the solution behaves.
- given that this is a relatively simple model, you can write the steady state analytically. Therefore, fill in the steady state values for capital (use the Euler equation), consumption (use the budget constraint) and productivity (what is it?).
- once you fill in the steady state, we need to create the grid for the state variables. Make an educated guess of what a good range for the capital grid could be and specify a lower and an upper bound. The code then creates equidistant nodes between those bounds. For the productivity grid, we're using the Tauchen method to create a grid for us.

3.3 Value function iteration

Now that we're all set to go, let's fill in the value function iteration algorithm. We'll be doing things one grid at a time. For each grid point (i and j combination) do the following:

- find the optimal capital choice, given the Bellman equation (and budget constraint). Tip: use the "max operator" (`Vnew(i,j)` is the "new" maxed value function and `inds` is the index of your optimal capital choice)
- use the above to define the optimal capital choice and associated consumption choice as `kpol(i,j)` and `cpol(i,j)`, respectively.

Outside of the VFI loop, define a sensible stopping criterion (`crit`) and update your guess of the value function accordingly (don't forget to do it slowly, using λ). The code will show you how the algorithm is (or isn't?) converging.

The final part of the code plots your solution against the true policy rules. Are they close? What could you do to make them even closer?

3.4 Bonus

We've coded up the value function iteration algorithm going one grid point at a time. While intuitive, this is also inefficiently slow. Try to figure out a way how you could speed this up – vectorization can go a long way!