Exercise: Solving Representative Agent RBC Model with `Dynare`

The goal of this exercise is to introduce you to the basics of using `Dynare`. I have provided sample code to solve the RBC model using `Dynare`. This exercise asks you to install `Dynare`, run those codes, and tweak them in a few different ways.

1. **Install Dynare**: Instructions are available at `https://www.dynare.org/`

2. **Solve basic RBC model**:

   (a) Ensure that your current working directory includes `rbc_model.mod`.

   (b) Type `dynare rbc_model.mod` into the command window.

   (c) Read through the RBC model codes to familiarize yourself with basic `Dynare` syntax. Refer to the `Dynare` documentation as needed.

3. **Compute a higher-order perturbation**: you'll notice that the codes I provided perform a first-order perturbation (i.e., they linearize the model). You can have `Dynare` compute a higher-order perturbation by setting `order = 2` or `order = 3`.

4. **Change a parameter value**: Karabarbounis and Neiman (2013) argue that the labor share has fallen over the past thirty years. We can capture this fact in the model by changing the value of the labor share parameter $1 - \alpha$. Please set $\alpha = 0.5$. How do the dynamics of the model change?

5. **Add habit formation to the model**: this simple RBC model abstracts from many of the bells and whistles people incorporate in DSGE models. One example is habit formation over consumption. Please add habit formation to the model by changing the flow utility over consumption to be $\frac{(c_t - bc_{t-1})^{1-\sigma}}{1-\sigma}$. Set the parameter $b = 0.5$ as a baseline value. How do the dynamics of the model change? How about when $b = 0.8$?

Hopefully this exercise has convinced you that `Dynare` is a powerful tool to solve DSGE models by perturbation. Use it! There is no need to reinvent the wheel every time you solve a model (e.g. take the model's derivatives, put it into matrix form, solve the quadratic matrix equation, simulate the model yourself, compute higher-order dertivatives, etc.). Doing so will likely create mistakes and waste time coding something that a team of experts has already programmed. Instead, you should use your time solving new problems.