Economics 701

Advanced Macroeconomics I
Project 0

Professor Sanjay Chugh

Fall 2011

Objective

To get you started using Matlab (the recommended language – although feel free to use whatever language you wish, such as Fortran, Gauss, etc.), and because it will be a building block needed for constructing approximations to (locally-accurate) decision rules of models we will encounter, you will computationally solve for the deterministic (aka non-stochastic) steady state of a few variants of the basic RBC economy. **A good, complete submission (excluding code) should not need to exceed a maximum of four pages.**

The Problem(s)

Numerically compute the deterministic steady-state values of *c, n,* and *k* (i.e., consumption, labor, and the capital stock) for both the **Social Planning problem** of the basic RBC economy and the associated **decentralized economy** with proportional labor-income and capital-income taxation**.** For the decentralized economy, also solve for the steady-state real wage and real rental rate of capital.

Recall that the solution to the Social Planning problem is described by

1. Consumption-leisure efficiency condition
2. Consumption-investment efficiency condition
3. Aggregate resource constraint

and the solution to the decentralized economy is described by

1. (Household) consumption-leisure optimality condition
2. (Household) consumption-savings optimality condition
3. (Firm) labor demand function
4. (Firm) investment demand function
5. Aggregate resource constraint

Throughout, assume that the household instantaneous utility function is

$$u(c_t,n_t)=\frac{c_t^{1-\sigma}-1}{1-\sigma}-\frac{\psi}{1+\nu}n_t^{1+\nu}$$

and the production technology is

$$f(k_t,n_t)=z_t k_t^{\alpha} n_t^{1-\alpha}.$$

Also, suppose throughout that the steady-state value of government purchases is zero ($\bar{g} = 0$) and the steady-state level of TFP is $\bar{z} = 1$. For the decentralized economy, assume that the steady-state labor tax rate is $\tau^n = 0.2$ and the steady-state capital income tax rate is $\tau^k = 0.4$.

For both the Social Planning problem and the decentralized equilibrium problem, compute the steady state for the three different sets of parameters shown in the table below.

	Parameter Set A (Baseline)	**Parameter Set B (High Depreciation)**	**Parameter Set C (High Risk Aversion)**
β	0.99	0.99	0.99
δ	0.02	0.10	0.02
σ	1.2	1.2	5
α	0.36	0.36	0.36
ν	0.7	0.7	0.7
ψ	6.6	6.6	6.6

Thus, you will compute six steady states in total:

1. **Steady State 1:** Social Planning solution with parameter set A
2. **Steady State 2:** Social Planning solution with parameter set B
3. **Steady State 3:** Social Planning solution with parameter set C
4. **Steady State 4:** Decentralized economy with parameter set A
5. **Steady State 5:** Decentralized economy with parameter set B
6. **Steady State 6:** Decentralized economy with parameter set C

In addition to computing these steady states, provide brief discussion/intuition of the **comparative statics** of the models. Specifically, discuss the economic reason(s) behind how/why the endogenous steady-state variables change as you move from parameter set A (the baseline economy) to parameter set B (the high-depreciation economy) to parameter set C (the high-risk-aversion economy) for both the Social Planning solution and the decentralized equilibrium.

Finally, compute the steady-state level of household lifetime utility for each of the six cases. For each parameter set, how does lifetime utility compare between the Social Planning solution and the decentralized equilibrium? Provide some (brief) economic discussion/interpretation of how/why they do or do not differ?

Feel free to provide any other discussion/interpretation/comments you find insightful for understanding your results.

Hint: For the **Social Planning** problem, what are the implicit values of the labor income tax rate and the capital income tax rate? Can you exploit this fact to streamline the amount of code you must write?

(Some) Computational/Programming Guidance

If using Matlab, you will have to acquaint yourself with Matlab's **fsolve** function. Matlab's **fsolve** function solves a system of (possibly non-linear) equations for a specified vector (or matrix) of variables and hence is suited for computing both the deterministic steady state of models we will encounter as well as for approximating their decision rules. Also perhaps useful (though I believe not necessary for this exercise) is Matlab's **if-else** structure, which allows you to control which lines of code get executed depending on an evaluation of conditions you (the programmer) specify. Matlab's built-in HELP system is useful for learning the basics of all built-in functionality. To access Matlab's HELP feature, you can type at the command line, for example, `help fsolve` or `help if`. More generally, invoking `help xxx` returns information about Matlab's functionality xxx.

Feel free to write your code/programs in whatever way you wish (and this will surely evolve as you experiment with different programming styles over the coming weeks and months (and perhaps years…), some that will suit you and some that will not), but there are three guidelines you should (must) follow:

1. Matlab program files are called "m-files." There are two broad types of m-files that you can/will need to create: script files and function files. Your code/programs should be written as one or more than one (obviously inter-related) m-files.

2. Your programs should be written flexibly enough that you can easily run your code for different values of model parameters (the coefficient of relative risk aversion, the share of capital in the production function, the depreciation rate of physical capital, and so on). That is, none of your model parameter values should be "hard-coded" in your programs – they should all be written as "variables." If using Matlab, this requires some block/section of code where the model parameters are "defined" or "declared."

3. Your programs should be at least somewhat readable by someone familiar with programming (i.e., me). Towards this end, you should at least sprinkle (and maybe even heavily douse) your programs with comments (look up what "comments" are if you are unfamiliar with the idea) that describe the tasks/computations that different sections of your code perform. More importantly than making it easier for a reader of your code to understand your logic and computational structure, useful commenting makes it easier for **you** to remember/understand what **you yourself** were originally thinking when you return to your code after initially writing it.

What To Submit

Your submission should essentially be a “scientist’s notebook” (as opposed to (mini-) papers in subsequent projects). Your submission **must be typed, not hand-written,** and include:

1. A clear, concise presentation of your numerical results (whether in tabular form, in text forn, some combination of both, or whatever format you find most instructive).
2. A (short) commentary/discussion/interpretation of your findings as outlined above (i.e., the economic intuition behind your findings, at least as requested above and any further thoughts/comments/interpretations welcome).
3. A print-out of your code (i.e., all the relevant files that one would need to re-construct your results).

A good, complete submission (excluding code) should not need to exceed a maximum of four pages.