Economics 8861.01
Project 0 – Suggested Solutions
Professor Sanjay Chugh
Fall 2015

Objective

To get you started using Matlab and because it will be a building block needed for constructing approximations to (locally-accurate) decision rules of models we will study, 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.

Specifically, in this short submission, a (short) Abstract should state the main question, the methodology used to answer the question, a terse statement of the main result(s), and brief economic interpretation/intuition. **(Note:** economic interpretation is **not** simply a verbal restatement of the mathematics or numerical results provided in tables. An example might help: when Fed Chairwoman Janet Yellen holds a press conference with millions of listeners around the world, pointing to a mathematical equation probably would not make for a good sound bite.)

Further details about "what to submit" appear at the end of this document.

The Problem

Numerically compute the deterministic steady-state values of *c, n,* and *k* (i.e., consumption, labor, and the capital stock) for the **Social Planning problem** of the basic RBC environment.

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

Throughout, assume that the household instantaneous utility function is

$$u(c_t, n_t) = \frac{\left[c_t(1-n_t)^{\psi}\right]^{1-\sigma} - 1}{1-\sigma}$$

and the stationary-state (aka, balanced-growth) production technology is

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

The steady-state value of government purchases is zero ($\overline{g} = 0$) and the steady-state level of TFP is $\overline{z} = 1$.

The goods resource constraint for the economy is

$$c_t + k_{t+1} - (1-\delta)k_t = z_t k_t^{\alpha} n_t^{1-\alpha}.$$

For the Social Planning problem, compute the deterministic steady state values for the two different sets of parameters shown in the table below. (Note: you are **not** solving for any dynamics, just long-run values for each set of parameterizations.)

	Baseline (Parameter Set A)	**High Risk Aversion (Parameter Set B)**
B	0.99	0.99
Δ	0.02	0.02
Σ	1	1.5
A	0.36	0.36
Ψ	**To be determined**	**To be determined**

For each parameter set, you should compute (calibrate) the value of ψ so that $n^{SS} = 0.30$.

Thus, you will compute two different deterministic steady states in total.

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 environment) to parameter set B (the high risk-aversion environment) **This brief discussion should compare and contrast interesting aspects of the different allocation(s), NOT simply provide a verbal restatement of the numerical solutions.**

Finally, compute the steady-state level of household lifetime utility for each of the two cases. 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.

Solution:

Parameter Set A
css = 0.9127, lss = 0.3, kss = 14.4898, psi = 1.9628, wss = 2.5846, rss = 0.0301
utilityss = -0.7822

Parameter Set B
css = 0.9127, lss = 0.3, kss = 14.4898, psi = 1.9628, wss = 2.5846, rss = 0.0301
utilityss = -0.9563

All long-run quantity values are identical across the two structural parameter sets. Because utility is just an ordinal, not a cardinal, value, "comparison" of the steady-state welfare across the two structural sets of parameters is meaningless.

(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 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 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 days or weeks or months after initially writing it.

What To Submit

Your submission **must be typed, not hand-written,** and include:

1. A BRIEF abstract of the results at the very beginning. Your abstract should include good, clear economic intuition, and (if it helps you explain something important) either one or at most two important numerical results. (NOTE: economic intuition is **not** simply a verbal restatement of the mathematics or the numerical values in tables, etc.)

2. A clear, concise definition of steady-state equilibrium.

3. A **clear, concise** presentation of your numerical results (whether in diagrams, tabular form, in text form, some combination of all three, or whatever format you find most instructive for your reader). But note that **you should NOT have to walk the reader through every aspect of the results presented in diagrams and/or tables.** Rather, what is needed is some useful and brief discussion of the ideas behind the results, with appropriate references made to some of the numerical results in your diagrams and/or tables.

4. If there is other **(short)** commentary/discussion/interpretation of your findings beyond that outlined above, present the ideas or discussion behind them briefly. That is, present any other numerical results you find useful to show, and present the **economic intuition** behind the results. (NOTE: economic intuition is **not** simply a verbal restatement of the mathematics or the numerical values in tables, etc.)

5. A print-out of your code (i.e., all the relevant files that one would need to replicate your results).

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