Macro II (M.E.A., UC3M)
Professor: Matthias Kredler
Problem Set 1
Due: 31 January 2020 (except for Problem 4, which is due on 7 Feb.)
You are encouraged to work in groups; however, every student has to hand in his/her own version of the solution.

1. **Lagrangian approach for neo-classical growth model with labor.** Consider an economy with aggregate production function $F(k_t, n_t)$, which is constant-returns-to-scale, continuously differentiable and concave in (k, n) and where $F_k(k, n) > 0$ and $F_n(k, n) > 0$ for all (k, n). We also impose the Inada conditions

$$F(0,n) = 0 \qquad \lim_{k\to 0} F_k(k,1) = \infty \qquad \lim_{k\to\infty} F_k(k,1) = \infty$$
$$F(k,0) = 0 \qquad \lim_{n\to 0} F_n(1,n) = \infty \qquad \lim_{n\to\infty} F_n(1,n) = 0.$$

The representative agent has an endowment of one unit of time per period, which may be divided between leisure l_t and work n_t. The agent orders sequences of consumption and leisure by

$$\sum_{t=0}^{\infty} \beta^t u(c_t, l_t)$$

where $0 < \beta < 1$, $u_c(c, l) > 0$ and $u_l(c, l) > 0$ for all (c, l). We impose the Inada conditions

$$\lim_{c\to 0} u_c(c,l) = \infty \qquad \lim_{l\to 0} u_l(c,l) = \infty.$$

Investment is standard:

$$k_{t+1} + c_t \leq f(k_t, n_t) \equiv F(k_t, n_t) + (1-\delta)k_t$$

   (a) Write down the Lagrangian for this problem – be careful to argue exactly which constraints will bind and which will not. Find the first-order conditions, write them in terms of the allocation $\{k_{t+1}, n_t, c_t\}$ and interpret them. Which of these conditions are *intratemporal* (i.e. they only involve variables corresponding to a single period t) and which are *intertemporal* (i.e. they involve variables corresponding to more than one period)?

   (b) From these first-order conditions, find a minimal set of equations that characterize a steady state where $k_t = \bar{k}$, $c_t = \bar{c}$ and $l_t = \bar{l}$ for all t.

2. **Euler approach: Choose trajectory $\{a_t\}$.** Consider a finite-horizon consumption-savings problem with time $t = 0, 1, \ldots, T$. The budget constraint at t is

$$a_{t+1} \leq R(w_t + a_t - c_t),$$

where $\{w_t\}_{t=0}^{T}$ is an exogenously-given sequence of earnings, a_t are assets coming into period t, and c_t is consumption at t. Borrowing is not constrained for $t = 0, 1, \ldots, T-1$, but the consumer cannot die with debt, i.e. we require $a_{T+1} \geq 0$.[1] The consumer maximizes the criterion

$$\sum_{t=0}^{T} \beta^t u(c_t),$$

[1]Formally, define the sequence of borrowing limits as $\bar{a}_1 = \bar{a}_2 = \cdots = \bar{a}_T = -\infty$ and $\bar{a}_{T+1} = 0$.

where where $0 < \beta < 1$, $u' > 0$, $u'' < 0$ and $\lim_{c \to 0} u'(c) = \infty$. Derive the Euler equations in a different way now:

(a) Write c_t as a function of a_t and a_{t+1} (argue that the budget constraint has to hold with equality) and write the agent's objective as a function of $(a_1, \dots, a_T)$ – recall that a_0 is given and fix the optimal choice $a_{T+1} = 0$.

(b) Take the first-order conditions for a_t, show that they are equivalent to the Euler equations that we derived in class and interpret them.

3. **Equivalence of dynamic-programming and sequence approach in finite horizon.** Let X be some non-empty set and let $\{\Gamma_t(\cdot)\}_{t=0}^{T}$ be non-empty[2] correspondences that map X to X. Also, let $\beta > 0$ and $\{F_t(\cdot,\cdot)\}_{t=0}^{T}$ be functions that map $X \times X$ to $\mathbb{R}$. Consider the sequence problem

$$\begin{aligned} &\sup_{\{x_{t+1}\}_{t=0}^{T}} \sum_{t=0}^{T} \beta^t F_t(x_t, x_{t+1}) \\ &\text{s.t. } x_{t+1} \in \Gamma_t(x_t) \qquad\qquad \text{for } t \in \{0, \dots, T\} \\ &\qquad x_0 \text{ given.} \end{aligned}$$

The dynamic-programming problem pertaining to this sequence problem is given by the following recursion on the value functions $\{V_t(\cdot)\}_{t=0}^{T+1}$:

$$\begin{aligned} V_{T+1}(x) &= 0, \\ V_t(x) &= \sup_{x' \in \Gamma(x)} \Big\{ F_t(x, x') + \beta V_{t+1}(x') \Big\} \qquad \text{for } t \in \{0, 1, \dots, T\}. \end{aligned}$$

Prove that the set of policies prescribed by the dynamic-programming approach is identical to the set of policies that attain the supremum in the original sequence problem.

4. **Matlab computation of finite-horizon growth model. NOTE: For this problem, please form groups of 2-3 students who hand in one solution. This problem is only due in two weeks' time.** Consider the neo-classical growth model in finite horizon: The production function is k^α and capital depreciates at the rate δ, so the law of motion for the capital stock is

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

Use the parameterization $\alpha = 0.3$ and $\delta = 0.1$. The representative agent orders consumption streams by

$$\sum_{t=1}^{T} \beta^t \ln c_t,$$

where $\beta = 0.96.$ and $T = 50$. In order to know what are reasonable quantities of capital in this economy, we will take the steady-state capital stock of the corresponding infinite-horizon economy

$$k^* = \left(\frac{\alpha}{\frac{1}{\beta} - (1-\delta)} \right)^{\frac{1}{1-\alpha}}$$

as a reference point.

[2]i.e., $\Gamma_t(x) \neq \emptyset$ for all $x \in X$, for all t.

(a) You should always start your Matlab programs defining the parameters:
```
alpha = 0.3;
beta = 0.96;
delta = 0.1;
```
(Tip: The semicolon makes Matlab supress the output when you run the program.) Now, define a column vector of size $N = 100$ which is linearly spaced between $2k^*/N$ and $2k^*$. These will be the different capital stocks that are possible in this economy. Tip: Use the function `linspace` here or use the syntax `x = 0:0.1:1`.

(b) Calculate a column vector f which contains $f(k)$, the amount of available resources, for each grid point k_i. Tip: `x.^a` performs exponentation to the a-th power of each element of x in Matlab – `x^a` a is interpreted as the exponentation of matrix, e.g. $A^2 = A \times A$.

(c) Obtain a matrix c which indicates the consumption levels c_{ij} that are implied if the economy moves from grid point k_i to grid point k_j; $c_{ij} = f(k_i) - k_j$ should be stored in the ith row and the jth column. Tip: Use matrix multiplication (e.g. `x * ones(1,N)`) to obtain a matrix where a row has the same number everywhere.

(d) Now, obtain a matrix `u` with the utility corresponding to each consumption level in `c`. Use the following code to get rid of negative consumption levels:
```
cneg = (c <= 0);
u = log(c);
u(cneg) = -Inf;
```
Comments: `x = (x>0)` returns a *logical vector*; try it. Logical vectors can be used for indexing, which is done in the last line. Matlab knows how to calculate with `Inf` and `-Inf`, which is very practical at times.

(e) Now we have everything in place to begin the loop with the iteration on the value function. First, set up empty matrices to store the value function and the optimal policies:
```
T = 50;
V = zeros(N,T);
P = zeros(N,T);
```
We can already fill the value function for the last period since we know that it is optimal to eat everything. Create a column vector v that contains this number for each point on the k-grid. Tip: Use the syntax
```
V(:,T) = v;
```
to write `v` into `T`th column of the matrix `V`. We will now always use `v` to store the current value function and `vp` to store the value for the period lying ahead.

(f) Now we can finally start the loop from period $T-1$ to 1. Look up the documentation for the function `for`-loop in Matlab to do this. In each step, obtain the new `v`-vector. Tip: Use the following syntax to pick off the maximal element `v(i)` and the index of the maximizing element `p(i)` of each row in the appropriate matrix `A`:
```
[v, p] = max(A,[],2);
```
(The third argument of the `max`-function refers to the dimension of the matrix from which the maximum has to be picked. We have to give Matlab an empty matrix `[]` as an argument, if not it would compare the matrix `A` to the number 2 for each element.) Then, store the value function and the policies in `V` and `P`.

(g) Use the function `surf` to plot a 3-dimensional surface plot of the value function.

(h) Report what happens to the value function and to optimal policies when you change the parameters β and δ.