Problem Set #2

MACS 40000, Dr. Evans
Due Monday, Oct. 9 at 1:30pm

1. **Checking feasibility of steady-state initial guess (2 points).** Using the calibration of the 3-period-lived agent model described in Section 5.6 of Chapter 5, write a Python function named `feasible()` that has the following form,

```
b_cnstr, c_cnstr, K_cnstr = feasible(f_params, bvec_guess)
```

   where the inputs are a tuple `f_params = (nvec, A, alpha, delta)`, and a guess for the steady-state savings vector `bvec_guess = np.array([scalar, scalar])`. The outputs should be Boolean (`True` or `False`, `1` or `0`) vectors of lengths 2, 3, and 1, respectively. `K_cnstr` should be a singleton Boolean that equals `True` if $K \leq 0$ for the given `f_params` and `bvec_guess`. The object `c_cnstr` should be a length-3 Boolean vector in which the sth element equals `True` if $c_s \leq 0$ given `f_params` and `bvec_guess`. And `b_cnstr` is a length-2 Boolean vector that denotes which element of `bvec_guess` is likely responsible for any of the consumption nonnegativity constraint violations identified in `c_cnstr`. If the first element of `c_cnstr` is `True`, then the first element of `b_cnstr` is `True`. If the second element of `c_cnstr` is `True`, then both elements of `b_cnstr` are `True`. And if the last element of `c_cnstr` is `True`, then the last element of `b_cnstr` is `True`.

   (a) Which, if any, of the constraints is violated if you choose an initial guess for steady-state savings of `bvec_guess = np.array([1.0, 1.2])`?

   (b) Which, if any, of the constraints is violated if you choose an initial guess for steady-state savings of `bvec_guess = np.array([0.06, -0.001])`?

   (c) Which, if any, of the constraints is violated if you choose an initial guess for steady-state savings of `bvec_guess = np.array([0.1, 0.1])`?

2. **Solve for the steady-state equilibrium (4 points).** Use the calibration from Section 5.6 and the steady-state equilibrium Definition 5.1. Write a function named `get_SS()` that has the following form,

```
ss_output = get_SS(params, bvec_guess, SS_graphs)
```

   where the inputs are a tuple of the parameters for the model `params = ((beta, sigma, nvec, L, A, alpha, delta, SS_tol))`, an initial guess of the steady-state savings `bvec_guess`, and a Boolean `SS_graphs` that generates a figure of the steady-state distribution of consumption and savings if it is set to `True`.

   The output object `ss_output` is a Python dictionary with the steady-state solution values for the following endogenous objects.

```
ss_output = {
  'b_ss': b_ss, 'c_ss': c_ss, 'w_ss': w_ss, 'r_ss': r_ss,
  'K_ss': K_ss, 'Y_ss': Y_ss, 'C_ss': C_ss,
  'EulErr_ss': EulErr_ss, 'RCerr_ss': RCerr_ss,
  'ss_time': ss_time}
```

Let `ss_time` be the number of seconds it takes to run your steady-state program. You can time your program by importing the time library

```
import time
...
start_time = time.clock() # Place at beginning of get_SS()
...
ss_time = time.clock() - start_time # Place at end of get_SS()
```

And let the object `EulErr_ss` be a length-2 vector of the two Euler errors from the resulting steady-state solution given in difference form $\beta(1+\bar{r})u'(\bar{c}_{s+1}) - u'(\bar{c}_s)$. The object `RCerr_ss` is a resource constraint error which should be close to zero. It is given by $\bar{Y} - \bar{C} - \delta\bar{K}$.

(a) Solve numerically for the steady-state equilibrium values of $\{\bar{c}_s\}_{s=1}^3$, $\{\bar{b}_s\}_{s=2}^3$, $\bar{w}$, $\bar{r}$, $\bar{K}$, $\bar{Y}$, $\bar{C}$, the two Euler errors and the resource constraint error. List those values. Time your function. How long did it take to compute the steady-state?

(b) Generate a figure that shows the steady-state distribution of consumption and savings by age $\{\bar{c}_s\}_{s=1}^3$ and $\{\bar{b}_s\}_{s=2}^3$.

(c) What happens to each of these steady-state values if all households become more patient $\beta \uparrow$ (an example would be $\beta = 0.55$)? That is, in what direction does $\beta \uparrow$ move each steady-state value $\{\bar{c}_s\}_{s=1}^3$, $\{\bar{b}_s\}_{s=2}^3$, $\bar{w}$, and $\bar{r}$? What is the intuition?

3. **Solve for the non-steady-state equilibrium time path (4 points).** Use time path iteration (TPI) to solve for the non-steady state equilibrium transition path of the economy from $(b_{2,1}, b_{3,1}) = (0.8\bar{b}_2, 1.1\bar{b}_3)$ to the steady-state $(\bar{b}_2, \bar{b}_3)$. You'll have to choose a guess for T and a time path updating parameter $\xi \in (0,1)$, but I can assure you that $T < 50$. Use an L^2 norm for your distance measure (sum of squared percent deviations), and use a convergence parameter of $\varepsilon = 10^{-9}$. Use a linear initial guess for the time path of the aggregate capital stock from the initial state K_1^1 to the steady state K_T^1 at time T.

(a) Plot the equilibrium time paths of the aggregate capital stock $\{K_t\}_{t=1}^{T+5}$, wage $\{w_t\}_{t=1}^{T+5}$, and interest rate $\{r_t\}_{t=1}^{T+5}$.

(b) How many periods did it take for the economy to get within 0.00001 of the steady-state aggregate capital stock $\bar{K}$? What is the period after which the aggregate capital stock never is again farther than 0.00001 away from the steady-state?