Macro I

Homework 3- Finite Dynamic Programming

1. Consider the optimal growth problem.

$$V^*(k) = \max \sum_{t=0}^{\infty} \beta^t u(c_t) \ s.t. \ c_t + k_{t+1} \leq F(k_t), given \ k_0 = k > 0$$

This problem has a known solution when $u(c) = \ln(c), F(k) = Ak^\alpha, A > 0, \alpha \in (0,1), \beta \in (0,1)$. The solution is $V^*(k) = E + F\ln(k)$ and the stationary optimal decision rule for the choice of capital is $k' = \alpha\beta A k^\alpha$. The constants equal $E = \frac{1}{(1-\beta)}[\ln A(1-\alpha\beta) + \frac{\beta\alpha}{1-\alpha\beta}\ln A\beta\alpha]$ and $F = \frac{\alpha}{1-\alpha\beta}$.

(a) Write a program to solve the (closely related) finite dynamic programming problem indicated below. Include your program with your homework. A finite dynamic programming problem has a finite number of states and controls.

$$V(x) = \max_{y \in Y(x)} u(F(x) - y) + \beta V(y)$$

$$X = \{x_1, ..., x_N\}$$

$$Y(x) = \{y \in X : 0 \leq y \leq F(x)\}$$

(b) Compute a solution $V(x)$ when you set $A = 18.5$, $\alpha = .3$ and $\beta = 0.9$. Set the state space $X = \{x_1, ..., x_N\}$ so that $N = 100$ and $x_i = 20(i)/(N)$. Thus, the state space has 100 grid points on the interval $[0, 20]$.

(c) Graph the function $V(x)$ and $V^*(x)$ over grid points in X on the same graph.

(d) Graph the computed decision rule solving Bellman's equation and the optimal decision rule over gridpoints in X on the same graph.

(e) Graph the function $V(x)$ and $V^*(x)$ when $N = 200$ and the gridpoints x_i are evenly spaced according to the rule used above.

SUGGESTIONS FOR PROBLEM 1:

1. Create arrays $U(NX, NY), V(NX, NJ), Y(NX, NJ)$. Initialize $V(NX, NJ)$ to a vector of zeros.

[Note: (i) $NX = NY = 100$]

2. Compute the array $U(NX, NY)$ once, assigning sufficiently small negative values to choices of controls that are not feasible.

3. Iterate backwards on Bellman's equation NJ times, where NJ is a choice (say $NJ = 100$). The right-hand-side of Bellman's equation involves terms with $V(NX, j+1)$, whereas the left-hand-side solves for $V(NX, j)$. Let $V(NX, 1)$ that results from (backwards) interations on Bellman's equation be the computed candidate for $V(x)$ in Bellman's equation in Problem 1. Let $Y(NX, NJ)$ be the matrix of decisions solving the right-hand-side of Bellman's equation from the initial guess specified in suggestion 1 above.

3. The main program will consist of two DO LOOPS. The outer DO LOOP iterates (backwards) over time (NJ). The inner DO LOOP iterates over states (NX). Inside these two DO LOOPS there is a maximization operation. Perform the maximization operation on the right-hand-side of Bellman's equation with a built-in vector maximization operation. Note: Fortran DO LOOPS are Matlab FOR LOOPS.

4. Check whether or not your value function $V(NX, NJ)$ and decision rule $Y(NX, NJ)$ resulting from $NJ = 100$ iterations on Bellman's equation changes in an important way as the value of NJ is doubled to $NJ = 200$. This is a low tech means of assessing convergence.