Non stochastic VFI (finite time) in Matlab

Application to the NGM

Fabrizio Leone
fabrizioleone93@gmail.com

February 3, 2018

Introduction

The code solves the basic non-stochastic neoclassical growth model in Matlab. The problem looks like:

$$
\begin{aligned}
V(k) = \max_{k' \in \Gamma(k)} & \{u(c) + \beta V(k')\} \\
& c = f(k) + (1-\delta)k - k' \\
& k_0 > 0 \text{ given}
\end{aligned}
$$

The solution method consists in simple value function iteration. The existence and the uniqueness of the solution are ensured by the *contraction mapping theorem*.

The Code

```
% NGM model with value function iteration
% Code for finite time DP
% we set V_{T+1}=0 and solve backwards

% Fabrizio Leone - 03-02-2018

%% 1. Housekeeping
clear all
clc

%% 2. Set parameters
sigma=1;
beta=0.96;
alpha=0.33;
delta=0.04;
kstar=(((1/(alpha*beta))-((1-delta)/alpha)))^(1/(alpha
    -1));


%create a grid for capital
kmin=kstar*.9;
kmax=kstar*1.1;
T=150; %number of periods. so T-1 is our last choice
    period
step=(kmax-kmin)/(T-1);
k=(kmin:step:kmax);

%% 3. Set variables

y=k.^alpha; %production function
ytot=repmat(y+(1-delta)*k,T,1);
kprime=repmat(k,T,1);
c=ytot-kprime'; %check how consumption is constructed

%define utility function

if sigma==1
u=log(c);
```

```
else
u=(c.^(1-sigma)-1)/(1-sigma);
end
u(c<0) = -Inf;

% Initialize VFI
V0=zeros(T,T); %initial value function guess
value=zeros(T,T);
policy=zeros(T,T);

for t=1:T-1


    v=u+beta*V0;
    [V1 p]=max(v,[],1); %compute value and policy
        function (p): search for the max along each
        column
    value(:,T-t)=V1'; %store the value of each
        iteration
    policy(:,T-t)=p';%store the policy of each
        iteration
    V0=V1';

end

%% 4. Plotting results

figure(1)
plot(k,V1)
title('Value Function against capital stock')
xlabel('capital stock')
ylabel('value function')
%notice: the value function is increasing in the
    current stock of capital

figure(2)
plot(k,p);
title('Policy function against capital stock')
xlabel('capital stock')
ylabel('value function')
```

```
73 % figure(3)
74 % surf(value)
75 % title('Value Function surface')
76 % xlabel('capital today')
77 % ylabel('capital tomorrow')
78 % zlabel('value function')
79
80 %% 5. Simulation of transition dynamics
81
82     P=100; % arbitrary length for transition path
83     capital_index=ones(1,P);
84     capital_transition=ones(1,P);
85     capital_index(1)=(3); % arbitrary starting point in terms of the index
86     %capital_index(1)=(150); % set an initial value above the ss.
87     capital_transition(1)= k(capital_index(1));
88
89     for t=2:P
90     capital_index(t)=(p(capital_index(t-1)));% evolution in index space
91     capital_transition(t)=k(capital_index(t)); % evoluation in capital space
92     end
93
94 figure(4)
95 plot(capital_transition)
96 title('Transitional dynamics for capital')
97 xlabel('time period')
98 ylabel('capital stock')
```

Explanation

- **Lines 1 to 20:** set parameters values, total numer of periods, the steady state capital value of the model and define the capital grid. Make sure to include the ss of capital inside the grid, as done at lines 16 to 20.

- **Lines 24 to 40**: define consumption, available resources and utility function. The matrices look like, *assuming there are only* 3 *grid points* *for k*:

$$k = (k_1 \quad k_2 \quad k_3)$$

$$y = (k_1^\alpha \quad k_2^\alpha \quad k_3^\alpha)$$

$$ytot = \begin{matrix} k_1^\alpha - (1-\delta)k_1 & k_2^\alpha - (1-\delta)k_2 & k_3^\alpha - (1-\delta)k_3 \\ k_1^\alpha - (1-\delta)k_1 & k_2^\alpha - (1-\delta)k_2 & k_3^\alpha - (1-\delta)k_3 \\ k_1^\alpha - (1-\delta)k_1 & k_2^\alpha - (1-\delta)k_2 & k_3^\alpha - (1-\delta)k_3 \end{matrix}$$

$$c = \begin{matrix} k_1^\alpha - (1-\delta)k_1 - k_1' & k_2^\alpha - (1-\delta)k_2 - k_1' & k_3^\alpha - (1-\delta)k_3 - k_1' \\ k_1^\alpha - (1-\delta)k_1 - k_2' & k_2^\alpha - (1-\delta)k_2 - k_2' & k_3^\alpha - (1-\delta)k_3 - k_2' \\ k_1^\alpha - (1-\delta)k_1 - k_3' & k_2^\alpha - (1-\delta)k_2 - k_3' & k_3^\alpha - (1-\delta)k_3 - k_3' \end{matrix}$$

$$u(c) = \begin{matrix} u(k_1^\alpha - (1-\delta)k_1 - k_1') & u(k_2^\alpha - (1-\delta)k_2 - k_1') & u(k_3^\alpha - (1-\delta)k_3 - k_1') \\ u(k_1^\alpha - (1-\delta)k_1 - k_2') & u(k_2^\alpha - (1-\delta)k_2 - k_2') & u(k_3^\alpha - (1-\delta)k_3 - k_2') \\ u(k_1^\alpha - (1-\delta)k_1 - k_3') & u(k_2^\alpha - (1-\delta)k_2 - k_3') & u(k_3^\alpha - (1-\delta)k_3 - k_3') \end{matrix}$$

- **Lines 43 to 45**: Initialize policy and value function.
- **Lines 46 to 59**: Main loop.
  1. Using out initial guess (a matrix of zeros) for the very last period, we find the value at $T-1$.
  2. We then look for the **maximum in each column**. $V1$ is the value of the maximum in each column, while p tells us the position of the maximum (i.e. which value of k' in the grid maximizes our current utility). The first one is our implied value function, the latter the policy function of each iteration. We then create two matrices, *value* and *policy* where we store, in each row, the $V1$ and the p of each iteration. Notice that this method of searching the maximum on the whole capital grid is inefficient, since many point of the matrix will never arise as a maximum. Looking at the policy function p, we see that the optimal capital choice lies indeed along a diagonal starting at the 9^{th} point of the matrix v.
  3. Prepare for the next iteration updating our value function.
- **Lines 63 to 80**: Plotting results
- **Lines 81 to the end**: Simulate transitional dynamics for capital
  1. Set a number of simulation, say P,

2. Set two row matrices of ones, *capital_index* and *capital_transition*, with the same length,
3. Set a random value in the first position of *capital_index* and evaluate *capital_transition* at the value of k corresponding to the random value we chose. ex. if we put a 3 as first value of *capital_index*, then the first value of *capital_transition* must be the third value of the grid k. We can either choose a value below or above the steady state of capital.
4. Within the loop, plug into each position *capital_index* the $P-1$th value of the policy. Then evaluate *capital_transition* at the value of k corresponding to that position. In this way *capital_transition* is a row matrix telling us what is the optimal capital level to choose, given our starting point,
5. Plot *capital_transition* to see the transitional path.