Note on Neoclassical Growth Model: Value Function Iteration + Finite Element Method

Makoto Nakajima, UIUC
January 2007

1 Introduction

We solve the neoclassical growth model using value function iteration, and finite element method. The algorithm requires an optimization algorithm on one-dimensional space. If you are not familiar with these numerical methods, please refer to notes on these methods at first.

Let us start by stating the problem again:

Problem 1 (Neoclassical Growth Model: Recursive Formulation)

$$V(K) = \max_{C\geq 0, K'\geq 0} \{u(C) + \beta V(K')\}$$

subject to

$$C + K' = zF(K, 1) + (1 - \delta)K$$

2 Value Function Iteration with Finite Element Method

The object that we want to find is the optimal value function, which is a function defined over a continuous state space (space of K). Therefore, it is natural to approximate the value function using one of the finite element methods. In this example, let's use the easiest one for the purpose of exposition, the piecewise-linear interpolation. But using more sophisticated finite element method is a trivial extension.

Suppose we can restrict our attention to the compact subset of the state space $[\underline{K}, \overline{K}]$, and suppose we set the knots as follows:

$$\underline{K} = K_1 < K_2 < ... < K_{n_K-1} < K_{n_K} = \overline{K}$$

Linearly interpolated value function is characterized by the value at these n_K points, $\{V_i\}_{i=1}^{n_K}$. For for review, the algorithm to evaluate the piecewise-linear approximated function at an arbitrary point $K \in [\underline{K}, \overline{K}]$ is as follows:

Algorithm 1 (Evaluating Piecewise-Linear Function)

1. *Suppose we have the knots $\{K_i\}_{i=1}^{n_K}$ which are in the increasing order, and the values $\{V_i\}_{i=1}^{n_K}$.*
2. *Suppose we want to evaluate the function at $K \in [\underline{K}, \overline{K}]$.*
3. *Find i such that:*

   $$K_i \leq K \leq K_{i+1}$$

   holds. In other words, find the interval in which K falls into.

4. *Piecewise-linear function evaluated at K, $\tilde{V}(K)$ is:*

$$\tilde{V}(K) = V_i + \frac{V_{i+1} - V_i}{K_{i+1} - K_i}(K - K_i)$$

Suppose there are two value functions which are characterized by $\{V_i^0\}_{i=1}^{n_K}$ and $\{V_i^1\}_{i=1}^{n_K}$, the two value functions are equivalent if and only if:

$$V_i^0 = V_i^1 \qquad \forall i$$

Therefore, we can claim that the two value functions are sufficiently close if, for some small ϵ

$$\max_i |V_i^0 - V_i^1| < \epsilon$$

Using these results, we can construct an algorithm to solve the neoclassical growth model.

Algorithm 2 (Value Function Iteration + Finite Element Method)

1. *Pick the number of knots, n_K.*

2. *Set a tolerance parameter ϵ, which is used to judge whether the value function iteration converged or not.*

3. *Set upperbound and lowerbound of the state space, $\underline{K}$ and $\overline{K}$. As we are interested in the dynamics below the steady state capital stock level, let's set $\overline{K}$ which is slightly higher than the steady state level (which can be computed analytically). As for $\underline{K}$, we can use a value slightly higher than zero. Including zero capital stock might be a problem as it implies zero consumption, and computing utility associated with zero consumption is a critical problem, since the computer usually refuse to compute log or negative power of zero.*

4. *Given the bounds on the state space, set the knots $\{K_i\}_{i=1}^{n_K}$ in the following way:*

   $$\underline{K} = K_1 < K_2 < ... < K_{n_K - 1} < K_{n_K} = \overline{K}$$

   There is no restriction about how to locate the knots, but an obvious candidate is to set the knots so that the distance between two adjacent knots are equal. Notice that wisely allocating the knots could greatly improve the performance of approximation without sacrificing the computational time.

5. *Given the knots, the value function which is approximated using piecewise-linear interpolation can be stored as an array of length n_K. Let's denote a value function as $\{V_i\}_{i=1}^{n_K}$.*

6. *The optimal decision rule $K' = g_K(K)$ can also be approximated using piecewise-linear interpolation. Let's denote an optimal decision rule as $\{g_i\}_{i=1}^{n_K}$.*

7. *Set the initial guess for the value function $\{V_i^0\}_{i=1}^{n_K}$. Two natural choices are:*

   (a) $V_i^0 = 0 \qquad \forall i$

   (b) $V_i^0 = \frac{u(zF(K_i,1)+(1-\delta)K_i - K_i)}{1-\beta}$

   The first option doesn't require an explanation. The second option is the value associated with the case when an agent with K_i capital stock chooses to save K_i capital stock for the next period.

8. *Let's call the value function which is implied by the piecewise-linear interpolation with* $\{V_i^0\}_{i=1}^{n_K}$ *as* $\tilde{V^0}(K)$*. Similarly, denote the optimal decision rule as* $\tilde{g}(K)$*.*

9. *For each* $i = 1, 2, ..., n_K$*, solve the following problem:*

$$g_i = \underset{K' \in [\underline{K}, \overline{K}]}{\arg\max} \{u(zF(K_i, 1) + (1-\delta)K_i - K') + \beta \tilde{V^0}(K')\}$$

You need to use one of the one-dimension optimization algorithm to find an optimum.

Once g_i *is obtained, use it to update value function. Specifically:*

$$V_i^1 = u(zF(K_i, 1) + (1-\delta)K_i - g_i) + \beta \tilde{v^0}(g_i)$$

10. *Compare* $\{V_i^0\}_{i=1}^{n_K}$ *and* $\{V_i^1\}_{i=1}^{n_K}$*. In particular, if:*

$$\max_i |V_i^0 - V_i^1| < \epsilon$$

holds, we are done. Treat $\{V_i^1\}_{i=1}^{n_K}$ *as the optimal value function and* $\{g_i\}_{i=1}^{n_K}$ *as the optimal decision rule associated with the optimal value function.*

11. *Otherwise, update the value function by:*

$$V_i^0 = V_i^1 \qquad \forall i$$

and go back to step 9.

12. *We should check the validity of the settings. First of all, check if the bounds on the state space are not binding. In particular, check if:*

$$\underline{K} < \min_K \tilde{g}(K)$$

and

$$\overline{K} > \max_K \tilde{g}(K)$$

hold.

13. *We also increase* n_K *and reduce* ϵ *and make sure that the results (the objects that you are interested in) are not sensitive to the changes.*

Some comments below:

1. It is almost costless (in terms of computational time) for piecewise-linear interpolation to evaluate the interpolated function. However, for more sophisticated interpolation method, it might not be the case. It might be costly to compute the coefficients which are necessary to evaluate the interpolation function. In this case, it's better to compute the coefficients before using the value function. In the algorithm above, compute the coefficients at the beginning of step 9. Once we get the coefficients associated with the current guess of the value function, We can use the same coefficients to evaluate the value function as long as the value function is not updated.

2. For piecewise-linear interpolation (and many other finite element methods as well), the performance of approximation is better (the error is smaller) if more knots are located in the region with higher absolute curvature. With the problem like ours, it means it is better to put more knots close to $\underline{K}$ and put less knots in the region close to $\overline{K}$. This is because, if the actual value function has small curvature, a linear approximation actually is a very good approximation. So we don't need a lot of knots where the actual value function is close to linear.

3. You can use the algorithm that we learned to speed-up the iteration process, namely *(i)* Howard's policy iteration, *(ii)* exploiting monotonicity of the optimal decision rule, *(iii)* exploiting the concavity of the maximand, and *(iv)* local search. Of course, you can combine some of them.