Lecture 2: Solving dynamic optimization problems with interpolation

1 Solving a dynamic optimization problem by value function iteration and interpolation

For a pure discretization approach (as described in Lecture 1) to be precise, we need a lot of grid points. Indeed, we should expect no more precision than the distance between the gridpoints. But if the value function (or decision rule) is well approximated by either a polynomial or a spline, then we can get away with fewer gridpoints and still achieve a great deal of precision.

In this context, you may want to begin by reading K. Kopecky’s notes on value function iteration.

Here we go straight to the bottom line. For simplicity, suppose we have opted for piecewise linear interpolation so that the coefficients just are the function values at the nodes; \( v(k^i) = \theta_i \).

Then, in the context of the one-sector deterministic growth model with no leisure, we want to iterate on

\[
\theta_i^1 = \max_{k^1 \leq k' \leq k^n} \left\{ u(f(k) - k') + \beta \hat{v}(k'; \theta_0) \right\}
\]

for \( i = 1, 2, \ldots, n \). This is pretty easy provided the maximization problem on the right hand side can be solved. But how to solve it? Clearly we can’t do it analytically, nor will a first-order condition be particularly helpful since \( \hat{v} \) is not necessarily differentiable. So we will take a look at numerical optimization. See my notes on numerical optimization.
2 Solving a dynamic optimization problem by policy function iteration and interpolation

Instead of approximating the value function, we can approximation the decision rule (policy function).

Suppose the solution to a dynamic stochastic optimization problem can be characterized by the following “Euler” equation.

\[ E\left[ f(x_t, x_{t+1}, x_{t+2}, \xi_t, \xi_{t+1}) \mid \mathcal{F}_t \right] = 0 \]

where \( x_t \) is constrained to be predictable with respect to the filtration \( \{\mathcal{F}_t\}_{t=0}^{\infty} \) generated by the stochastic process \( \{\xi_t\}_{t=0}^{\infty} \). This means that there should be functions \( g \) such that

\[ E\left[ f(x, g(x, \xi), g(g(x, \xi), \xi')) \mid \xi \right] \equiv 0. \]

where \( \xi' \) whose distribution conditional on \( \xi \) is the same as that of \( \xi_{t+1} \) conditional on \( \xi_t \), assuming that this distribution is independent of calendar time. To find this (hopefully unique) \( g \) we can proceed as follows. We start by defining a grid (finite set) of values for \( x \) and \( \xi \). We then establish an interpolation rule, approximating \( g \) by the function \( \hat{g}(x, \xi; \theta) \) where \( \theta \) is some finite-dimensional vector and where by definition of interpolation we have that \( g \) and \( \hat{g} \) agree on at the grid points. It is understood that we know how to compute \( \theta \) from the values of \( g \) at the grid points. We then come up with an initial guess \( \theta^0 \) of the parameter vector. To update it, solve, for \( y \), the following equation for each grid point \( k \).

\[ E\left[ f(x_k, y, \hat{g}(y, \xi'; \theta^0)) \mid \xi_k \right] = 0. \]

This gives rise to an updated parameter vector \( \theta^1 \).

This approach, though conceptually straightforward and reasonably robust, though there is no guarantee that it will converge. (Or is there? Apparently there is a literature on this.) A potentially more efficient approach is the following. Consider the equation

\[ E\left[ f(x, \hat{g}(x_k, \xi; \theta), \hat{g}(\hat{g}(x, \xi; \theta), \xi'; \theta)) \mid \xi_k \right] \equiv 0. \]

This is a non-linear system of equations in the unknown vector \( \theta \). If there are just as many grid points as there are parameters in \( \theta \) (and there had better be if we are interpolating) then this
system can in principle be solved as such, using, for instance, Newton’s or Broyden’s method. Such an approach is called a weighted residual method with collocation. (For more context on this, see the virtual handout on projection methods.)

3 Solving a dynamic optimization problem by the method of endogenous grid

Suppose you want to maximize

$$E \left[ \sum_{t=0}^{\infty} \beta^t u(c_t) \right]$$

subject to

$$a_{t+1} + c_t \leq e_t + ra_t$$

where \( \{e_t\} \) is a finite state Markov chain with state space \( \{e^e, e^h\} \) and \( \{c_t\} \) is constrained to be adapted to the filtration generated by this process.

Of course, \( a_0 \geq 0 \) is given. We will also impose, as a constraint, that \( a_t \geq a \) for all \( t \). We call this a “borrowing limit”, a “credit constraint” or a “liquidity constraint”.

For this kind of problem, the method of endogenous grid\(^1\) is the fastest and cleverest that I am aware of. Alas, it forces one to update point by point, but this updating is lightning fast thanks to the fact that it is done analytically.

Consider the first order condition

$$[e^e + ra - y]^{-\sigma} = \beta r \left\{ \gamma \left[ e^e + ry - \hat{h}(y; \theta^{e,0}) \right]^{-\sigma} + (1 - \gamma) \left[ e^h + ry - \hat{h}(y; \theta^{h,0}) \right]^{-\sigma} \right\} = 0$$

This is a messy function of \( y \). Certainly no analytical solution is available. But suppose \( y \) is given. Then solving for \( a \) is a cinch! Using brutal but useful short hand notation, we can write the above foc as

$$[e^e + ra - y]^{-\sigma} = A^e(y; \theta^0)$$

\(^1\) See Carroll (2005).
Evidently
\[ a = \frac{[A^\ell(y; \theta^0)]^{-1/\sigma} - e^\ell + y}{r}. \]

On this idea we can base an algorithm. We begin by defining, once and for all, a grid not for assets but for savings. Denote this savings grid by \( y = \{y_1, y_2, \ldots, y_n\} \) where it is of course a good idea to set \( y^1 = a \) and \( y^n = \bar{a} \). The corresponding grid for assets \( a = \{a_1, a_2, \ldots, a_n\} \) is, as the name of the method suggests, computed. Indeed there will be one for the low earnings state, \( a^\ell \) and one for the high earnings state, \( a^h \).

Define
\[
A^\ell(y; \theta) = \beta r \left\{ \gamma \left[ e^\ell + ry - \hat{h}(y; \theta^\ell) \right]^{-\sigma} + 
\right.
\]
\[
+(1 - \gamma) \left[ e^h + ry - \hat{h}(y; \theta^h) \right]^{-\sigma}
\]
\[= 0 \]

and similarly for \( A^h(y; \theta) \).

Then the algorithm proceeds as follows.

1. Initialize \( a^\ell,0 = y \) and \( a^h,0 = y \)

2. Update according to
\[
a^\ell,1_k = \frac{[A^\ell(y_k; \theta^0)]^{-1/\sigma} - e^\ell + y_k}{r}
\]

and
\[
a^h,1_k = \frac{[A^h(y_k; \theta^0)]^{-1/\sigma} - e^h + y_k}{r}
\]

What about the borrowing constraint? That pretty much takes care of itself! You will find that, endogenously, \( a^1 > a \). To evaluate \( \hat{h}(a, e^\ell) \) below that, just add another gridpoint \( a_0^\ell \) with associated value \( \underline{a} \).

**References**