3 Introduction to dynamic programming

References:

- Ljungqvist and Sargent (2000), Chapters 2 and 3: introductory
- Stokey and Lucas (1989), Chapters 4 and 9: theory
- Acemoglu (2009), Chapters 6 and 16: theory (summary of relevant chapters in Stokey/Lucas)
- Judd (1998), Chapter 12: numerics

3.1 Abstract example: discrete finite-horizon problem

Consider the problem

$$\min_{a_0,\dots,a_T} \mathrm{E} \sum_{t=0}^{T} \beta^t C(x_t, a_t, t) + \beta^{T+1} Q(x_{T+1}) \tag{1}$$

where

$$x_t \in \mathcal{X} = \left\{X^1, X^2, \dots, X^n\right\} \tag{2}$$
$$a_t \in \mathcal{A} = \left\{A^1, A^2, \dots, A^m\right\} \tag{3}$$

and the dynamic transition law is given by

$$x_{t+1} = X^i \text{ with probability } p_i(x_t, a_t, t), \qquad i = 1, \dots, n \tag{4}$$

Here $p_i(x, a, t)$ is the conditional probability that the system will be in state x^i at $t+1$ if we start at t from x_t and apply the policy a_t.

The policy at time t, a_t, can depend only on information available at time t. This information comprises the values of the variables up to time t. In the current example, all the information we need to compute the optimal a_t is x_t. So a_t will be a function of x_t.

We solve the problem recursively by use of Bellman's value function $v(x)$:

1. We start the recursion by

$$v_{T+1}(X^i) = Q(X^i), \qquad i = 1, \dots, n \tag{5}$$

   Store these values in a vector v_{T+1}.

2. Given a vector v_{t+1}, we calculate $v_t(x)$, for each $x \in \mathcal{X}$, recursively by the Bellman equation

$$v_t(X^i) = \min_{a \in \mathcal{A}} \left\{ C(X^i, a, t) + \beta \sum_{j=1}^{n} p_j(X^i, a, t) v_{t+1}(X^j) \right\}, \qquad i = 1, \dots, n \tag{6}$$

The last term in (6) is the expectation of $v_{t+1}(x_{t+1})$, condition on information at time t.

To obtain the minimum, we calculate the rhs of (6) for each $a \in \mathcal{A}$, and take the minimum of these values.

We store the values in the vector v_t.

3. We iterate until we have obtained the vector v_0.

3.2 Infinite horizon

We now change the objective to

$$\min_{a_0, a_1, \dots} \mathrm{E} \sum_{t=0}^{\infty} \beta^t C(x_t, a_t) \tag{7}$$

Note that the cost function now has to be time-independent. We have to assume the same for the transition probabilities $p_i(x, a)$.

We solve the problem by the same iterative algorithm, but repeat step 2 until convergence is achieved. For this to make sense, we first have to show that the iterations will actually converge.

It is useful to express step 2 of the algorithm by the operator $\mathcal{T}$ on the space of value vectors. The i-th component of $\mathcal{T}(v)$ is defined as

$$\mathcal{T}(v)^i = \min_{a \in \mathcal{A}} \left\{ C(X^i, a) + \beta \sum_{j=1}^{n} p_j(X^i, a) v(X^j) \right\} \tag{8}$$

Therefore $v_{t+1} = \mathcal{T}(v_t)$. The solution to the infinite horizon problem v^* is a fixed point of $\mathcal{T}$, which means $\mathcal{T}(v^*) = v^*$.

This operator is a **contraction** with modulus β, which means

$$\|v - w\|_\infty \leq M \implies \|\mathcal{T}(v) - \mathcal{T}(w)\|_\infty \leq \beta M \tag{9}$$

where $\|.\|_\infty$ is the supremum norm.

Proof:

Assume $\|v - w\|_\infty \leq M$ and

$$\mathcal{T}(w)^j - \mathcal{T}(v)^j > \beta M \tag{10}$$

for some j. We will show that this cannot be the case.

Denote by $\hat{a}^j$ the action that is taken at X^j with value function v. This means $\mathcal{T}(v)^j = C(X^j, \hat{a}^j) + \beta \sum_i p_i(X^j, \hat{a}^j) v(X^i)$. With value function w one could chose the same action and obtain a value $C(X^j, \hat{a}^j) + \beta \sum_i p_i(X^j, \hat{a}^j) w(X^i)$.

Since the p_i are positive and sum up to 1, we have

$$\left| \sum_i p_i(X^j, \hat{a}^j)(v(X^i) - w(X^i)) \right| \leq \sum_i p_i(X^j, \hat{a}^j) \left| v(X^i) - w(X^i) \right| \leq \sum_i p_i(X^j, \hat{a}^j) M \leq M \tag{11}$$

But then

$$\mathcal{T}(w)^j \leq C(X^j, \hat{a}^j) + \beta \sum_i p_i(X^j, \hat{a}^j) w(X^i) \leq \mathcal{T}(v)^j + \beta M \tag{12}$$

so that (10) is wrong. Q.E.D.

Then the contraction mapping theorem (Stokey and Lucas 1989, p.50) tells us that the operator $\mathcal{T}$ has a unique fixed point (the solution of the Bellman equation), to which the iterations converge.

Moreover, at each stage k of the iteration, we can give an upper bound for the deviation of the value function solution v^* from the estimate v_k.

Define $\delta_k = \|\mathcal{T}(v_k) - v_k\|_\infty$, then

$$\|v^* - v_k\|_\infty = \|v^* - \mathcal{T}(v_k) + \mathcal{T}(v_k) - v_k\|_\infty = \|\mathcal{T}(v^*) - \mathcal{T}(v_k) + \mathcal{T}(v_k) - v_k\|_\infty \tag{13}$$

$$\leq \|\mathcal{T}(v^*) - \mathcal{T}(v_k)\|_\infty + \|\mathcal{T}(v_k) - v_k\|_\infty \leq \beta\|v^* - v_k\|_\infty + \delta_k \tag{14}$$

and therefore

$$\|v^* - v_k\|_\infty \leq \frac{\delta_k}{1-\beta} \tag{15}$$

3.3 Continuous state space

The problems we have to solve in economics differ from the problems of the last subsections because both the state and the action space are usually not discrete, but continuous (e.g., a continuous range of possible levels of capital). Often they are even unbounded.

To apply the Bellman algorithm to these problems, we have to approximate the continuous problem on a discrete grid. This involves the following issues:

- Interpolating the value of $v(x)$ at points of x not in the grid; cf. Section 1.4.
- In case of an unbounded state space, approximating the problem on a bounded subset.
- Computing the expectations numerically.
- Finding the optimal action (step 2 above) for each $x \in \mathcal{X}$ in each iteration by numerical techniques.

We now modify the problem of Section 1.2 by assuming that both x and a are continuous variable, but we still approximate the value function on the finite grid $\mathcal{X}$. Our problem now is to minimize (7) s.t.

$$x_{t+1} = T(x_t, a_t, z_{t+1}) \tag{16}$$

where z_t is some exogenous i.i.d. process. Both the state x_t and the control a_t are continuous variables. The Bellman equation is

$$v_t(x) = \min_a \{C(x,a) + \beta \,\mathrm{E}_t\, v_{t+1}(T(x,a,z_{t+1}))\} \tag{17}$$

Example: Growth with stochastic discount factor

We maximize

$$\mathrm{E}\sum_{t=0}^{\infty} u(c_t) \tag{18}$$

s.t.

$$K_{t+1} = f(K_t) - \delta_{t+1}K_t - c_t \tag{19}$$

where δ_t is an exogenous stochastic (i.i.d.) discount factor. We assume that $f'(K) \to \infty$ as $K \to 0$, so it is not necessary to impose any lower bounds on K.

The outline of the solution algorithm is the following:

1. Compute the steady state value of K^*.
2. Choose some range of approximation, for example $K \in (0.5K^*, 1.5K^*)$.
3. Choose a set of n grid points $\mathcal{K} = \{K_1, K_2, \ldots, K_n\}$ from this range, for example equidistant points, or points equidistant in logs.
4. Our aim is to find the n-vector v that gives the value function at each grid point. We initialize by $v^0 = 0$.
5. Given the estimate v^k (step k in our iteration), we find v^{k+1} by solving (note that the k goes backward in time)

$$v_i^{k+1} = \max_c \left\{ u(c) + \beta \int \tilde{v}^k \left(f(K_i) - \delta K_i - c\right) dF(\delta) \right\}, \qquad i = 1, \ldots, n \tag{20}$$

   where $F(\delta)$ is the probability distribution of δ, and $\tilde{v}^k(K)$ is the interpolation of the vector v^k between grid points, as explained in Section 1.4.

   The optimization in (20) uses a numerical method like fmin. [If the problem is convex, as it is here, we can alternatively solve the first order condition to find c.]

6. Iterate the last step until the changes in v become negligible.

3.4 Interpolation

For concreteness, we now assume that z follows a discrete distribution and take on n_z different values $z_1, z_2, \ldots, z_{n_z}$, with probabilities $p_1, p_2, \ldots, p_{n_z}$, such that

$$p_j \geq 0 \tag{21}$$

$$\sum_{j=1}^{n_z} p_j = 1 \tag{22}$$

Then the expectation in (17) takes the form[1]

$$\mathrm{E}_t\left[v\left(T(x,a,z_{t+1})\right)\right] = \sum_j p_j v(T(x,a,Z_j)) \tag{23}$$

The problem is that $T(x,a,Z_j)$ will generally not be part of the grid $\mathcal{X}$. To approximate the value $v(x)$ for x not in $\mathcal{X}$, we will choose an interpolation formula

$$v(x) = \sum_i q_i(x)v(x_i), \qquad q_i \geq 0, \quad \sum_i q_i = 1 \tag{24}$$

where the x^i are elements of $\mathcal{X}$. The $q_i(x)$ will be determined below (and most of them will be zero!).

Defining $q_i^j \equiv q_i(T(x,a,Z_j))$, the expectation then becomes

$$\mathrm{E}_t\left[v\left(f(x,a,z_{t+1})\right)\right] = \sum_j p_j(x,a)\sum_i q_i^j v(x^i) = \sum_i p_i^*(x,a)v(x^i) \tag{25}$$

where

$$p_i^*(x,a) = \sum_j p_j(x,a)q_i^j \tag{26}$$

One can easily show that if, for each x^j, the q_i^j have the formal properties of a probability ($q_i^j \geq 0$ and $\sum_i q_i^j = 1$), then the p_i^* are formally also probabilities. And this implies that the contraction property of the value iteration also holds when we use this interpolation (the proof is unchanged).

With the interpolation, reaching point x with certainty is equivalent to reaching the points $x^i \in \mathcal{X}$ with probabilities $p_i^*(x,a)$. The Bellman equation with this interpolation scheme is formally equivalent to the Bellman equation of a stochastic model on a finite grid, and all the convergence properties for this model carry over to our model with the interpolation.

Two interpolation schemes of the form (24) are

- piecewise linear interpolation on simplices (in 2 dimensions: triangles)
- multilinear interpolation

cf. Judd (1998, p.240-244).

The downside of these interpolation schemes is that linear interpolation of the value function usually leads to low accuracy, or put another way, we need a lot of grid points to reach a certain level of accuracy. We might therefore want to use nonlinear interpolation schemes such as polynomials or cubic splines. (Rather than interpolation, we could also use least squares fitting of polynomials or splines to the functions v^k, in each of the iterations). They might in practice give better approximations in most applications, but we cannot prove convergence of the algorithm. In some examples, dynamic programming with cubic spline interpolation really becomes unstable, cf. Judd (1998, p.438).

[1]If z is a continuous variable, we will always approximate the expectation by a “quadrature formula” that amounts to approximate the variable by a discrete distribution z_j such that (21) holds.

Piecewise linear interpolation

Assume the function $v(x)$ is observed at the points $x \in \mathcal{X} = \left\{X^1, X^2, \ldots, X^n\right\}$, where $X^i > X^{i-1}$ for $i = 2, 3, \ldots, n$.

Piecewise linear interpolation in one dimension means that

$$v(x) = \left(1 - \frac{x - X^i}{X^{i+1} - X^i}\right) v(X^i) + \frac{x - X^i}{X^{i+1} - X^i} v(X^{i+1}) \tag{27}$$

where $X^j \leq x \leq X^{j+1}$. Notice that (**??**) is of the form (24), with $q_j = \left(1 - \frac{x-X^i}{X^{i+1}-X^i}\right)$, $q_{j+1} = \frac{x-X^i}{X^{i+1}-X^i}$ and $q_i = 0$ for $i \neq j, j+1$.

Advantages:

- Satisfies (24), so that stochastic interpretation is possible. Convergence guaranteed.
- If $v(x)$ are generated by convex (concave) function, interpolated function is also convex (concave). Notice: this holds for one-dimensional piecewise-linear interpolation, not in higher dimensions.

Disadvantages:

- Low accuracy in many cases.
- Interpolant not differentiable.

Multilinear interpolation

Assume the function $v(x, y)$ is observed at the Cartesian product of the grids $\{\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n\}$ and $\{\bar{y}_1, \bar{y}_2, \ldots, \bar{y}_m\}$. where $\bar{x}_i > \bar{x}_{i-1}$ for $i = 2, 3, \ldots, n$ and $\bar{y}_i > \bar{y}_{i-1}$ for $i = 2, 3, \ldots, m$.

Multilinear interpolation means that

$$\begin{aligned} v(x, y) &= \left(1 - \frac{x - \bar{x}_i}{\bar{x}_{i+1} - \bar{x}_i}\right)\left(1 - \frac{y - \bar{y}_j}{\bar{y}_{j+1} - \bar{y}_i}\right) v(\bar{x}_i, \bar{y}_j) \\ &= \frac{x - \bar{x}_i}{\bar{x}_{i+1} - \bar{x}_i}\left(1 - \frac{y - \bar{y}_j}{\bar{y}_{j+1} - \bar{y}_i}\right) v(\bar{x}_{i+1}, \bar{y}_j) \\ &= \left(1 - \frac{x - \bar{x}_i}{\bar{x}_{i+1} - \bar{x}_i}\right) \frac{y - \bar{y}_j}{\bar{y}_{j+1} - \bar{y}_i} v(\bar{x}_i, \bar{y}_{j+1}) \\ &= \frac{x - \bar{x}_i}{\bar{x}_{i+1} - \bar{x}_i} \frac{y - \bar{y}_j}{\bar{y}_{j+1} - \bar{y}_i} v(\bar{x}_{i+1}, \bar{y}_{j+1}) \end{aligned} \tag{28}$$

where $\bar{x}_i \leq x \leq \bar{x}_{i+1}$. and $\bar{y}_j \leq y \leq \bar{y}_{j+1}$.

Notice that the coefficients multiplying the $v(x,y)$ in (27) are non-negative and sum to unity. Therefore, if $\mathcal{X}$ is the Cartesian grid of (27) is again of the form (24), where now 4 of the indices q_i are non-zero.

The list of advantages and disadvantages of multilinear interpolation is similar to that of piecewise linear, but now, even if the $v(\bar{x}_i, \bar{y}_j$ are generated by a convex function, the interpolated function is not necessarily convex!

Higher-order polynomial interpolation

Options for interpolation in one dimension:

- Interpolation with high-dimensional polynomials: interpolated the n data points $v(\bar{x}_1), v(\bar{x}_2), \ldots, v(\bar{x}_n)$ by a polynomial of order n (degree n-1).
- Least-squares regression: approximate the function given by the n data points $v(\bar{x}_1), v(\bar{x}_2), \ldots, v(\bar{x}_n)$ by a polynomial of order $m < n$. Fit the function by least squares. Notice that the approximate function in general does not go through the data points
- Piecewise polynomial interpolation: splines.

  Example: cubic spline interpolation: between nodes, interpolant is cubic (degree 3) polyonomial. Interplation is twice differentiable at nodes.

Interpolation in more dimensions:

- Polynomials can be generalized to higher dimensions by tensor productions of one-dimensional polynomials, or by complete polynomials.
- Splines are generalized to higher dimensions by tensor B-splines.

Example: Second-order approximations

Let us approximate the value function $v(x)$ as

$$v(x) = a + bx + cx^2 \tag{29}$$

based on the grid $\mathcal{X} = \{-1, 0, 1\}$. Then we get

$$v(-1) = a - b + c \tag{30a}$$
$$v(0) = a \tag{30b}$$
$$v(1) = a + b + c \tag{30c}$$

From this we get

$$a = v(0) \tag{31a}$$
$$b = \frac{v(1) - v(-1)}{2} \tag{31b}$$
$$c = \frac{v(1) + v(-1)}{2} - v(0) \tag{31c}$$

Then (29) becomes

$$v(x) = v(0) + \frac{v(1) - v(-1)}{2} x + \left(\frac{v(1) + v(-1)}{2} - v(0) \right) x^2 \tag{32}$$

This can be rewritten as

$$v(x) = \frac{x^2 - x}{2} v(-1) + (1 - x^2) v(0) + \frac{x + x^2}{2} v(1) \tag{33}$$

Properties of the approximation (32):

- For given data $v(-1)$, $v(0)$ and $v(1)$, (32) is nonlinear (quadratic) in x; this can be seen from (32).
- The approximation $v(x)$ for a given x depends linearly on the data $v(-1)$, $v(0)$ and $v(1)$; this can be seen from (33).

  The coefficients for $v(-1)$, $v(0)$ and $v(1)$ are $\frac{x^2-x}{2}$, $(1 - x^2)$ and $\frac{x+x^2}{2}$, respectively. These coefficients sum to 1 for any x, but they cannot be interpreted as probabilities, because they can be negative. (For example, $\frac{x^2-x}{2} < 0$ for $< x < 1$.)

  Consequence: unlike piecewise linear interpolation, quadratic interpolation cannot be given a stochastic interpretation. Interpolating the value function by (33) is not equivalent to specifying a transition law such that $v(-1)$, $v(0)$ and $v(1)$ are reached with certaint probabilies.

From this example, we can draw the following conclusions for higher-order polynomial approximations:

Disadvantages:

- No stochastic interpretation possible, therefore the convergence of value function iterated is not guaranteed.
- Iteration in policy space breaks down: matrix in general not invertible.

Advantage: if the algorithm converges, it probably achieves higher accuracy than the linear interpolation algorithms.

3.5 Concavity

For concave maximization problems such as the growth model with concave utility, one can show that the value function is also concave in each step of the Bellman iteration. For precise conditions, see Stokey and Lucas (1989, Theorems 4.8 and 9.8).

It is important to note that the good theoretical properties of dynamic programming (such as the contraction property) *do not depend* on concavity. However, in the nonconvex case with continuous policy variable, one must be very careful to find the *global* minimum in (17).

3.6 Iteration in policy space

With infinite horizon problems, we often get the solution more quickly by iterating not in value space, but in policy space.

1. Start with a guess of the value function $v^0(x)$, e.g. $v^0(x) = 0$, for all $x \in \mathcal{X}$.
2. Given $v^k(x)$, compute $a^{k+1}(x)$, for each $x \in \mathcal{X}$, as the value a that minimizes $C(x, a) + \beta \sum_i p_i(x, a) v^k(x^i)$.
3. Compute $v^{k+1}(x)$ by solving
$$v^{k+1}(x) = C(x, a^{k+1}(x)) + \sum_i p_i(x, a^{k+1}(x)) v^{k+1}(x_i) \tag{34}$$
4. Iterate steps 2 and 3 until convergence is achieved.

Remarks:

- $v^k(x)$ is the utility one achieves by starting from state x and following in all periods the policy a^k.
- In vector-matrix notation, equation (34) can be written
$$v^{k+1} = c(a^{k+1}) + \beta \Pi(a^{k+1}) v^{k+1} \tag{35}$$
or equivalently
$$\left(I - \beta \Pi(a^{k+1})\right) v^{k+1} = c(a^{k+1}) \tag{36}$$
This is a system of linear equations and can be solved by the methods that we have discussed. Note that the matrix $\left(I - \beta \Pi(a^{k+1})\right)$
  – is diagonally-dominant. This implies that Gauss-Jacobi and Gauss-Seidel methods converge.
  – is often sparse, which can be exploited in several ways. The matrix can be stored as a sparse matrix, and the system of equations can be solved either by Gauss-Jacobi or Gauss-Seidel, or more complicated and often more efficient methods such as GMRES (see Matlab's "help sparse").
- It is not necessary (and usually not efficient) to solve the system (36) accuractely in each iteration. We can rather alternate one optimization step (find new $a's$) with some (10-50, say) Gauss-Seidel steps on (36).
- One often needs only few iterations in policy space. Therefore the costly step of computing new a's need to be done only a few times.

Convergence

If we abbreviate the matrix $\beta\Pi(a)$ by $R(a)$, the definition of v^k implies that

$$v^k = R(a^k)v^k + c(a^k)$$

for each policy iteration k. Iterative substitution leads to

$$v^k = R^m(a^k)v^k + \sum_{i=0}^{m-1} R^i(a^k)c(a^k)$$

For $m \to \infty$, the first term on the right hand side goes to zero because of the contraction property, and we get

$$v^k = \sum_{i=0}^{\infty} R^i(a^k)c(a^k)$$

Proceeding to policy iteration $k+1$, we have

$$\begin{aligned} v^k &= R(a^k)v^k + c(a^k) \\ &= R(a^{k+1})v^k + c(a^{k+1}) + \gamma \end{aligned}$$

for some vector $\gamma \geq 0$, since, by definition, a^{k+1} minimizes $R(a)v^k + c(a)$ over a. Iterating again, we obtain

$$\begin{aligned} v^k &= \sum_{i=0}^{\infty} R^i(a^{k+1}) \left(c(a^{k+1}) + \gamma\right) \\ &= v^{k+1} + \sum_{i=0}^{\infty} R^i(a^{k+1})\gamma \end{aligned}$$

Since R contains only non-negative elements, we get

$$v^k \geq v^{k+1} \tag{37}$$

The value function is monotonically decreasing in each iteration, and since it is bounded from below, it must converge.

Again, the convergence of the algorithm is not disturbed by the interpolation if we use an interpolation scheme with positive q_i.

3.7 Curse of dimensionality

The problem of dynamic programming on discrete grids is that the computational effort grows *exponentially* with the dimension of the state vector. Assume that we need a grid of 100 points in a one-dimensional problem to reach a certain level of accuracy. Then we need approximately 100^2 grid points in a two-dimensional problem, 100^3 grid points in a three-dimensional problem, etc. This means that the problem becomes untractable even with a moderate number of dimensions.

It may be possible to avoid this problem by exploiting the smoothness of typical economic examples (all relevant functions smooth, or even convex). Then one can approximate the value function by complete polynomials (Judd 1998, Section 12.7). However, the convergence of the DP algorithm with nonlinear approximations of the value function cannot be proven, cf. Section 1.4

References

Acemoglu, D. (2009). *Introduction to Modern Economic Growth.* Princeton University Press.

Judd, K. L. (1998). *Numerical Methods in Economics.* Cambridge and London: MIT Press.

Ljungqvist, L. and T. J. Sargent (2000). *Recursive Macroeconomic Theory.* Cambridge, Ma.: MIT Press.

Stokey, N. L. and R. E. Lucas (1989). *Recursive Methods in Economic Dynamics.* Cambridge, MA: Harvard University Press.