

# 1 Finite State Space Dynamic Programming

Assume that the state space is finite:  $k, k' \in \{k_1, \dots, k_n\}$ . The value function then takes on only finitely-many values:

$$v(k_i) = \max_{k' \in \{k_1, \dots, k_n\}} \{u(f(k_i) + (1 - \delta)k_i - k') + \beta v(k')\}.$$

Straightforward grid search can be implemented to solve this functional equation. For each  $i$ , evaluate the RHS at each grid point for  $k'$ :

$$\widehat{v}(k_i, k_j) = u(f(k_i) + (1 - \delta)k_i - k_j) + \beta v(k_j).$$

Choose the  $j$  that attains the maximum value of  $\widehat{v}$  and then update the value function guess:

$$v^{n+1}(k_i) = u(f(k_i) + (1 - \delta)k_i - k_{j^*}) + \beta v(k_{j^*}).$$

It is trivial to prove that the RHS has a solution (finite number of possibilities) and that the above procedure defines a contraction map with a linear convergence rate equal to  $\beta$ .

Since the above approach is quite slow, we will exploit some methods to speed it up. First, since we know that the policy function is nondecreasing, we need only search over a limited range of  $k'$ . Assume that  $j^*(k_{i-1})$  is the policy choice at grid point  $i - 1$ . Then we can begin our search for grid point  $i$  at this value, rather than  $k_1$ , since it cannot be optimal to choose less capital when endowed with more (although the planner may choose the same  $j^*$  at  $i$  as at  $i - 1$ ). Second, we know that the true objective on the RHS is concave (as opposed to the discrete approximation), so that we need only search until the objective begins to decline; that is, if

$$\widehat{v}(k_i, k_j) > \widehat{v}(k_i, k_{j+1}),$$

$j$  is the optimal choice (provided that you guessed a discrete set of points that actually are concave). Third, we can evaluate the current return function  $u(f(k_i) + (1 - \delta)k_i - k_j)$  in advance, limiting the repeated evaluation of non-linear functions.

A fourth acceleration tool is called Howard's improvement algorithm. Assume we have computed the entire set of optimal choices for our current guess  $v^n$ :

$$k'(k) = \{k_{j^*(1)}, \dots, k_{j^*(n)}\}.$$

Normally one would update the value function once and then recompute the optimal choices using the updated value function. Howard's improvement algorithm says to update many times without recomputing the decision rules. That is, do the following update:

$$\begin{aligned} \widetilde{v}^0(k_i) &= u(f(k_i) + (1 - \delta)k_i - k_{j^*(i)}) + \beta v^n(k_{j^*(i)}) \\ \widetilde{v}^{n+1}(k_i) &= u(f(k_i) + (1 - \delta)k_i - k_{j^*(i)}) + \beta \widetilde{v}^n(k_{j^*(i)}) \end{aligned}$$

and so on. This defines a contraction map with locally quadratic convergence; call the converged value  $\tilde{v}^*(k_i)$ . Then set

$$v^{n+1}(k_i) = \tilde{v}^*(k_i).$$

We save ourselves the task of computing decision rules that are ultimately simply discarded, since they depend on the wrong value functions. Each value iteration now takes more time but we do about two orders of magnitude fewer iterations, so total computing time declines a lot. For some problems you may not want to do this step to convergence, but instead some prespecified number of times, because the procedure can be unstable.

Finally, we can implement a Gauss-Seidel sweep through the state space (pure value iteration solves the system of equations using Gauss-Jacobi sweeping). In this procedure, we use the updated value function  $v^{n+1}(k_i)$  as the continuation value; that is, instead of

$$v^{n+1}(k_i) = \max_{k' \in \{k_1, \dots, k_n\}} \{u(f(k_i) + (1 - \delta)k_i - k') + \beta v^n(k')\}$$

we have

$$v^{n+1}(k_i) = \max_{k' \in \{k_1, \dots, k_n\}} \{u(f(k_i) + (1 - \delta)k_i - k') + \beta \mathbf{1}(k' \leq k_i) v^{n+1}(k') + \beta \mathbf{1}(k' > k_i) v^n(k')\}$$

where  $\mathbf{1}(A)$  is the indicator function for set  $A$ . It can be shown that both Howard's improvement and Gauss-Seidel sweeping generate contraction mappings that converge faster than pure value iteration. Furthermore, they are not significant computational burdens and can be combined into a powerful and fast algorithm for discrete models. More tricks can be found in the OR literature on Markov decision problems – see, for example, Martin Puterman's book *Markov Decision Processes: Discrete Stochastic Dynamic Programming* (John H Wiley Series in Probability and Statistics) or Warren Powell's book *Approximate Dynamic Programming: Solving the Curses of Dimensionality* (John H Wiley Series in Probability and Statistics). If one has a large number of possible choices to make (say, because there are a number of assets or other static choices) then the use of specialized optimization routines in place of systematic grid search may be desirable. It seems likely that integer programming routines can be exploited to solve these problems, although I am not aware of anyone who has actually implemented this approach.

You may ask yourself whether the contraction maps defined by adding either Howard's improvement or Gauss-Seidel sweeping converge to the same fixed point as pure value iteration does. They do, in fact, which can be checked easily.

The problem with discrete state space models is that the curse of dimensionality can become problematic, particularly when there is uncertainty. There are three aspects to the curse of dimensionality: the dimension of the state space, the dimension of the choice space, and the dimension of any expectations. We will deal with the dimensionality of the state and choice spaces by parametrizing

the value function in a continuous and differentiable manner and using the tools of convex programming. The expectations space issue will be handled using tools from numerical integration.