Chapter 13

Dynamic Programming: Theory

Dynamic Programming is a set of powerful tools and methods for analyzing a broad set of sequential (or dynamic) decision problems that appear in virtually all fields of economics, as well as in applications in engineering, operations research, and so on. A *sequential decision problem* can be thought of as one where at any given point in time a decision maker takes an action which (i) yields an immediate reward or utility (or cost) and (ii) modifies state of the problem starting from tomorrow. A broad range of frameworks developed to study substantively different questions—such as Markov decision problems, dynamic games, and stochastic optimal control problems—turn out to share very similar mathematical structure and dynamic programming provides a unifying approach to analyze all such problems.

Mastery of the theory of dynamic programming at the Phd level is an essential prerequisite for this book and I will assume that the reader has already studied these tools carefully.[1] For completeness, this chapter reviews the main results of dynamic programming without going into details.

The results of dynamic programming can be presented for environments with differing levels of generality, so before moving forward it is useful to provide a classification of these environments and how they compare to each other. To understand the taxonomy that will follow, it is useful to note that, broadly speaking, the results of dynamic programming theory can be viewed in two categories: (i) substantive

[1] Sundaram (1996) is an excellent first reading that builds toward dynamic programming starting from the basics. Acemoglu (2009, Ch. 6 and 16) contains a concise yet precise exposition of the subject. The gold standard for textbook treatment of this subject remains Stokey et al. (1989), although it can be a daunting read as a first text on the subject. Bertsekas (2001, 2005) provide a comprehensive treatment of the subject, and present applications from a very broad background in sciences and engineering.

results, and (ii) technical (measurability) considerations.[2] By this I mean, there are a number of key ideas that give the theory its immense power—such as the equivalence between sequence and recursive problems, the contraction mapping, the characterization of the value function and policy rules, and so on—and some more technical issues that do not necessarily add new insights but makes sure that the whole theory in its most general form is well specified mathematically.

With these considerations in mind, it is useful to consider the following taxonomy of dynamic programming problems. The first class of problems are deterministic—they contain no probabilistic elements of any kind. This class provides the simplest framework in which most of the basic ideas (substantive considerations) of dynamic programming can be understood. Having said that, of course, randomness is a key aspect of many real life decision problems, and the question is: how do we introduce it? The second class is the stochastic decision problems, where random outcomes are restricted to a domain that is either finite or *countably* infinite. It turns out that this case can be studied with relatively straightforward extensions of the results from case I. Furthermore, this case contains arguably all the substantive ideas of this theory. One drawback is that this case does not allow for a continuous space for randomness, which can sometimes be a useful theoretical modeling tool. So, the third case is the stochastic decision problems with *uncountably* infinite (e.g., continuous) support for stochastic elements. This latter case raises a whole host of technical issues involving measurability that requires substantially more advanced probability theory.[3]

For the purposes of this book, the sweet spot is case II and below I focus on it. It contains all the essential ideas without burdening the reader with extra details that do not provide a deeper understanding of the method.

13.1 Monotone Mappings

It would not be an overstatement to say that the theory of dynamic programming was born out of the realization that the power of fixed point theory (or monotone mappings) can be applied to study sequential decision problems.[4] So what are monotone mappings and what properties do they possess?

The main ideas can be most easily explained in a deterministic framework, so in this section I shall abstract from uncertainty. First let us formally define two closely-

[2] Following Bertsekas and Shreve (1978)'s terminology.

[3] There are various ways to address these measurability concerns, as discussed in detail in Bertsekas and Shreve (1978). Among these, the approach adopted most closely by economists is the one pioneered by Blackwell (1965) and developed more fully by Bertsekas and Shreve (1978), which imposes Borel-measurability on all relevant functions and objects from the outset. This is also the approach followed by Stokey et al. (1989).

[4] Similarly, fixed point theory also underlies the key ideas in game theory (used in the earliest work by John Nash) as well as in general equilibrium theory by Kenneth Arrow and Gerard Debreu.

related mappings. The first one is what I will refer to as the Bellman mapping and is defined as:

$$TJ := \max_{y \in \Gamma(x)} \left[U(x, y) + \beta J(y) \right]. \tag{13.1.1}$$

Notice that this is simply the right-hand-side of the Bellman equation, where J denotes the value function and U denotes the period return function. This mapping can be split into two steps. The first stage is a mapping that evaluates the right hand side of the Bellman equation for a fixed policy (let us denote it with $\hat{y}_t$):

$$T_{\hat{y}(x)} J := \left[U(x, \hat{y}(x)) + \beta J(\hat{y}(x)) \right].$$

I refer to this second mapping as Howard's mapping, because of his key insight in how to use it to speed of the computation of a solution, discussed in more detail later (Howard (1960)). The Bellman mapping is simply the maximized value of the Howard mapping:

$$TJ = \max_{\hat{y}(x) \in \Gamma(x)} T_{\hat{y}(x)} J.$$

If J and U are bounded continuous functions, it can be shown that both T and T_y map this space into itself.[5] And as it turns out both of these mappings are monotone. These features will play a key role in establishing the contraction mapping theorem.

Definition 13.1. Contraction Mapping. Let (S, d) be a metric space and $T : S \to S$ be a mapping of S into itself. T is a contraction mapping with modulus β, if for some $\beta \in (0, 1)$ we have

$$d(Tv_1, Tv_2) \leq \beta d(v_1, v_2)$$

for all $v_1, v_2 \in S$.[6]

It turns out that both the Bellman and Howard mappings defined above are contraction mappings. This can be easily shown by appealing to a sufficient set of conditions established by Blackwell (1965).

Theorem 13.2. [Blackwell's Sufficiency Conditions] *Let $X \subseteq \mathbb{R}^K$ and $B(X)$*

[5] Unbounded returns can be dealt with; see, e.g., Alvarez and Stokey (1998) for a full treatment of the case with homogenous return functions and constraints. The key to establishing results is to impose sufficient restrictions to bound the growth rate of the state variables from above or below (along some or all feasible paths), which in turn ensures that the objective is bounded along the optimal path(s). However, this additional generality comes at the expense of extra notation and technical arguments that do not add insights that are relevant for the purposes of this book. Therefore, I shall confine the analysis to the case of bounded returns throughout this chapter.

[6] The definition of a contraction given here is a bit stronger than it needs to be. Most of the following results would go through if we simply consider mappings that contract all elements of the space after N repeated application of T for some $N > 0$. See for example, Bertsekas and Shreve (1978), pages 52–54.

be the space of bounded functions $f : X \to \mathbb{R}$ defined on X equipped with the sup-norm. Suppose that $B'(X) \subset B(X)$ is a subset and let $T : B'(X) \to B'(X)$ be a mapping satisfying the following two conditions:

A *Monotonicity: For any $f, g \in B'(X)$ and $f(x) \leq g(x)$, for all $x \in X$, implies $Tf(x) \leq Tg(x)$ for all $x \in X$.*

B *Discounting: There exists a $\beta \in (0, 1)$ such that*

$$T(f + c)(x) \leq Tf(x) + \beta c$$

for all $f \in B(X)$, $x \in X$, and positive constants c.

Then, T is a contraction mapping on $B'(X)$ with modulus β.

These sufficiency conditions are especially useful for our purposes, because we will be interested in spaces of functions that satisfy certain properties (such as boundedness or continuity, etc.), and checking whether a mapping of functions is a contraction can, in general, be very challenging. In contrast, Blackwell's conditions are often simple to verify. Notice also that the distinction between $B(X)$ and $B'(X)$ is to emphasize that, depending on the application, we can take a subset of the full space—such as the space of bounded, continuous, and concave functions—and would need to only verify the sufficiency conditions for functions in this subset to establish the contraction property, which is typically easier to do.

We next state the contraction mapping theorem, a simple but very useful fixed point theorem. It only requires the space to be a complete metric space and the mapping to be a contraction. For example, the space of continuous bounded functions endowed with the sup-norm is a complete metric space making the results of this theorem applicable to dynamic programming problems (with bounded returns. More on this later.)

Theorem 13.3. [Contraction Mapping Theorem] *Let (S,d) be a complete metric space and suppose that $T : S \to S$ is a contraction mapping. Then, T has a unique fixed point $v^* \in S$ such that*

$$Tv^* = v^* = \lim_{N \to \infty} T^N v_0$$

for all $v_0 \in S$.[7]

In the next section, we present the main theoretical results of dynamic programming theory. In that context, S will be thought of as the space of functions (that

[7]The logic of the proof is very simple. Repeated applications of the contraction mapping generates a sequence $\{v_0, v_1, ...\}$ that is a Cauchy sequence. Observing that all Cauchy sequences converge to a limit point in complete metric spaces delivers the desired result.

possess certain properties, such as boundedness and continuity) and the proofs will rely on showing that the recursive problem defines a mapping that is a contraction that maps the space of functions into itself. For example, the contraction mapping theorem is the key result that establishes the existence and uniqueness of the solution to the Bellman equation given in (13.2.2).

In addition to proving existence and uniqueness, another key application of the contraction mapping theorem is for characterizing the properties of the fixed point, v^*, such as strict concavity or strict monotonicity (Theorems 13.10 and 13.11 below). It might be tempting to think that we can take S to be the set of strictly concave functions and apply the theorem, which would establish that the fixed point itself is strictly concave—since $v^* \in S$. Unfortunately, this is not possible, because the set of strictly concave functions is not a complete metric space, making the contraction mapping theorem inapplicable.[8] In such instances, the following corollary to Theorem 13.3 will be very useful.

Corollary 13.4. *Let (S,d) be a complete metric space and $T : S \to S$ be a contraction mapping with $Tv^* = v^*$.*

a. If $\overline{S}$ is a closed subset of S, and $T(\overline{S}) \subset \overline{S}$, then $v^* \in \overline{S}$.

b. If, in addition, $T(\overline{S}) \subset \tilde{S} \subset \overline{S}$, then $v^* \in \tilde{S}$.

To see how this result helps, take $\overline{S}$ to be the space of continuous, bounded, concave real-valued functions (endowed with the sup-norm). This space is a complete metric space. Then define $\tilde{S}$ to be a proper subset of $\overline{S}$ by adding the condition of *strict* concavity. This set is not a complete metric space, but that is fine. All we need show is that T maps elements of $\overline{S}$, which are concave, into functions in $\tilde{S}$, which are strictly concave. This can be ensured in dynamic programming applications we will study later by assuming that the period utility (or reward) function is strictly concave. Then applying part (b) of the corollary establishes that the fixed point will also be strictly concave.

For the purposes of computation, equally important is that the contraction mapping theorem provides a convenient way of obtaining this solution, under very general conditions. Specifically starting from any initial guess for the value function that is bounded and continuous, repeatedly applying the mapping T yields convergence to the unique solution. If this sounds familiar, it is because this is the classic value function iteration method for solving a dynamic program, which we shall cover in a moment. Furthermore, the modulus of the mapping is β (the time

[8] Notice that while the space of continuous, bounded, and (weakly) concave real-valued functions endowed with the sup-norm is a complete metric space, it ceases to be so once we strengthen the requirement to *strict* concavity.

discount rate) and determines the rate of convergence (which is linear in β).[9] In particular, the contraction property can be used to derive a simple bound for the maximum deviation between the current iterate and the (yet unknown true fixed point v^* :

Corollary 13.5. *Let $\|\cdot\|_\infty$ denote the supnorm. The maximum deviation between the fixed point and the n-th iterate is bounded as follows:*

$$\|v^* - v_n\|_\infty \leq \frac{1}{1-\beta} \|v_{n+1} - v_n\|_\infty . \tag{13.1.2}$$

This corollary provides a way to estimate the remaining distance to the fixed point with the right hand side of this expression. It is often used in stopping rules in dynamic programming algorithms. As we shall see in the next chapter, tighter error bounds have been derived[10] in subsequent work that we shall use in implementing some algorithms.

13.2 Main Theoretical Results

Understanding the basic theoretical results on dynamic programming is critical for numerically solving a dynamic programming problem. It is especially critical to understand what assumptions are necessary for delivering concavity and differentiability of the value function. For completeness, I briefly review and state the main results here without proof.

Let $x_t \in \mathcal{X} \subset \mathbb{R}^N$ represent the (continuous) endogenous state and $z_t \in \mathcal{Z} \subset \mathbb{Z}$ represent the discrete exogenous state of a system that follows a first-order Markov chain, with transition matrix denoted with $Q(z, z')$. Every period, an individual can choose among a set of actions, consistent with the constraint set given the current state: $y_t \in \Gamma(x_t, z_t) \subset \mathcal{X} \subset \mathbb{R}^N$. For example, Γ can be a budget constraint for an individual, or a resource constraint for an aggregate economy. A reward function, $U(x_t, y_t, z_t)$, assigns a value to action y_t in each state, so: $U : \mathcal{X} \times \mathcal{X} \times \mathcal{Z} \to \mathbb{R}$. Finally, next period's state is determined as:

$$x_{t+1} = G(x_t, y_t, z_t).$$

Given this relationship, choosing y_t for given (x_t, z_t) is the same as choosing x_{t+1}, we will alternatively use y_t and x_{t+1} as the choice variable when convenient. Finally

[9] If there is no discounting, e.g., $\beta = 1$, then we can still define a meaningful DP problem by focusing on average value function, not total. See Bertsekas (2001) for how this is done. Furthermore, an alternative way to write a Bellman equation is by defining the value function as: $V(x) = \mathbb{E}\left(\sum_{t=1}^{\infty} \beta^t u(x_t)\right)$, so that the Bellman equation becomes: $V(x) = \max \mathbb{E}(u(x) + \beta V(x')|x)$. Bertsekas (2001) and Powell (2011) show how this can suggest a set of new computational techniques for solving DP problems more efficiently in high dimension.

[10] Most notably by MacQueen (1966) and Porteus (1971).

let $0 < \beta < 1$ be the discount factor that applies to future rewards.

To properly set up the sequence problem, define the history of shocks (exogenous state) as $z^t \equiv (z_1, z_2, ..., z_t)$, which are vectors in the product space $\mathcal{Z}^t$, and define a contingent plan as $x_{t+1} = \tilde{x}(z^t)$ for every history z^t.

Sequence Problem: Consider the lifetime optimization problem in sequence form:

$$\tilde{V}(x_0, z_0) = \max_{\{\tilde{x}(z^t)\}_{t=0}^{\infty}} \mathbb{E}_0 \left[\sum_{t=0}^{\infty} \beta^t U(\tilde{x}(z^{t-1}), \tilde{x}(z^t), z_t) \right] \tag{13.2.1}$$
$$\text{s.t.} \quad \tilde{x}(z^t) \in \Gamma(\tilde{x}(z^{t-1}), z_t) \qquad \forall t \geq 0, \text{ and } x_0, z_0 \text{ given.}$$

Recursive Problem: Now consider the recursive functional (Bellman) equation:

$$V(x_t, z_t) = \max_{y_t \in \Gamma(x_t, z_t)} \left[U(x_t, y_t, z_t) + \beta \mathbb{E}(V(y_t, z_{t+1}) | z_t) \right]. \tag{13.2.2}$$

It will be useful to define the set of all feasible plans (infinite sequences) starting from the current state (x_t, z_t) that can be generated from the period constraint set. Specifically, let

$$\pi(x_t, z_t) = \{\{\tilde{x}(z^\tau)\}_{\tau=t}^{\infty}, \tilde{x}(z^\tau) \in \Gamma(\tilde{x}(z^{\tau-1}), z_\tau), \quad \text{for } \tau = t, t+1, ...\}$$

be the set of all feasible plans starting from (x_t, z_t).

Before moving ahead, it is useful to study a concrete example.

Example 13.6. An Income Fluctuation Problem. Consider an individual who derives utility from consumption according to function $\mathcal{U}(c_t)$, earns a stochastic income stream y_t, and trades a single riskless bond to smooth consumption. His lifetime utility maximization problem is thus:

$$\tilde{V}(k_0, y_0) = \max_{\{c_t, k_t\}_{t=0}^{\infty}} \mathbb{E}_0 \left[\sum_{t=0}^{\infty} \beta^t \mathcal{U}(c_t) \right] \tag{13.2.3}$$
$$\text{s.t.} \quad c_t + k_t = (1 + r)k_{t-1} + y_t \qquad \forall t \geq 0, \tag{13.2.4}$$
$$k_t \geq k_{\min}, \text{ and } k_0, y_0 \text{ given.} \tag{13.2.5}$$

Comparing this problem to the sequence problem in (13.2.1), it is clear that the latter does not have the same structure. For one thing, the sequence problem above is written entirely in terms of a single sequence whose values at t and $t-1$ enter the return function U, and the budget constraint for today's choice is in terms of yesterday's choice and today's shock $(\tilde{x}(z^{t-1}), z_t)$. To put the income fluctuation problem in this structure, use the budget constraint (13.2.4) to substitute $c_t =$

$(1+r)k_{t-1} - k_t + y_t$ into the objective to get

$$U(k_t, k_{t-1}, y_t) \equiv \mathcal{U}((1+r)k_{t-1} - k_t + y_t),$$

and the choice variable in period t is:

$$k_t \in \Gamma(k_{t-1}, y_t) \equiv [k_{\min}, (1+r)k_{t-1} + y_t].$$

Using the sequence notation for the history of shocks, we have:

$$\tilde{V}(k_0, y_0) = \max_{\{\tilde{k}(y^t)\}_{t=0}^{\infty}} \mathbb{E}_0 \left[\sum_{t=0}^{\infty} \beta^t U(\tilde{k}(y^t), \tilde{k}(y^{t-1}), y_t) \right] \tag{13.2.6}$$

$$\text{s.t.} \quad \tilde{k}(y^t) \in \Gamma(\tilde{k}(y^{t-1}), y_t) \equiv [k_{\min}, (1+r)\tilde{k}(y^{t-1}) + y_t], \tag{13.2.7}$$

which has exactly the same structure as the sequence problem (13.2.1).

One point of this exercise is to remind the reader that the reward function U has different arguments from the utility function $\mathcal{U}$. In particular, it is obtained by substituting some of the constraints into the utility function. Therefore, it depends both on the current state k_t as well as today's choice variable (and tomorrow's state) k_{t+1} and the stochastic stream, z_t. Furthermore, the constraint set for k_t is a closed and convex interval. These points are useful to remember when evaluating some of the assumptions that we will need to impose on U and Γ in a moment. ■

To obtain our first set of theoretical results, we make two assumptions.

Assumption 1. For all $(x, z) \in \mathcal{X} \times \mathcal{Z}$, the constraint correspondence Γ is *nonempty valued* and for all initial conditions (x_0, z_0) the value of the sequence problem $\tilde{V}(x_0, z_0)$ *exists and is finite.*

Now let A be the graph of Γ, that is $A = \{(x, y, z) \in \mathcal{X} \times \mathcal{X} \times \mathcal{Z} \to \mathbb{R} : \ y \in \Gamma(x, z)\}$.

Assumption 2. $\mathcal{X}$ is *compact* and Γ is *nonempty valued*, *compact-valued*, and *continuous with a continuous graph* A.

Theorem 13.7. [Equivalence of Problems]. *Under assumption 1, any solution $\tilde{V}(x, z)$ to the sequence problem (13.2.1) is also a solution to the recursive problem (13.2.2) and vice versa for $V(x, z)$. Therefore, for all $(x, z) \in \mathcal{X} \times \mathcal{Z}$, we have $\tilde{V}(x, z) = V(x, z)$.*

Notice that Theorem 13.7 does not establish the existence of a solution to either problem. It merely states that if such solutions exist, they must coincide.

Theorem 13.8. [Principle of Optimality] *Suppose that Assumption 1 holds. If a feasible plan* $\{\tilde{x}(z^\tau)\}_{\tau=0}^{\infty} \in \pi(x_0, z_0)$ *attains the maximum in the sequence problem then it also attains the maximum in the recursive problem. And the same is true in the opposite direction.*

Theorem 13.9. [Existence of Optimum] *Existence of OptimumUnder Assumptions 1 and 2, there exists a unique value function* V*, which is continuous and bounded in* x *for each* $z \in \mathcal{Z}$*. Further, for every initial state* $(x_0, z_0) \in \mathcal{X} \times \mathcal{Z}$ *an optimal plan* $\{\tilde{x}^*(z^\tau)\}_{\tau=0}^{\infty} \in \pi(x_0, z_0)$*.*

To summarize these three theorems, under mild conditions on the constraint correspondence Γ and finiteness of lifetime utility, the sequence problem and recursive problem coincide: the value function and decision rules that solve the recursive problem generate sequences that solve the sequence problem for any initial conditions and deliver the same lifetime utility. The existence of a solution relies on Weierstrass' theorem and therefore requires stronger assumptions, namely compactness and continuity provided by Assumption 2. These assumptions are rather mild and are satisfied by a very broad range of economic problems, making dynamic programming a very useful tools.

However, for computation, we want to know more about the properties of the solution. For this, we need to make further assumptions.

Assumption 3. [Strict Concavity of Period Utility] For all $z \in \mathcal{Z}$, $U(.,.,z)$ is *strictly concave* in (x, y) over A. That is, for any $\theta \in (0, 1)$ and (x, y) and $(x', y') \in A$, we have

$$U(\theta(x, y) + (1 - \theta)(x', y'), z) \geq \theta U(x, y, z) + (1 - \theta)U(x', y', z)$$

with *strict inequality* when $x \neq x'$.

Assumption 4. [Convex Choice Set] For all $z \in \mathcal{Z}$, $\Gamma(., z)$ is *convex in* x. That is, for any $\theta \in (0, 1)$ and x and $x' \in \mathcal{X}$, $y \in \Gamma(x, z)$ and $y' \in \Gamma(x', z)$ implies that

$$\theta y + (1 - \theta)y' \in \Gamma(\theta x + (1 - \theta)x', z).$$

The assumption of a convex choice set is critical for establishing the concavity of the value function as we shall we see in a moment. A convex choice set rules out increasing returns. For example, suppose that y represents consumption and $f(x)$ is output produced with capital level x. So $\Gamma(x, z) = \{y \in \mathcal{X} : 0 < y < zf(x)\}$. Convexity implies: $\theta y + (1 - \theta)y' < z(\theta f(x) + (1 - \theta)f(x'))$. Therefore, f needs to be concave for Γ to be convex.

Another case ruled out by Assumption 4 is the discretization of the choice space, which was a technique used in the early computational literature and is now all but irrelevant.[11] This approach breaks the convexity of the choice set and therefore can give rise to value functions that are not strictly concave.

Theorem 13.10. [Strict Concavity of V] *Under Assumptions 1–4, the unique value function that satisfies (13.2.2) is strictly concave in x for every $x \in \mathcal{Z}$. Moreover, a continuous policy function, $g : \mathcal{X} \times \mathcal{Z} \to \mathcal{X}$, exists such that the optimal plan satisfies $\tilde{x}^*(z^\tau) = g(x_t^*, z_t)$.*

Assumptions 1 and 2 are more technical in nature and are usually satisfied in a broad set of problems. Thus, concavity of the return function U in both of its arguments, ensured by assumption 3, and the convexity of the choice set (assumption 4), are very often the critically important ones. The strict concavity of the value function is very useful and important in a variety of contexts (e.g., it is required for the Euler equation to be sufficient), so it is important to make sure whether assumption 3 is satisfied or not in a problem before we attempt to solve it numerically.

Assumption 5. For all $(y, z) \in \mathcal{X} \times \mathcal{Z}$, $U(., y, z)$ is strictly increasing in x and Γ is a monotone correspondence: if $x \leq x'$ then $\Gamma(x, z) \subset \Gamma(x', z)$.

Theorem 13.11. [Strict Monotonicity of V] *Under Assumptions 1, 2, and 5, the optimal value function $V(x, z)$ is strictly increasing in x for all $z \in \mathcal{Z}$.*

Assumption 6. $U(x, y, z)$ is continuously differentiable in x in the interior of its domain.

Theorem 13.12. [Differentiability of V] *Under Assumptions 1-3 and 6, the optimal value function $V(x, z)$ is differentiable with respect to x in the interior of x's domain.*

Differentiability is another key property that, if violated, makes a number of techniques inapplicable or less powerful. Thus, it is essential to figure out whether or not its assumptions are satisfied. The only additional assumptions needed, relative, to strict concavity, is the continuous differentiability of the period utility function in the current state, x. This is violated more often than one might think, leading to kink(s) in the value function. One of the most common scenario is if our problem has occasionally binding constraints, or fixed costs. etc.

[[[

[11] Having said that, one can find several recent papers that still use this approach in the name of convenience. As I argue later, there is no excuse for this approach with today's computational resources and the practicality of numerical techniques that allows one to avoid this approach.

13.3 Examples

Example 13.13. Analytical Solution of Dynamic Programming Problem. In this example, we show how to apply the Bellman operator repeatedly (as dictated by the contraction mapping theorem) to solve a dynamic programming problem. This is a special example that has a closed-form solution, which is an exception rather than the rule.

Consider the following stylized version of the neoclassical growth model in a deterministic setting, with log utility over consumption, a Cobb-Douglas production function, $y = Ak^{\alpha}$, and full depreciation of the capital stock:

$$\begin{aligned} V(k) &= \max_{c,k'} \{\log c + \beta V(k')\} \\ \text{s.t} \quad c &= Ak^{\alpha} - k'. \end{aligned}$$

Rewrite the Bellman equation as:

$$V(k) = \max_{k'} \{\log(Ak^{\alpha} - k') + \beta V(k')\}. \tag{13.3.1}$$

Our goal is to find $V(k)$ and a decision rule $k' = g(k)$. As our initial guess, we take $V_0(k) \equiv 0$ (which would be the natural choice for the last period of life in a finite-horizon problem with no bequest motive). Therefore, we have:

$$\begin{aligned} V_1(k) = TV_0(k) &= \max_{k'} \{\log(Ak^{\alpha} - k') + V_0(k')\} \Rightarrow k' = 0 \\ \Rightarrow V_1(k) &= \log A + \alpha \log k. \end{aligned}$$

Now, substitute V_1 into the RHS of V_2 :

$$\begin{aligned} V_2(k) = TV_1(k) &= \max_{k'} \{\log(Ak^{\alpha} - k') + \beta(\log A + \alpha \log k')\} \\ \Rightarrow \text{FOC}: \quad \frac{1}{Ak^{\alpha} - k'} &= \frac{\beta\alpha}{k'} \quad \Longrightarrow \quad k' = \frac{\alpha\beta}{1+\alpha\beta} \times \underbrace{Ak^{\alpha}}_{y}. \end{aligned}$$

Therefore, the optimal policy is to save a constant fraction of output, which we can substitute into the RHS to obtain V_2:

$$V_2(k) = \left[(1+\beta+\alpha\beta)\log A + \log\frac{1}{1+\alpha\beta} + \alpha\log(\frac{\alpha\beta}{1+\alpha\beta})\right] + \alpha(1+\alpha\beta)\log(k).$$

We can keep iterating to find the solution, but there is a faster way. Note that both V_1 and V_2 are of the form: $a + b\log k$, for some constants a and b. Thus, T maps the family of functions of this form into itself and it is straightforward to show that T is also a contraction. Therefore, the contraction mapping theorem tells us that this

Bellman equation has a unique solution, which is itself log-linear in k.[12] So let us denote the unique fixed point as $V^*(k) = a + b \log k$, where a and b are coefficients that need to be determined. Let us then use this conjecture in (13.3.1) to get:

$$V^* = a + b \log k = \max_{k'} \{\log (Ak^\alpha - k') + \beta (a + b \log k')\} = TV^*.$$

The first order condition with respect to k' of this problem is:

$$\frac{1}{Ak^\alpha - k'} = \frac{\beta b}{k'} \Rightarrow k' = \frac{\beta a}{1 + \beta b} Ak^\alpha.$$

Let $LHS = a + b \log k$. Plug in the expression for k' into the RHS:

$$RHS = \log \left(Ak^\alpha - \frac{\beta b}{1 + \beta b} Ak^\alpha \right) + \beta \left(a + b \log \left(\frac{\beta b}{1 + \beta b} Ak^\alpha \right) \right).$$

Imposing the condition that $LHS \equiv RHS$ for all k, we can solve for a and b :

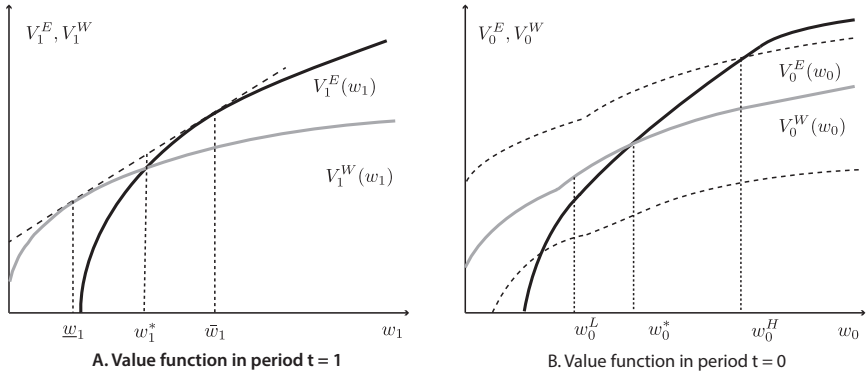
$$\begin{aligned} a &= \frac{1}{1-\beta} \frac{1}{1-\alpha\beta} \left[\begin{array}{c} \log A + (1 - \alpha\beta) \log (1 - \alpha\beta) \\ + \alpha\beta \log \alpha\beta \end{array} \right] \\ b &= \frac{\alpha}{1 - \alpha\beta} \end{aligned}$$

We have solved the model! Although this was a very special example—analytical solutions are hard to come by—some aspects of the approach used here are informative for the contraction mapping theorem more generally using computational methods. As long as the true value function is "well-behaved" (smooth, etc), we can choose a sufficiently flexible functional form that has a finite (ideally small) number of parameters. Then we can apply the same logic as above and solve for the unknown coefficients (sometimes called the "method of undetermined coefficients"), which then gives us the complete solution. Many solution methods rely on various versions of this general idea (perturbation methods, collocation methods, parametrized expectations). ■

Example 13.14. Non-strictly concave value function. This can arise more often than one might initially think. One class of models that give rise to this outcome is when the individual faces a discrete choice (so non-convex choice set), where each choice on their own deliver a strictly concave value function, but then the individual is allowed to randomize between the two choices. Without randomization, the time-zero value function would be the upper envelope of the two concave value functions, which may well be convex, as shown in Figure 13.3.1. Randomization, or lotteries, are used to convexify the choice set and convert the discrete choice set into a continuous one, leading to the disappearance of the convex value function.

[12] Of course, it needs to be shown that this class forms a complete metric space too. This is not straightforward, since logarithmic function is unbounded. See Stokey et al. (1989) for details.

FIGURE 13.3.1 – Discrete Choice, Non-Convex Value Function, and Convexification via Lotteries



This idea was first pursued in a pioneering paper by Rogerson (1988) and has been employed as an effective "trick" to convexify problems (rather than an economically meaningful mechanism) that would otherwise lead to non-concave value functions (among others, see, Hansen (1985), Clementi and Hopenhayn (2006), and Paulson et al. (2006)).

A particularly interesting example of this has been studied by Hopenhayn and Vereshchagina (2009) in the context of risk taking by entrepreneurs. This paper goes beyond lotteries as a trick and considers a framework in which lotteries naturally arise as an economically plausible entrepreneurial project risk choice. In its simplest form, they consider a two period model in which agents choose one of two occupations in period—being a worker or an entrepreneur. Conditional on each choice the value function is strictly concave under the assumption that the period utility function is strictly concave.

13.4 The Euler Equation

The solution(s) to the sequential decision problems of the kind that are typically studied in economics problems can often be characterized as the solution to a second-order (stochastic) difference equation, called Euler equations.[13]

[13]The class of problems studied here grew out of the famous brachistochrone curve problem first posed by Johann Bernoulli and studied by many of the great mathematicians of his time. Swiss mathematician Leonhard Euler made extensive contributions to this field, including refining the approach that uses the second-order difference (or differential) equation and giving the field its name "calculus of variations," with his treatise of the same title. The equation is also sometimes referred to as Euler-Lagrange equation.

To derive the Euler equation, start with the Bellman equation:

$$\begin{aligned} V(x,z) &= \max_{y\in\Gamma(x,z)} \{U(y,x,z) + \beta\mathbb{E}\left(V(x',z')|z\right)\} \\ \text{s.t. } x' &= H(y,x,z) \end{aligned}$$

An optimal policy of the form: $y = g(x,z)$ must satisfy the first-order optimality condition (FOC):

$$U_y(y,x,z) + \beta\mathbb{E}\left(V_x\left(x',z'\right)|z\right) H_y(y,x,z) \leq 0, \tag{13.4.1}$$

with equality if the solution is interior.[14] The envelope condition (Benveniste and Scheinkman (1979)) (with $y = g(x,z)$) is:

$$\begin{aligned} V_x(x,z) = U_y(y,x,z)g_x(x,z) + U_x(y,x,z) + \\ + \beta\mathbb{E}\left(V'\left(x',z\right)|z\right) H_x(y,x,z) + \beta\mathbb{E}\left(V_x\left(x',z\right)|z\right) \partial H_y(y,x,z)g_x(x,z). \end{aligned} \tag{13.4.2}$$

From the FOC, assuming an interior solution, it is easily seen that the first and last terms in (13.4.2) sum up to zero, leaving:

$$V_x(x,z) = U_x(y,x,z) + \beta\mathbb{E}\left(V_x\left(x',z\right)|z\right) H_x(y,x,z). \tag{13.4.3}$$

In some problems it is possible to define the state and choice variables such that the transition equation does not depend on current period state: $x' = H(y,z)$.[15] We have seen an example of this in the income fluctuations problem presented above, where choosing current assets to be the state variable and next period's assets as the choice variable delivered this result. In this case, the second term in (13.4.3) vanishes, since $H_x(y,z) \equiv 0$, leaving:

$$V_x(x,z) = U_x(y,x,z)|_{y=g(x,z)} \tag{13.4.4}$$

The intuition for the envelope condition is that because the objective function has already been maximized with respect to the choice variable, y, the indirect effect of y on the objective from that optimal point is zero. The envelope condition typically requires convexity conditions and differentiability of the value function, although Milgrom and Segal (2002) show that it extends to arbitrary choice sets as long as V is differentiable at point x.[16]

[14] H_y would stay inside the expectations operator if the transition function depends on an variable that is not in time t information set.

[15] In fact, Stokey et al. (1989) and Acemoglu (2009) adopt this notation throughout, whereas Bertsekas (2001) uses the more general notation defined above.

[16] They have a very useful simple example in the intro of their paper. Check it out. The intuition is useful to discuss here.

Substituting (13.4.4) into the first order condition (13.4.1), we obtain the Euler equation:

$$\begin{aligned} 0 =& U_y(y,x,z) + \beta \mathbb{E}\left(U_x(y',x',z')\right) H_y(y,z) \\ =& U_y(g(x),x) + \beta \mathbb{E}\left(U_x(g(x',z'),x',z')|z\right) H_y(g(x,z),z). \end{aligned}$$

13.4.1 Variational Approach to Deriving Euler Equations

There is an alternative way to derive the Euler equation that works on the sequence problem and relies on the definition of the optimum sequence that solves that problem.[17] Although working on the sequence problem might seem antithetical to the idea of recursive methods on which this book builds, this approach can often have important advantages, so it is essential to master it. I first show how the derivation works and then discuss when it is advantageous to use this approach.

Consider the sequence problem (13.2.1), specialized to the case of a consumption-savings problem here for simplicity. We have:

$$\begin{aligned} \tilde{V}(a_0,z_0) = \max_{\{\tilde{x}(z^t)\}_{t=0}^{\infty}} & \mathbb{E}_0 \left[\sum_{t=0}^{\infty} \beta^t U(c(z^t),z_t)\right] \\ \text{s.t.} \quad & c(z^t) + a(z^t) = (1+r)a(z^{t+1}) + y(z^t) \qquad \forall t \geq 0, \end{aligned} \tag{13.4.5}$$

and z_0 and $a_0 = a(z^0)$ given. Let $c^*(z^t)$ denote the optimal consumption sequence, i.e., the one that attains the value function $\tilde{V}(a_0,z_0)$. Construct an alternative sequence that reduces consumption in period t (after all possible histories up until that date) by an amount $\varepsilon > 0$, and invests this extra savings in the risk-free asset, which yields $(1+r)\varepsilon$ in $t+1$, which is then consumed at that date. Therefore, the new sequence is

$$\tilde{c} \equiv \{c^*(z^0), z^*(z^1), ..., c^*(z^t) - \varepsilon, c^*(z^{t+1}) + (1+r)\varepsilon, c^*(z^{t+2}), ...\}.$$

Notice that this alternative sequence is also budget-feasible, since it only affects the constraints in periods t and $t+1$ and in a way that offsets each other. Let V^{alt} denote the lifetime utility from this alternative policy. This perturbed sequence will change the lifetime value of the individual by

$$V^{\text{alt}} - \tilde{V}(a_0,z_0) = [U(c^*(z^t) - \varepsilon) - U(c^*(z^t))] + \beta[U(c^*(z^{t+1}) + (1+r)\varepsilon) - U(c^*(z^{t+1}))].$$

Because c^* is the optimal sequence by assumption, this increase cannot be positive, therefore: $V^{\text{alt}} - \tilde{V}(a_0,z_0) \leq 0$. Since $\varepsilon > 0$ we can divide through both sides of the

[17] In fact, this is how the great mathematicians of the 17th century originally approached calculus of variations problems.

inequality with it, and take the limit as $\varepsilon \to 0$, which yields:

$$-U'(c^*(z^t)) + (1+r)\beta U'(c^*(z^{t+1})) \leq 0,$$

with equality when the sequence $\{c^*\}$ is in the interior of the choice set in t and $t+1$. Rearranging it for an interior solution yields:

$$1 = (1+r)\beta \frac{U'(c^*(z^{t+1}))}{U'(c^*(z^t))}.$$

Using the same steps, it can be easily shown that for an asset with a stochastic return the usual Euler equation is obtained:[18]

$$1 = \mathbb{E}\left[\beta \frac{U'(c^*(z^{t+1}))}{U'(c^*(z^t))}(1+\tilde{r}(z^{t+1}))\right].$$

This approach makes several issues clear. First, we derived the Euler equation by using the argument that $\{c^*\}$ cannot be the optimal sequence unless it satisfies the resulting Euler equation. This makes clear that the Euler equation is a necessary condition for optimality but may not be sufficient. Because even though the particular one-step deviation we introduced may not yield utility improvements perhaps others can. Or even though small perturbations (e.g., a small ε) may not increase utility there is no guarantee that large deviations would not improve utility. This gives us a hint that for this not to be the case, the utility function and the choice set must possess certain properties. Indeed, this is the case. If the choice set is convex and the utility function is concave then and only then the Euler equation will also be sufficient for optimality.

Second, in complicated problems, this approach can often allow one to derive the Euler equation in a more intuitive and easier way. For example, Mirleesian taxation problems, which usually have complex constraint sets, are typically more amenable to the variational approach. Thus, unsurprisingly, many of the proofs in this literature (among others, Rogerson (1985), Golosov et al. (2003), Werning (2014)) use variational arguments to derive, what is called, the *inverse* Euler equation that captures the key trade-offs in that framework. Similarly, problems with time inconsistency (due to non-geometric discounting in preferences or government commitment problems) often yield, what is called, generalized Euler equations, which are often derived more easily using this approach. Third, this sequential for-

[18] A more general derivation is the following. Consider the same problem but allow the choice set to be more general, so that $y \in \Gamma(x)$. Now take the optimal sequence $\{x_t^*\}$ and attempt to vary only one element to improve the objective value. If the period return function is time separable then only utility in the current period and the next will be affected. So $\max_y \left[U(x_t^*, y) + \beta U(y, x_{t+2}^*)\right]$ subject to $y \in \Gamma(x_t^*)$ and $x_{t+2}^* \in \Gamma(y)$. Assuming that the original sequence is in the interior of the choice set of all t, the first order conditions are $U_y(x_t^*, x_{t+1}^*) + \beta U_x(x_{t+1}^*, x_{t+2}^*) = 0$. Since this variation can be done for every t, we get a second-order difference equation of this kind for every t.

mat of the Euler equation will come in handy in one type of algorithm, where we shall track the history of shocks. See Chapter 20.10.

13.5 Fancier Euler Equations

13.5.1 Ben-Porath/Irreversible Investment

Consider the problem of an individual who divides one unit of time each period between accumulating human capital, h, and working in the market...

$$\begin{aligned} & \max_{\{i_t\}_{t=1}^{T}} \left[\sum_{t=1}^{T} \delta^{t-1} R h_t (1-i_t)\right] \\ & s.t \\ h_{t+1} &= h_t + A\left(h_t i_t\right)^{\alpha}, \quad \text{with } h_0 > 0 \text{ given} \\ i_t &\in [0,1] \end{aligned}$$

Defining the newly produced human capital as $Q = A(hi)^{\alpha}$ and denoting the opportunity cost of investment as $C(Q_s) \equiv h_s i_s = (Q_s/A)^{1/\alpha}$, we can rewrite the dynamic programming problem in this way (that turns out to be more convenient to work with):

$$V_s\left(h_s\right) = \max_{Q_s \in [0, A h_s^{\alpha}]} \left[R\left(h_s - C(Q_s)\right) + \delta V_{s+1}\left(h_s + Q_s\right)\right]$$

Assuming interior solution, the FOC:

$$RC'(Q_s) = \delta V'_{s+1}\left(h_s + Q_s\right)$$

Envelope condition

$$V'_s\left(h_s\right) = R + \delta V'_{s+1}\left(h_s + Q_s\right).$$

Notice that the envelope condition did not allow us to eliminate the derivative of the value function as it did before. Instead the derivative of today's value function turns out to depend on the derivative of tomorrow's value function. But this is still useful if we observe that such an equation holds in every period, so we can substitute out V_{s+1} on the right hand sides as a function of V_{s+2} :

$$V'_s\left(h_s\right) = R + \delta\left[R + \delta V'_{s+2}\left(h_{s+2}\right)\right]$$

and at the terminal date we have $V_{T+1} \equiv 0$, which allows us to obtain and expression without value functions in it. Here is the first order condition:

$$RC'(Q_s) = \delta \left\{ R + \delta R + ... + \delta^{S-s-1} R \right\} \tag{13.5.1}$$

Notice an interesting feature of this problem: the optimal choice does not depend on any endogenous state variable, that is Q_s is not a function of h_s; it only depends on s. This suggests that the value function formulation above included a redundant state variable. This result—which is called "neutrality"—is due to the fact that the Ben-Porath production function adopted here has the same exponent α for both i_t and h_t. Relaxing this assumption would make h_s a proper state variable again. But this is to keep in mind that what state variables a dynamic programming problem has takes some consideration and understanding the economic problem at a deep level. Later we shall see some examples in which we can reduce the number of state variables in even more subtle ways.

13.5.2 Time Non-Separable Preferences

Here is an example of an Euler equation for preferences that exhibit habit formation, whose Euler equation is straightforwardly obtained using this approach. Consider the consumption and saving decision of a household given by,

$$\tilde{V}(a_0, z_0) = \max_{\{c_t, a_{t+1}\}_{t=0}^{\infty}} \mathbb{E}_0 \left[\sum_{t=0}^{\infty} \beta^t U(c(z^{t+1}) - \theta c(z^t)) \right], \tag{13.5.2}$$

s. to

$$c(z^t) + a(z^t) = (1 + r(z^{t+1}))a(z^{t+1}) + y(z^t), \qquad c_{-1} > 0 \quad \text{given} \tag{13.5.3}$$

in which z^t is the history of random realizations of the income shock $y(z^t)$. Therefore, the momentary utility in the current period depends on consumption spending both today and yesterday. Let $c^*(z^t)$ denote the optimal consumption sequence and consider as before reducing consumption in t for all states by an amount of $\epsilon > 0$ and investing this amount at gross rate of return $R(z^{t+1})$. Hence, this new sequence is

$$\tilde{c} = \{c^*(z^0), c^*(z^1), ..., c^*(z^t) - \epsilon, c^*(z^{t+1}) + \epsilon R(z^{t+1}), c^*(z^{t+1}), ...\}, \tag{13.5.4}$$

which in turn yields the following sequence of momentary utilities:

$$\{U_0^*, ..., U((c^*(z^t) - \epsilon) - \theta c(z^{t-1})), U((c^*(z^{t+1}) + \epsilon R(z^{t+1})) - \theta(c^*(z^t) - \epsilon)),$$
$$U(c^*(z^{t+2}) - \theta(c(z^{t+1}) + \epsilon R(z^{t+1})), U_{t+3}^*, ...\},$$

where U_t^* denotes the momentary utility under the optimal sequence. This alternative sequence is feasible, which is easy to verify. It also affects momentary utility in $t+2$ by modifying the habit stock in period $t+1$. This perturbed sequence changes lifetime utility by

$$\begin{aligned}\Delta V = [U((c^*(z^t) - \epsilon) - \theta c^*(z^{t-1})) - U(c^*(z^t) - \theta c^*(z^{t-1}))] + \\ \beta \mathbb{E}_t[U(((c^*(z^{t+1}) + \epsilon R(z^{t+1})) - \theta(c(z^t) - \epsilon)) - U(c^*(z^{t+1}) - \theta c(z^t))] + \\ \beta^2 \mathbb{E}_t[U(c^*(z^{t+2}) - \theta(c^*(z^{t+1}) + \epsilon R(z^{t+1}))) - U(c^*(z^{t+2}) - \theta c^*(z^{t+1}))] \leq 0,\end{aligned}$$

which cannot be positive since c^* is the optimal sequence and the new sequence is budget feasible. Since $\epsilon > 0$, we can divide in both side of this equation and take the limit when $\epsilon \to 0$ and assume that U is a power utility function, which yields

$$1 = \mathbb{E}_t \left[\beta R(z^{t+1}) \times \frac{(c(z^{t+1}) - \theta c(z^t))^{-\sigma} - \beta\theta(c(z^{t+2}) - \theta c(z^{t+1}))^{-\sigma}}{(c(z^t) - \theta c(z^{t-1}))^{-\sigma} - \beta\theta(c(z^{t+1}) - \theta c(z^t))^{\sigma}} \right] \quad (13.5.5)$$

at an interior optimum, which is the Euler Equation.

The last two examples relied a sequential formulation, leading to Euler equations where the choice function is expressed as a function of the entire history of shocks. In the next example, we use a recursive formulation with explicit state variables.

13.5.3 Time Inconsistency Problems and a Generalized Euler Equation

Consider the consumption-savings problem facing an individual with preferences that display quasi-geometric discounting defined as follows.[19] Let V_t denote the lifetime utility starting in period t :

$$\begin{aligned}V_0 &= U_0 + \beta\left(\delta U_1 + \delta^2 U_2 + \delta^3 U_3 ...\right) \\ V_1 &= U_1 + \beta\left(\delta U_2 + \delta^2 U_3 + ...\right) \\ V_2 &= U_2 + \beta(\delta U_3 + ...)\end{aligned} \quad (13.5.6)$$

Notice that there are two parameters governing patience: β determines the discounting between today and tomorrow, i.e., it is a measure of short-term impatience, whereas δ is what the individual thinks he will use to discount future dates starting tomorrow. When $\beta = 1$, these preferences reduce to the usual geometric discounting, leading to a recursive problem that is time consistent. However, when $\beta < 1$, the optimal plan made by the current self will not typically be viewed as optimal by the individual tomorrow. To refer to this time inconsistency we refer to the individual at each point in time as a separate self. Of course an open question is

[19] See Krusell et al. (2002).

whether the individual today will take into account this time inconsistency and in particular the fact that he cannot control the choices that the self tomorrow makes.

A reasonable assumption is that the individual is rational and then an interesting question is whether there is a policy that is time consistent in this environment—that is tomorrow's self chooses the same policy as the one chosen by the self today. Before going further, it is useful to note that here we discuss an application to preferences, largely motivated by experimental evidence showing that time discounting resembles a hyperbolic function (here approximated by quasi-geometric form), a very similar kind of time inconsistency arises in government policy problems with broad applications.[20]

Now let us solve the planner's problem in this economy. The problem can be written recursively once we properly distinguish between the current self's value function $J(k)$ and what he perceives as the value function of the self tomorrow, $V(k)$:

$$J(k) = \max_{k'} \left[U(f(k) - k') + \beta\delta V(k')\right] \tag{13.5.7}$$

where, denoting the optimal policy $k' = h(k)$ we have

$$V(k) = \left[U(f(k) - h(k)) + \delta V(h(k))\right] \tag{13.5.8}$$

Now the FOC is:

$$U'(f(k) - h(k)) = \beta\delta V'(h(k)). \tag{13.5.9}$$

Differentiate future self's value function using (13.5.8):

$$V'(k) = U'(f(k) - h(k))(f'(k) - h'(k)) + \delta V'(h(k))h'(k). \tag{13.5.10}$$

Substitute $V'(h(k))$ from (13.5.9) into (13.5.10), then lead the resulting equation by one period, and substitute the new expression for $V'(h(k))$ back into the FOC (13.5.9), which yields

$$1 = \beta\delta \frac{U'(f(h(k)) - h(h(k)))f'(h(k))}{U'(f(k) - h(k))} \left(f'(h(k)) + \left(\frac{1}{\beta} - 1\right) h'(h(k))\right).$$

This is a functional equation in $h(k)$ and can be solved using one of the methods described in the coming chapters. Alternatively, substitute back $k_{t+1} = h(k_t)$ to get

[20] e.g., Klein et al. (2008).

$$1 = \beta\delta \frac{U'(f(k_{t+1}) - k_{t+2}) f'(k_{t+1})}{U'(f(k_t) - k_{t+1})} \left(f'(k_{t+1}) + \left(\frac{1}{\beta} - 1 \right) h'(k_{t+1}) \right),$$

for $t = 0, 1, 2, ...$ Notice that when $\beta = 1$, this reduces to the standard Euler equation derived above (give equation number). When $\beta < 1$, the return to savings perceived by the current self is higher than the marginal product of capital by a (positive) amount $\left(\beta^{-1} - 1\right) h'(k_{t+1})$, which captures the fact that current self, while being impatient about tomorrow, is more patient than tomorrow's self about the future starting tomorrow. Therefore, he wants to save an additional amount whose intensity increases with how much of that additional wealth tomorrow's self will save (given by $h'(k_{t+1})$).

Exercise 13.15. Consider a firm's optimal investment problem in the presence of convex capital adjustment costs. Specifically, the firm's managers choose investment policy $\{I_t\}$ and labor demand $\{L_t\}$, to maximize the value of the firm, which equals the value of the future dividend stream generated by the firm, $\{D_{t+j}\}_{j=1}^{\infty}$, discounted by the marginal rate of substitution process of firm owners, $\left\{\beta^j \Lambda_{t,t+j}\right\}_{j=1}^{\infty}$. The firm's dividend is $D_t = Z_t K_t^{\theta} L_t^{1-\theta} - W_t L_t - I_t$, where $\log Z_t$ follows an AR(1) process, K_t and L_t are capital and labor employed by the firm, W_t is the wage rate

$$P_t^s = \max_{\{I_{t+j}, L_{t+j}\}} \mathbb{E}_t \left[\sum_{j=1}^{\infty} \beta^j \Lambda_{t,t+j} D_{t+j} \right] \tag{13.5.11}$$

subject to the law of motion for capital, which features "adjustment costs" in investment:

$$K_{t+1} = (1 - \delta) K_t + \Phi \left(\frac{I_t}{K_t} \right) K_t. \tag{13.5.12}$$

Let $\Phi = a_1 \left(I_t / K_t\right)^{1-1/\xi} + a_2$ and choose a_1 and a_2 such that the balanced growth path is invariant to ξ. Derive the Euler equation that characterizes the optimal policy of this firm.

13.6 Dynamic Programming: An Alternative Formulation

While the formulation adopted so far is the most common way to write down a dynamic programming problem in economic applications, it is not the only possible one. In particular, in the formulation of Section 13.2, we had $x_{t+1} = G(x_t, y_{t,} z_t)$, implying that today's state (x_t, z_t) together with today's choice y_t determined tomorrow's endogenous state variable, x_{t+1}, which was an argument into the value

function on the right hand side of the Bellman equation. Consequently, the first order condition for y_t (equation (13.4.1)) involved the derivative of tomorrow's value function—a function that is not known until the problem is solved.

An alternative formulation proceeds as follows. First, we assume that the state space is discrete: $x \in \{\overline{x}_0, \overline{x}_1, \overline{x}_2, ..., \overline{x}_N\}$. In general, this is not a palatable assumption because in the original formulation it would have required the choice variable to also be discrete (so that we can move from today's grid to tomorrow's grid and not land off-grid). However, we will not be making this assumption. Instead, we assume that the choice variable determines the transition probabilities between today's and tomorrow's states: $\pi_{ij}(y_t) \equiv \Pr\{x_{t+1} = \overline{x}_j | x_t = \overline{x}_i, y_t)$. This specification is more general than it may first appear. For example, if y_t is savings, we can allow it to be a continuous choice, and specify the function $\pi_{ij}(y_t)$ judiciously. For example we could assume that higher savings today results in a probability distribution over tomorrow's wealth (x_{t+1}) that first order stochastically dominates the one with lower savings.[21] Similarly, one can model deterministic transitions with a matrix properly placed zeroes and ones. We can allow for constraints by adding virtual states with large negative payoffs. And so on.

With this formulation, we can now write the Bellman equation analogous to (13.2.2) as:

$$V(x_i) = \max_{y \in \Gamma(x_i)} \left[U(x_i, y) + \beta \sum_{n=1}^{N} \pi_{ij}(y_t) V(x_j) \right]. \tag{13.6.1}$$

Notice several differences of this formulation from the previous one. First, z_t is no longer a separate state variable, because we assume that all the effects of Markov uncertainty is subsumed into the transition matrix, π. Second, and more importantly, now today's choice variable, y, no longer enters the unknown value function V as an argument. Instead, it is also subsumed into the transition matrix. To see why this matters, differentiate the right hand side with respect to y to obtain the first order condition:

$$U_2(x_i, y) + \beta \sum_{n=1}^{N} \pi'_{ij}(y_t) V(x_j) = 0.$$

We have quite a bit more to say on this formulation in the coming chapters. Some especially useful computational tricks have been developed in this framework and they extend nicely to frameworks with continuous state. More on this later.

[21] $F\{x_{t+1} \leq \overline{x}_j | x_t = \overline{x}_i, y_a\} \leq F\{x_{t+1} \leq \overline{x}_j | x_t = \overline{x}_i, y_b\}$ for all $y_a > y_b$, (where F is the cumulative distribution function corresponding to π_{ij}) with strict inequality for some a, b.

13.7 Further Exercises (From Sergio Salgado)

This is the first batch of examples in Dynamic Programing. The first section corresponds to examples from Richard Bellman's original book while the second part has problems coming from Eric Denardo's book. In the third section I also adapt some questions from past midterm and final exams and some prelims.

13.7.1 Richard Bellman's Dynamic Programing

➢ *Ch1. Problem 6.* Consider the problem of maximizing the function,

$$F(x_1, x_2, ..., x_N) = \sum_{i=1}^{N} \phi(x_i), \tag{13.7.1}$$

subject to,

$$x_i \geq 0, \sum_{i=1}^{N} x_i = c. \tag{13.7.2}$$

Show that the maximum is $\phi(c)$ under the assumption that $\phi(x)$ is convex.

➢ *Ch1. Problem 7.* Consider the previous problem and assume that $\phi(x)$ is a monotonically increasing function which is strictly concave. Show that the solution of the corresponding functional equation,

$$f_n(c) = \max_{0 \leq y \leq c} [\phi(y) + f_{N-1}(c-y)], N \geq 2, \tag{13.7.3}$$

$$f_1(c) = \phi(c), \tag{13.7.4}$$

has the form,

$$y_N = 0, 0 \leq c \leq c_N, c_N = z_N, c > c_N, \tag{13.7.5}$$

where z_N is the unique solution of

$$\phi'(y) = f_{N-1}(c-y), \tag{13.7.6}$$

for $N \geq 2$. Show how to determine the sequence of c_N.

➢ *Ch1. Example and Problem 17.* Assume that we have a quantity x that must be divided into two nonnegative parts, y and $x - y$ obtaining from the first quantity y a return of $g(y)$ and from the second a return of $h(x-y)$. In a one-stage problem, we wish to find a maximum of the following function,

$$R_1(x, y) = g(y) + h(x-y),$$

for all $y \in [0, x]$. Assume that g and h are continuous functions of x. Consider now a two stage process. Suppose that as a price of obtaining the return $g(y)$ the original quantity y is reduced to ay where $a \in [0, 1)$ and similarly $(x - y)$ it is reduce to $b(x - y)$. With the remaining part, the process is repeated. Then the total return for the two-stage process is,

$$R_2(x, y, y_1) = g(y) + h(x - y) + g(y_1) + h(x_1 - y_1),$$

where y_1 and x_1 are the decision in the second stage and the total resources for the second stage respectively.

A Define the total return for the N-stage process.

B Define f_N as the maximum return obtained from an N stage process starting with an initial quantity x, for $N = 1, 2....$ and $x \geq 0$. Define the functional equation that solves the maximization problem for N stages.

C Assume that it is not required to use all the resources available at each stage. Show that the functional equation obtained in this way has the form,

$$f(x) = \max_{y_1+y_2 \leq x} [g(y_1) + h(y_2) + f(ay_1 + by_2 + x - y_1 - y_2)]. \quad (13.7.7)$$

Does this equation have a solution if $g(x)$ and $h(x)$ are both concave functions of x? Does it have a solution if they are both convex? Under what conditions upon $g(x)$ and $h(x)$ does it have a solution?

➢ *Ch1. Problem 19.* Consider the process of the previous problem under the assumption that additional resources are added at each stage, either externally of from the conversion of all or part of the return $g(y)+h(x-y)$ into resources. Obtain the corresponding recurrence relations.

➢ *Ch2. Problem 11.* Assume that an individuals owns two gold mines, A and B, the first of which possesses an amount of gold of x and the second an amount of gold of y. In addition we have a single machine with the property that if used to mine gold in A, there is a probability p_1 that it will mine a fraction r_1 of gold and will remain in working order, and a probability $(1 - p_1)$ that will mine no gold and be damaged beyond repair. Similarly, the mine B has similar probabilities associated. Define the functional equation $f_N(x, y)$ as the expected amount of gold mined before the machine is damaged when A has x and B has y and a optimal policy which can last at most N period is employed. Write the the problem of deciding in which mine to use the machine in a recursive form. Now assume that there is a probability p_1 of obtaining r_1x and continuing, a probability p_2 of obtaining nothing and continuing and a probability p_3 of obtaining nothing and terminating, if A is chosen, with

$p_1 + p_2 + p_3 = 1$, with similar probabilities q_1, q_2 and q_3 if B is chosen. Show that the corresponding functional equation is,

$$f(x) = \begin{bmatrix} A: & p_1[r_1 x + f((1-r_1 x, y))] + p_2 f(x,y) \\ B: & q_1[s_1 x + f((1-s_1 x, y))] + q_2 f(x,y) \end{bmatrix}, \tag{13.7.8}$$

and this can be written in the simpler form,

$$f(x) = \begin{bmatrix} A: & \frac{p_1}{1-p_1}[r_1 x + f((1-r_1)x, y)] \\ B: & \frac{q_1}{1-q_1}[s_1 x + f((1-s_1)x, y)] \end{bmatrix}. \tag{13.7.9}$$

➢ *Ch4. Problem 35.* Assume that a gambler plays to maximize the expected value of the logarithm of his capital after N stages. Let $f_N(x)$ denote the expected value obtained using an optimal policy. Show that

$$f_{N+1}(x) = \max_{0 \le y \le x} [p f_N(x+y) + q f_N(x-y)], N = 1, 2, ..., \tag{13.7.10}$$

assuming that there are equal odds, with,

$$f_1(x) = \max_{0 \le y \le x} [p \log(x+y) + q \log(x-y)]. \tag{13.7.11}$$

Show that the optimal value function is given by $f_N(x) = \log(x) + Nk$ where $k = \max_{0 \le r \le 1}[p \log(1+r) + q \log(1-r)]$, and hence that there is a number r_0 such that the optimal policy at each stage is determined by the relation $y = r_0 x$.

➢ *Ch4. Problem 1.* Determine the structure of the optimal policies associated with the functional equation,

$$f(p) = \max_q [R(p,q) + f(T(p,q))], \tag{13.7.12}$$

under the assumption that $R(p,q)$ and $T(p,q)$ are convex functions of p and q and that $R(p,q)$ and $T(p,q)$ are monotone increasing functions of p for each q.

13.7.2 Eric V. Denardo's Dynamic Programing

➢ *Ch1. Example 1.* Trout Rader is planning on poaching in the national preserve, which contains K lakes. A careful planner, he has assembled the following data,

y_i = the number of trout he expects to catch in an hour's fishing at lake i,

$t_{i,j}$ = the travel time (an integer) in hours from lake i to lake j.

Travel times satisfy the triangle inequality $t_{ik} \leq t_{ij} + t_{jk}$. Trout recognizes that there is some risk of capture by the game warden. To minimize this risk, he has decided to fish only for one hour in each lake he visit. This offers him some protection against the possibility that a passerby finds the game wared and brings him back. Trout allows himself the option of returning to a lake after having finished in another, as the game warden will likely have left. Trout has just entered the preserve the lake 1. It is 12 hours until suppertime.

A What constitutes the state of Trout's problem?

B Define the functional equation that Trout solves.

C Write a functional equation for which the state is interpreted as arriving at the lake i and the number of hours left n.

D Suppose that Trout's wife is camping at lake 1 and that she wants him return there so that his catch may be poached for supper. Alter the first functional equation to reflect this constraint.

➢ *Ch1. Example 2.* Interpret the integer K as the volume of a knapsack. Each unit of commodity n occupies a volume c_n (a positive integer) in the knapsack and yields a profit p_n. The optimization problem is to find the most profitable way to pack the sack. The problem can be writes as

$$\max\{p_0x_0 + ... + p_Nx_N\}, \tag{13.7.13}$$

$$\text{subject to,}$$

$$c_0x_0 + ... + c_Nx_N = K; x_n \geq 0, \tag{13.7.14}$$

$$n = 0, ..., N. \tag{13.7.15}$$

A What is a meaningful state variable in this problem,

B define the function form that solves this problem .

➢ *Ch6. Theorem 1.* Define $R_i^k(n)$ as the reward earned during period n if state (n, i) is occupied and if decision k is selected. Then, one can define $v^k(n, i)$ as the expectation of the total income earned in periods n trough N if state (n, i) is occupied and if policy k is used. If P_{ij} is the transition probability between the exogenous state i to j, then we can write $v^k(n, i)$ as,

$$v^k(n, i) = R_i^k(n) + \sum_j P_{i,j}^k v^k(n + 1, j), \tag{13.7.16}$$

Define $f(n, i)$ the maximum total expected earnings obtainable from periods n and N if the state in period n is i. Then we can write $f(n, i) = \max_k v^k(n, i)$. Assume that the state space has finitely many elements. Show that an maximal value for (13.7.16) exists and is exactly equal to $f(n, i)$. Show also that

the optimal policy exists and solves,

$$v(n,i) = \left\{ \begin{array}{ll} \max_k R_i^k(n) + \sum_j P_{ij}^k v(n+1,j) & n < N \\ \max_k R_i^k(N) & n = N \end{array} \right]. \tag{13.7.17}$$

13.7.3 Questions from Prelims and Exams

➢ *Macro Spring 2008 - Question I.1.* Define the problem

$$V(h,k) = \sup_{c_t,x_t,x_{ht},k_t,h_t,n_t} \sum_{t=0}^{\infty} \beta^t \log(c_t), \tag{13.7.18}$$

subject to

$$c_t + x_{ht} + x_{kt} \leq A k_t^{\alpha} (n_t h_t)^{1-\alpha}, \tag{13.7.19}$$

$$k_{t+1} = k_t(1-\delta_k) + x_{kt}, \tag{13.7.20}$$

$$h_{t+1} = h_t(1-\delta_h) + x_{ht}, \tag{13.7.21}$$

$$0 \leq n_t \leq 1, \forall t, \tag{13.7.22}$$

where c_t is consumption, n_t is labor supply, x_{ht} and x_{kt} are investment in human and physical capital and δ_h and δ_k are the depreciation rates. Assume an initial value of $(h,k) = (h_0, k_0)$ given. Consider some sequence $\{h_t, k_t\}_{t=1}^{\infty}$ such that $(h_t, k_t) \in \mathbb{R}_{++}^2$ for all t. Show that there exists a function $B : \mathbb{R}_+ \to \mathbb{R}_+$ such that $V(\lambda h, \lambda k) = B(\lambda) + V(h,k)$ for all $\lambda > 0$.

➢ *Macro Spring 2008 - Question I.2.* Consider a household that lives for a finite number of periods, T. Wealth, W_t, at the beginning of the period is allocated towards current consumption, c_t, purchases of a risky asset x_t, and a risk-free asset, y_t, according,

$$c_t + x_t + y_t \leq W_t. \tag{13.7.23}$$

Then wealth evolves according to

$$W_{t+1} = R x_t + R_f y_t, \tag{13.7.24}$$

where R denotes the gross return to the risky asset and R_f denotes the gross return to the risk free-asset. Assume that R is *i.i.d.* over time and is drawn from a distribution $F(R)$. The household's preferences are given by $\sum_{t=0}^{T} \beta^t u(c_t)$ where β is the discount factor and $u(c_t) = c_t^{1-\sigma}/(1-\sigma)$. The household maximizes expected utility.

A Set up the household's problem as a dynamic program.

B Financial planners recommend that households' portfolios should become less risky as they age. Evaluate this recommendation. Hint: Guess at

the functional form of the value function. Derive the household's optimal asset allocation given this guess.

➢ *Macro Spring 2008 - Question II.1.* Here you are asked to show that the policy function of a standard growth model is increasing in the discount factor. Let T_β be the operator mapping weakly concave, bounded and continuous functions $f : X \to \mathbb{R}$ into itself: $T_\beta : C'(X) \to C'(X)$.

$$T_\beta(f)(k) = \max_{k \in \Gamma(k)} \{u(f(k) - k') + \beta f(k')\}. \tag{13.7.25}$$

Fix $\hat{\beta} > \beta$. Let $T_{\hat{\beta}}$ be the operator using $\hat{\beta}$.

A a) Argue that there is a unique fixed point for each operator.

Let $v^*(.,\beta)$ and $v^*(.,\hat{\beta})$ be the fixed points of the operators T_β and $T_{\hat{\beta}}$ respectively. Assume that both functions are differentiable. Define $g^*(.,\beta)$ and $g^*(.,\hat{\beta})$ as the corresponding optimal policy functions. Also let $v_k(.,\hat{\beta})$ be the k^{th} iteration using the operator $T_{\hat{\beta}}$ and $g_k(.,\hat{\beta})$ the corresponding policy function.

A b) What optimality condition is satisfied by $g^*(k;\beta)$.

B c) Now, use $v^*(.,\beta)$ as the initial guess for the operator $T_{\hat{\beta}}$. Show the optimality condition that $g_1(k,\hat{\beta})$ satisfies.

C d) Show that $g_1(k,\hat{\beta}) > g_1(k,\beta)$.

D e) Show that $v_1^{'}(k,\hat{\beta}) > v_1^{'*}(k,\beta)$ (Hint: think in the envelope theorem).

E f) Show that $g_n(k,\hat{\beta}) > g^*(k,\hat{\beta})$ for all n.

➢ *Macro Spring 2012 - Question I.1.* Consider an economy in which the representative consumer lives forever. There is a good in each period that can be consumed or saved as capital as well as labor. The consumer's utility function is $\sum_{t=0}^{\infty} \beta^t \log(c_t)$. Here $\beta \in (0,1)$. The consumer is endowed with 1 unit of labor in each period and with k_0 units of capital in period 0. Feasible allocations satisfy, $c_t + k_{t+1} = \theta k_t^\alpha l_t^{1-\alpha}$ with $\theta > 0$ and $\alpha \in (0,1)$.

A Formulate the problem of maximizing the representative consumer's utility subject to feasibility conditions as a dynamic programming problem. Write down the appropriate Bellman's equation.

B Guess that the value function has the form $a_0 + a_1 \log(k_t)$. Solve the dynamic programming problem.

➢ *Macro Spring 2009 IV.1-Spring 2013 II.1.* Consider the following Planner's Problem:

$$\max \sum_{t=0}^{\infty} \beta^t \frac{c_t^{1-\gamma}}{1-\gamma} \tag{13.7.26}$$

subject to,

$$c_t + x_{ht} + x_{kt} \leq A k_t^{\alpha} (z_t)^{1-\alpha}, \tag{13.7.27}$$

$$k_{t+1} \leq k_t(1-\delta_k) + x_{kt}, \tag{13.7.28}$$

$$h_{t+1} = h_t(1-\delta_h) + x_{ht}, \tag{13.7.29}$$

$$z_t \leq n_t h_t, 0 \leq l_t + n_t \leq 1, \forall t, \tag{13.7.30}$$

A Show the Functional Equation that solve this problem. Argue that the solution of the Sequential Problem and the Functional Equation coincide.

B Show that the Functional Equation of this problem is homogeneous of of degree $(1-\sigma)$ in the capital stocks, (k_0, h_0).

C Characterize the homogeneity properties of the optimal decision rules in the initial capital stock (k_0, h_0). Specifically if the initial conditions on the capital stock are $(\eta k_0, \eta h_0)$ instead of (k_0, h_0) where $\eta > 0$, how are the optimal paths of consumption, labor supply, investment, etc. affected? Prove your claims.

➢ *Final 2007 (Jones) P3.* Consider the following problem,

$$\max E_0 \left[\sum_{t=0}^{\infty} \beta^t \frac{c_t(s^t)^{1-\sigma}}{1-\sigma} \right] \tag{13.7.31}$$

subject to

$$c_t(s^t) + x_t(s^t) = F(k_t(s^{t-1}); s^t), \tag{13.7.32}$$

$$k_{t+1}(s^t) \leq x_t(s^t), \tag{13.7.33}$$

$$k_0(s^{-1}) = k_0 > 0 \text{given},$$

where $\sigma > 1$ and F is continuously differentiable, strictly concave and strictly increasing. Assume that $F(k; s^t) = A(s^t)k$ where $A(s^t) = As_t$. In the first parts assume that s^t follows a first order Markov process with the following properties,

$$s_t \in \{1-\bar{\epsilon}, 1+\bar{\epsilon}\},$$

$$Pr(s_{t+1} = 1-\bar{\epsilon} | s_t = 1-\bar{\epsilon}) = Pr(s_{t+1} = 1+\bar{\epsilon} | s_t = 1+\bar{\epsilon}) = 1 > q > \frac{1}{2}.$$

$$Pr(s_0 = 1-\bar{\epsilon}) = \frac{1}{2}.$$

A Write this problem as a Bellman equation.

B Guess that the value function $V(k,s)$ is of the form $k^{1-\sigma}D(s)$. Verify that this guess is true. Calculate $D(s)$. Your answer should only contain $\bar{\epsilon}, q, \beta, \sigma, A$.

C At each history s^t calculate the expected growth rate of the economy.

D Now assume $\bar{s}_t, \tilde{s}_t$ that are *i.i.d.* process and $E(\bar{s}_t) = E(\tilde{s}_t) = 1$. Assume also that $\tilde{s}_t$ is a mean preserving spread of $\bar{s}_t$.

➢ *(Midterm 2007 (Jones) P3)* Consider the following sequence problem

$$\max_{c_t, k_{t+1}} \sum_{t=0}^{\infty} u(c_t), \tag{13.7.34}$$

subject to,

$$c_t + k_{t+1} \leq F(k_t), \forall t \tag{13.7.35}$$

$$k_{t+1} \geq 0, \forall t, k_0 \text{ given.}$$

A Write the problem in canonical form,

B Assume $u(c) = -e^{-\alpha}$ and $F(k) = Ak$. Write the Bellman equation and solve for the value function. Hint: Guess that $v(k) = -Ce^{-\gamma k}$ and verify. Ignore the positive consumption constraint.

C Assume u and F are strictly concave and continuously differentiable. Assume that the solution to the right-hand-side of Bellman equation is interior. Show that policy function is increasing in capital.

➢ *(Macro Prelim QIII S2003, QI.1 S2005, QI.1 F2005, QI.4 F2006, QI.4 S2007)* Consider the decision problem of a single worker who has preferences given by $\sum_{t=0}^{\infty} \beta^t u(c_t)$ where c_t denotes consumption, β is a discount factor and u is strictly concave. At the beginning of each period the worker is either unemployed or employed. If unemployed, the worker receives a wage offer from a distribution $G(w)$ and if employed, gets a new offer from a distribution $F(w)$. These offers are constant for the duration of the jobs. These draws are i.i.d. Jobs disappear with probability δ at the end of each period. The worker cannot save or borrow and if unemployed receives a benefit b.

A Set up the decision problem as a dynamic program.

B Show that the solution of the dynamic program exists. In particular, show that the Blackwell's sufficient conditions hold.

➢ *Macro Spring 2008 QII.3* Consider an economy in which workers accumulate human capital while working. A worker currently working at a job produces a single nonstorable good according to h_t^{α} where h_t is human capital and

$0 < \alpha < 1$. Jobs disappear at the end of the period with probability δ. If the job does not disappear, the worker's human capital next period is $h_{t+1} = \gamma h_t$ where $\gamma > 1$. If the job disappears, the worker becomes unemployed. Unemployed workers receive one new job offer in each period. The offers are parameterized by a random variable z which is uniformly distributed between 0 and 1 and is i.i.d. over time. A worker who accepts a job at t has human capital $h_t = zh_{t-1}$. That is, part of the human capital $(1 - z)h_{t-1}$ disappears forever. Workers are risk-neutral and maximize $\sum_{t=0}^{\infty} \beta^t c_t$. Assume no borrowing or lending is possible.

A Set up the decision problem as a dynamic program.

B Show that the solution of the dynamic program exists. In particular, show that the Blackwell's sufficient conditions hold.

➢ *Macro Fall 2008 I.2* Consider the problem faced by each of a large number of inventors. An inventor can either choose to invent a new product at a cost of b or not. If the inventor chooses to invent, the quality of the invention is random. Let z denote the quality of an invention where z is drawn independently across inventors and time from a distribution $F(z)$. The quality of the invention is also the per period profits from the invention. Once an invention is produced, the inventor must manage the product if he wants to sell it. While managing, an inventor has no time to invent. With probability δ, an invention becomes worthless, and the inventor can go back to his first love, inventing new products. Assume investors are risk neutral and discount future profits.

A Set up a typical inventor's problem.

B Show that the solution of the dynamic program exists. In particular, show that the Blackwell's sufficient conditions hold.

C Suppose the costs b fall. What happens to the fraction of those engaged in invention in a stationary equilibrium?

Chapter 14

Dynamic Programming: Numerical Methods

MANY APPROACHES to solving a Bellman equation have their roots in the simple idea of "value function iteration" and the "guess and verify" methods. These are the very first steps typically one learns about for obtaining analytical solutions, but they are also practical and useful in numerical work. In general, the algorithms for dynamic programming can be broadly classified along two dimensions:

A Algorithms that work with the Bellman equation versus those that work through the Euler equation. Each approach has its strengths, although in more complex problems working with the Bellman equation can be more reliable.

B Algorithms that work with a class of flexible functional forms, such as polynomials, versus finite element methods.

To illustrate how each approach works, let us recall how the two methods work.

14.1 Value Function Iteration

Value function iteration (VFI) is the oldest and most basic approach to solving a dynamic programming problem. It was originally proposed by Richard Bellman in his seminal book (Bellman (1957)) and has been the workhorse of DP analysis since then. Its performance is guaranteed by the contraction mapping theorem (Theorem 13.3) stated in the previous chapter and is applicable in any problem where the conditions of that theorem are satisfied. In particular, we must be able

to establish that the Bellman operator for the specific problem we are dealing with is a contraction mapping over the space of appropriately specified functions, which itself must be a complete metric space. These conditions are rather mild and are satisfied by a broad class of problems you will encounter in economics. VFI is the primary approach to DP problems and arguably the most generally applicable one. It can be used to solve problems with non-convexities and non-differentiabilities (various binding constraints or discrete choice sets), where other methods can easily fail.

The solution algorithm is provided by the following result in the theorem:

$$TV^* = V^* = \lim_{N \to \infty} T^N V_0$$

for all $V_0 \in S$, where T^N denotes N repeated applications of T. That is, the unique fixed point is attained by repeated applications of the Bellman operator, starting from any point (i.e., function) that lies in the space of interest, S. This gives us enormous flexibility in choosing an initial guess and we are ensured to converge to the correct solution, *eventually*.

Thus, conceptually, VFI is a fairly straightforward algorithm. In an abstract form, the *standard VFI algorithm* can be written as follows.

Algorithm 6: Standard Value Function Iteration

Step 0. Set $n = 0$. Choose an initial guess $V_0 \in S$.

Step 1. Obtain V_{n+1} by applying the mapping: $V_{n+1} = TV_n$, which entails maximizing the right-hand side of the Bellman equation.

Step 2. Stop if convergence criterion i satisfied: $\|V_{n+1} - V_n\|_\infty < \texttt{toler}$. Otherwise, increase n and return to step 1.

Although the algorithm is conceptually simple, implementing it (and other algorithms like it) on a computer raises a host of practical issues. Before delving into the details of implementation, we first discuss some of the most important of these issues.

14.2 Four Practical Issues

Four issues in particular require special attention.

A How do we represent a general function—which maps infinitely-many points into infinitely-many points—on a computer?

B How do we pick a sensible initial guess, V_0? How do we determine a reasonable stopping rule for the algorithm ("toler")?

C Initialization: How do we determine the domains of various functions? This requires choosing appropriate bounds for state variables, choice variables as well as a way to represent the state space

D How do we improve the speed of the algorithm for a given desired accuracy?

We begin by discussing the first three of these practical issues in this section. The fourth one—accelerating VFI based algorithms—is a major topic on its own. Therefore, starting in Section 14.3, we devote the rest of this chapter to covering a variety of useful acceleration techniques.

To provide context, it is useful to have a concrete framework to discuss these practical issues. For this purpose consider the stochastic version of the neoclassical growth model, with CRRA preferences, a persistent TFP process, z_t, and partial depreciation of capital:

$$\begin{aligned} V(k,z) &= \max_{c,k'} \left\{ \frac{c^{1-\gamma}}{1-\gamma} + \beta \mathbb{E}\left(V(k',z') \,|z\right) \right\} \\ \text{s.t} \quad c + k' &= zk^{\alpha} + (1-\delta)k \\ \ln z' &= \rho \ln z + \eta', \qquad\qquad k' \geq k_{\min}. \end{aligned} \tag{P1}$$

We are now looking to obtain a numerical solution, which consists of decision rule for optimal savings, $k' = g(k,z)$, as well as a value function, $V(k,z)$, that "solve" the dynamic programming problem—that is, satisfy the functional equation (P1) with negligible error.[1]

I. How to Represent V?

The value function that solves a Bellman equation is the product of all the features of the dynamic program—from the shape of the utility function to the choices (or margins of adjustment) available, and from the nature of uncertainty to the types and tightness of constraints. The simple example (13.13) studied in the previous chapter yielded a value function that was log-linear in k, which we could solve by guess-and-verify. And as noted there, this was a very exceptional case: there are only a handful of problems (that are sufficiently rich and interesting) in which the

[1] Once we have the savings rule, the consumption rule, $c(k,z)$, comes for free via the resource constraint.

value function belongs to a simple parametric family.[2] More generally, we should work under the assumption that the value function will not have an analytical formula. Solution methods can be broadly categorized into two groups, according to how they deal with this issue.

Global Parametric Methods. The first approach is to appeal to a variety of approximation theorems in mathematics that broadly establish that under rather mild assumptions any "well-behaved" general function can be approximated to any desired accuracy by a finite number of *simpler* (basis) functions, such as polynomials or rational functions.[3] Thus, we can parametrize V over its entire domain using a convenient family of basis functions, which is fully characterized by a finite set of parameters. Therefore, solving the Bellman equations is reduced to searching over a finite set of parameter values, rather than searching in the infinite space of functions. It should be evident that the parametrization should be flexible enough that we can be confident it does not impose any unintended (and artificial) constraints on the problem. This approach is global, in the sense that the same basis functions and coefficients represent the entire value function over its entire domain. A variety of implementations based on this approach exist in the economics literature, and have found widespread usage especially in representative-agent models.[4]

Grid-Based Methods. The value functions and decision rules that we will encounter in models with heterogeneity often display challenging features, such as a very high curvature in certain parts of the domain combined with remarkable flatness in other parts, as well as multiple kinks (or points of non-differentiabilities) in some other applications. Thus, a second approach that provides more flexibility—at cost of some computational speed—is to approximate V in a piecewise fashion. That is, as discussed in Chapter 8 (Interpolation), the idea is to divide the domain of V into subdomains—most commonly, a multi-dimensional grid—and approximate V with separate basis functions within each subdomain. Because each approximation is over a small range, this allows us to use lower order polynomials, which also helps with the stability of the numerical algorithm. One idea described in detail in Chapter 8 was Spline interpolation and we will make good use of that knowledge in this and coming chapters.

[2] Another class of problems with explicit solutions are those with linear constraints and quadratic objective that has been exploited thoroughly by researchers in many fields. While this framework provides a useful departure point, it is rarely very useful for state of the art quantitative and empirical work.

[3] Perhaps the most famous of these theorems is the *Weierstrass approximation theorem*, which establishes that any continuous real-valued function over a closed interval can be uniformly approximated with polynomials. This theorem has been substantially extended in the *Stone-Weierstrass theorem* to general compact Hausdorff spaces and more general classes of basis functions.

[4] As noted at the outset, I will restrict attention to techniques that are most suitable for heterogeneous-agent models. Parametric methods are covered in great detail in many textbooks. See, e.g., Judd (1998); Heer and Maussner (2009).

For example, in problem (P1), we would construct a discrete grid (for current period state variables) as the product of individual grids in each state variable—$k_i \in G_k \equiv \{k_1, k_1, ..., k_I\}$ and $z_j \in G_z \equiv \{z_1, z_1, ..., z_J\}$. When needed, the values off the grid are obtained by various types of interpolation to provide a continuous and differentiable decision rule over the rectangle $G_k \times G_z = [k_0, ..., k_I] \times [z_0, ..., z_J]$.[5] Therefore, V would be represented by a matrix of values $\{V_{ij}, i = 1, ..., I; j = 1, ..., J\}$ on the grid and an appropriate interpolation algorithm everywhere else (off-grid).

In grid-based methods, it will be important to distinguish between the true continuous state variables and the (finite) set of grid points. We use (k, z) to denote the actual variables and denote particular values on the grid with subscript indices: e.g., (k_i, z_j). Therefore, we will seek to solve the following problem:

$$\begin{aligned} V(k_i, z_j) &= \max_{c,k'} \left\{ \frac{c^{1-\gamma}}{1-\gamma} + \beta \mathbb{E}\left(V(k', z') \,|z_j\right) \right\} \\ \text{s.t} \quad c + k' &= z_j k_i^{\alpha} + (1-\delta) k_i \\ \ln z' &= \rho \ln z_j + \eta', \qquad\qquad k' \geq k_{\min}. \end{aligned} \tag{P2}$$

The lack of subscript on c and k' is to emphasize that these variables are chosen from a continuous choice set. Here I have also not put a subscript on η', which allows z' to take on continuous values. We shall allow this approach in some applications whereas in others we will restrict z' to lie on the grid G_z. As discussed in the previous chapters on numerical tools, the choice of grid, including how to space the grid points and where to place the minimum and maximum, require careful consideration. There is more to be said on these, but I will postpone that discussion until the next chapter. For the purposes of the general discussion of algorithms that follow, it will be sufficient to think that a sensibly constructed grid is available.

II. Initial Guess and Stopping Rule

(Educated) Initial Guess for V_0

Many numerical algorithms, especially those solving with large-scale problems, have an iterative nature, which makes the starting point a critical part of the solution algorithm. A good choice can mean quick convergence to the correct solution, a bad one can slow down the algorithm or, even worse, can cause it to crash.[6] Therefore,

[5] And in fact even outside of it through extrapolation, but that is not recommended unless necessary. See Chapter XX.

[6] Unless an algorithm is *globally convergent* (and they are the exception rather than the rule), it is only guaranteed to converge if the starting point is inside its *basin of attraction.* As we have seen, the contraction mapping theorem ensures that the VFI is globally convergent but many of the subsequent algorithms we will cover are not.

the choice of initial points will be a recurring theme in many methods covered in this book. So, how do we find a good starting point?

In many DP problems, a typically harmless choice for V_0 is the zero function: $V_0 \equiv 0$ or more generally a constant function $V_0 = c$. This is because a constant function is continuous, bounded, weakly convex and weakly concave, all at once. Most Bellman operators, T, will quickly map this function in the next iteration into something that is consistent with the properties of the problem (for example, into a strictly concave and strictly increasing one).[7] However, the fact that this choice is feasible does not mean that it is a good idea. Recall that the convergence rate of VFI is linear in the modulus, which is typically the discount rate, β, in our problems. This is can be very slow when β is close to 1, which is often the case (especially, in models with monthly or weekly frequencies). Thus, picking an "initial guess", V_0, that is as close to the true solution as possible is essential, and $V_0 \equiv 0$ is almost never a close enough choice.

A reliable heuristic is to begin by first solving a simplified version of the problem that we eventually intend to solve. Such a simplified problem is obtained, for example, by ignoring certain constraints or eliminating uncertainty. The key is that this problem should be much faster and easier to solve than the original problem. For example, in problem (P1) above, one approach is to solve a deterministic version of this problem by setting z to its mean. This problem should be much faster to solve because it has one fewer state variable: for example, if we use a discrete grid for z with 20 points, the deterministic problem will require 20 times fewer maximizations. Furthermore, because there is no uncertainty, there is no need to compute a conditional expectation.[8] Finally, there is no need—and really not much benefit—to solve this preliminary problem to full convergence, since it is only an initial guess that will be quickly updated in the real problem. Even iterating on this problem 30 or 50 times will provide a much better initial guess than simply starting with an uneducated guess.

In the preliminary problem, it is generally advisable not to change the grid or the bound from the original problem and would definitely not change the preference parameters.[9] After all, our goal is to get a good approximation to the general shape of the value function, and preferences (especially the curvature of utility) matters greatly for this shape, often much more than uncertainty or constraints.

The heuristic described here is applicable much more generally. For example, in a heterogeneous-agent model a good choice of V_0 is often obtained by first solving the representative-agent counterpart of the same problem. Similarly, in a general

[7] The zero function also has an intuitive interpretation in a life cycle model as the value function in the last period of life (with no bequests).

[8] Of course, this solution will generate a ??

[9] There is a set of acceleration algorithms that work by aggregating the number of states. That is, they proceed by solving the problem on a coarser grid initially, and increase the fineness of the grid as one gets closer to the true solution. Give citation..

equilibrium model with aggregate shocks, V_0 can be taken to be the value function that solves the counterpart problem with no aggregate shocks. And so on. In fact, it is not unusual to solve a sequence of increasingly richer preliminary models and using the solution of each as the initial guess for the next problem, all the way up to the final model that we intend to solve. However, in more complicated problems, sometimes

We discuss more examples in Chapter 21.

Stopping Rule: When to Terminate the Algorithm?

We terminate a numerical algorithm when the current solution V_n is "sufficiently close" to the fixed point, V^*. In practice, of course, the latter is not known, so the stopping rule needs to rely on numbers that can be computed during the course of the numerical algorithm. Since Richard Bellman's original contribution, a number of authors have derived increasingly tighter "error bounds" that link the computed solutions to some measure of the true error: $d(V_n, V^*)$ where d is a distance measure. For example, the basic bound given in (13.1.2) says that the maximum gap between the current solution and the true one cannot be greater than the maximum gap between the current and last solutions, multiplied by $1/(1-\beta)$. When β is close to 1, this scaling factor can be very large, so it bears to pay attention to it.

One challenge with this formula is that it is expressed in the level of the value function. But it is not unusual for the value function to cover a very wide range of values across its domain. This is especially true when we consider a wide capital grid, when the utility curvature is high, and very low income levels are possible. One idea is to stop when the change in the value function is small in percentage terms. But this requires a lot of care because the value function can change signs (and hence cross zero!). Parameterizing the value function as a CES form as suggested in Chapter 8.5 comes in handy for this purpose too, as it ensures that the value function is always positive and its range of values does not vary too much. We will discuss other ideas later, such as basing the stopping rule on the change in decision rules (which almost always converge faster than does the value function), checking accuracy further with Euler equation errors (Marcet and den Haan, etc.), and using alternative tighter bounds that can also help accelerate the algorithm as we shall see in the next section.

III. Initialization

The third issue is chronologically the first step in implementing any algorithm: the initialization of various mathematical objects that are part of the algorithm. It includes the choice of bounds for each state variable, as well as the spacing of grid points: i.e., the choice of G_k and G_z. It also covers decisions regarding how to

handle cases when one or more of the bounds are exceeded during the course of the algorithm.

Choice of Bounds.

The choice of bounds should be done by a careful study of the economics of the dynamic programming problem. To see a concrete example, consider again problem (P2). A natural approach would be to use the budget constraint to eliminate c as a choice variable, so the Bellman objective reads:

$$\max_{k' \in \Gamma(k_i, z_j)} \left\{ \frac{(z_j k_i^{\alpha} + (1-\delta)k_i - k')^{1-\gamma}}{1-\gamma} + \beta \mathbb{E}\left(V\left(k', z'\right) | z_j\right) \right\}. \tag{14.2.1}$$

There are three issues to address here: (i) how to determine the bounds for k_i grid? (ii) how to determine the bounds for k', that is the choice set, $\Gamma(k_i, z_j)$, when performing the maximization? These two issues are closely related and will be addressed together. A third issue is how to determine bounds for z_j grid. This choice is largely dictated by the stochastic properties of z and has been discussed extensively in Chapter 10. We now discuss the first two issues.

Grid for k_i Because of the constraint $k' \geq k_{\min}$, the capital value can never fall below $k_{\min}$, which gives us the lower bound: $k_1 \equiv k_{\min}$. The choice of upper bound, k_I, is less straightforward and depends on many considerations. I discuss some of them here, and others that are model-specific will be mentioned later when those models are presented.

An important consideration in deciding on the value of k_I is the maximization problem that the agent faces at that state. Let $k'^*(k_I, z_j)$ be the optimal savings choice when the agent's assets today is at the upper bound. It is entirely possible that this savings choice exceeds the current capital level: $k'^*(k_I, z_j) > k_I$. A common (first-time) mistake is to restrict the agent's choice set for k' to be bounded by k_I, thereby imposing a constraint that at the highest capital level the agent cannot save a positive amount: $k'(k_I, z_j) \leq k_I$. Clearly, this restriction is not part of the original—theoretical—problem. If the parameters of the problem and the current guess about the value function are such that this constraint binds, this will lead to a distortion in the shape of the updated value function and decision rules at the top end, which then propagates downward as we iterate on the Bellman equation, possibly breaking down the solution algorithm.

How do we deal with this? First, we need to understand the structure of the problem. In particular, if the problem is such that the resulting savings choice leads to a stationary distribution of capital in the long run, our job becomes a bit easier. Because then we can choose the upper bound such that the upper end

of the stationary distribution is fully within the grid that we constructed. Then we can be sure that at k_I the agent's optimal decision will be not to increase his capital holdings—so the constraint will not bind. The tricky issue of course is that, we cannot compute the stationary distribution, without first solving the dynamic program! One solution is to first take a sufficiently large value of k_I that we can be confident the constraint will not bind. This will require us to use lots of grid points, possibly more than what is efficient. But once we solve this problem we can compute the stationary distribution and then adjust k_I downward to a level that still leaves breathing room but is not excessively wide. The problem with a grid that is too wide is that it is not efficient as we solve the problem in a range of capital values that will never be realized in any simulation.[10] As I will discuss in Chapter 19, Den Haan (2010a) evaluates seven different methods that solve the Krusell-Smith model and concludes that the most precise and fastest methods are those that do not waste grid space in regions of the capital space that are irrelevant.

The described approach is not always feasible however. For example, sometimes we will be calibrating or estimating such a model and as we search over the parameter space, the grid that is wide enough for one parameterization can be small for another one. So the artificial constraint could be binding sometimes and not binding other times, which in turn makes the whole exercise suspect. As a precaution against such scenarios, one could also do the following: create a virtual grid point beyond the upper bound, $\tilde{k}_{I+1} = k_I + (k_I - k_{I-1})$. It should be placed probably no more than one full grid spacing beyond the upper bound. Then define the choice set to be $k' \in \Gamma(k_I, z_j) = [k_1, \tilde{k}_{I+1}]$.[11] Of course, the value function is only computed for values of $k \in [k_1, k_I]$, so we need to (carefully!) extrapolate between k_I and $\tilde{k}_{I+1}$. Because extrapolation is a very risky endeavor, one needs to be very cautious in how this is implemented. One relatively safe idea is to take a convex combination of a spline extrapolation and a linear extrapolation, whereby we increase the weight of the linear component from zero at k_I to one at $\tilde{k}_{I+1}$.[12] Like I said, most of the time our grid will be wide enough that this is not a major issue and $k'(k_I, z_j) < k_I$ naturally.

One of the algorithms that will be discussed later in this chapter—the endogenous grid method—has the appealing feature that maximization at the upper bound ceases to be an issue. We will see how that works momentarily.

One point of this rather long discussion is to hopefully convince the reader that choosing grids for state variables, especially endogenous ones, requires careful

[10] Expanding grids as described in Chapter 8.4 help in this respect, because doubling the upper bound of the grid typically results in a small increase the number of points. This is because the value function and decision rules are typically quite linear at the top end, so even few grid points at the top end usually deliver a satisfactory results when interpolating.

[11] Of course, we need to make sure that this choice set does not violate any other constraints, such as, e.g., making consumption non-positive.

[12] Specifically, $\hat{V}(k) = \theta V_{\text{spline}}(k) + (1-\theta)V_{\text{linear}}(k)$ and $\theta = (\tilde{k}_{I+1} - k)/(\tilde{k}_{I+1} - k_I)$, where $\hat{V}$ is the final extrapolated value.

consideration. It also makes sense to automate the choice of upper bound by linking it to the parameters of the model, such as β and γ as well as the variance of the shock, so that when we change the calibration the grid scales automatically.

Finally, there are some other issues that arise in the construction of multi-dimensional grids, such as paying attention that the constraints in the problem do not prevent the construction of a rectangular grid.[13] These will be discussed in more detail in Chapter xyz.

Spacing of Grid Points In Chapter 8, we discussed grid spacing in detail. One issue that has not been covered there is what to do when the value function or the decision rule (or both) has a kink (or multiple kinks). In such cases the advise of putting more grid points where there is curvature still broadly applies, but the location of the kinks hence of curvature is not as easy to pin down. One idea that has been pursued in some papers is to construct a double exponential grid—that is a grid that expands in two directions—starting from somewhere in the middle of the range.

14.2.1 VFI Algorithm: Details

Now that we have discussed the various essential preliminaries, we are ready to discuss how to implement the VFI algorithm on a computer.

Algorithm 7: VFI Algorithm: Details

Step 0. Construct an appropriate grid for state variables: $(k_i, z_j) \in G_k \times G_z$. Choose an educated initial guess $V_0 \in S$. Set $n = 0$.

Step 1. For each element in the (k_i, z_j) grid, execute the mapping $V_{n+1}(k_i, z_j) = TV_n(k, z)$. For $(k, z) \notin G_k \times G_z$, obtain $V_n(k, z)$ by an appropriate interpolation method.

Step 2. Stop if convergence criterion satisfied: $\|V_{n+1} - V_n\|_\infty < \texttt{toler}$. Otherwise, increase n and return to step 1.

While this algorithm is reliable, it is often too slow. In the rest of this chapter we cover a variety of acceleration techniques that can speed it up, often by *several orders of magnitude*. So, this basic algorithm serves as a foundation for what is to come, and is almost never a good one to implement on its own.

[13] Non-rectangular grids can be handled but often at high cost. They prevent the application of many convenient techniques or make them significantly more complicated.

14.3 Accelerating VFI: Two Useful Tricks

14.3.1 Policy Iteration Algorithm

To understand why the standard VFI algorithm is often slow, notice that it works by repeatedly applying the Bellman mapping, which in fact does two things at once. First, it maximizes the right hand side of the Bellman equation for every grid point today's state space. That is, define $s \equiv k'$ (for clarity of notation in the following discussion) and for all i, j solve:

$$\tilde{s}_n(k_i, z_j) := \arg \max_{s \geq k_{\min}} \left\{ \frac{(z_j k_i^{\alpha} + (1-\delta)k_i - s)^{1-\gamma}}{1-\gamma} + \beta \mathbb{E}\left(V_n\left(s, z'\right)|z\right) \right\}. \quad (14.3.1)$$

This step can be extremely time consuming for obvious reasons. When the maximization problem is multi-dimensional, this is even more the case. The second step is then to take the maximand in the first step and evaluating the right hand side with this policy once to update the value function from V_n to V_{n+1}:

$$V_{n+1} = T_{\tilde{s}_n} V_n.$$

Recall that this second step is simply the Howard mapping defined above and it is very fast since it only requires a single evaluation of the Bellman objective. A simple but key insight is that the Howard mapping itself is also a contraction with modulus β. Therefore if we were to repeatedly apply $T_{\tilde{s}_n}$, it would also converge to a fixed point itself at rate β. Of course, this fixed point would not be the solution of the original Bellman equation we would like to solve (unless the current policy $\tilde{s}_n$ happens to be the optimal one).[14] But because each application of the Howard mapping costs little, we would be converging much faster than applying T. So one strategy called the (Howard's) policy iteration algorithm is to apply Howard operator until convergence in between every application of the Bellman operator. Therefore, the *VFI with policy iteration algorithm* can be written as:

Two important properties of policy iteration algorithm can be understood via an important result established by Puterman and Brumelle (1979). These authors show that policy iteration is equivalent to Newton's method (for solving nonlinear equations) applied to dynamic programming. Thus, just like Newton's method it has two properties: (i) it is guaranteed to converge to the true solution when the initial point, V_0, is in the domain of attraction of V^*, and (ii) when (i) is satisfied, it converges at a *quadratic rate* in iteration index n.

[14] Another way to think of what the standard VFI entails is that we go to great lengths to maximize the Bellman objective to obtain a new optimal policy but then use it only for the current period when updating from V_n to V_{n+1}. Instead, Howard's algorithm discards V_n once step n is completed and takes the new policy, $\tilde{s}_n$, to apply at all future dates to obtain V_{n+1}.

Algorithm 8: VFI with Policy Iteration Algorithm

Step 0: Set $n = 0$. Choose an initial guess $V_0 \in S$.

Step 1: Obtain $\tilde{s}_n$ as in (14.3.1) and take the updated value function to be: $V_{n+1} = \lim_{m\to\infty} T^m_{\tilde{s}_n} V_n$, which is the (fixed point) value function resulting from using policy, $\tilde{s}_n$, forever.[15]

Step 2: Stop if convergence criteria satisfied: $\|V_{n+1} - V_n\|_\infty < \texttt{toler}$. Otherwise, increase n and return to step 1.

So there is both good news and bad news. The good news is that we are able to move from the linear rate under VFI to a quadratic rate. However, this is not an apples to apples comparison, because Howard's algorithm needs to perform many more computations per iteration compared with standard VFI, because the former needs to solve for the value function assuming policy $\tilde{s}_n$ is used forever, whereas VFI requires a single evaluation of the Bellman equation for the policy. This difference becomes especially important when the state space is large, which can slow down Howard's algorithm considerably. We will look into a modification to alleviate this problem in a moment.

Second, Puterman and Brumelle's bad news is that Howard's algorithm can simply fail if we start from afar. To understand why, notice that $\lim_{m\to\infty} T^m_{\tilde{s}_n} V_n$ is the wrong limit for the Bellman equation, especially early on, when the current iterate of the policy rule is far away from the true solution. This means that, at a given stage n, the more we iterate on the Howard mapping, the more we might (and generally would) converge rapidly to the wrong point. Of course, then applying the Bellman mapping should bring us back towards the correct solution, but in general there are no guarantees. In other words, the policy improvement algorithm can generate oscillations (sometimes large ones) that can make the whole algorithm unstable, causing it to crash.[16] Therefore, a sensible approach is to start with standard VFI (or other methods we discuss in a moment) that are guaranteed to converge and then applying policy iteration as we get closer to the solution to benefit from quadratic rates of convergence.

Modified Policy Iteration Algorithm

A simple modification to the basic algorithm solves both problems. The modified policy iteration algorithm essentially takes the number of applications of the Howard

[16] One can prove convergence from any v_0 when the state space is discrete. However, in most of the analysis in this book we will consider a continuous state space, especially in key endogenous variables.

mapping to be a finite number m.[17] Therefore we modify Step 1 above to the following:

Algorithm 9: VFI WITH MODIFIED POLICY ITERATION ALGORITHM

Step 1': Modify *Step 1* of Howard's algorithm: Obtain $\tilde{s}_n$ as in (14.3.1) and take the updated value function to be: $V_{n+1} = T^m_{\tilde{s}_n} V_n$, which entails m applications of Howard's mapping to update to V_{n+1}.

In small and well-behaved problems, especially those that do not involve an outer loop (iterating on prices or other general equilibrium objects), we can usually take m to be fairly large, 50 or even 100. In my experience, for high β, sometimes values as high as 500 continue to improve the computation speed. However, this comes with a caution that the higher m is the closer we get to the original Howard algorithm with its potential pitfalls of oscillations and non-convergence. Therefore, in high dimensional problems and with general equilibrium, it seems prudent to use a smaller value of m, say 20 or less. And as noted above, it is usually advisable to start with a lower m (e.g., $m = 1$ is a good choice) and gradually increase it as we approach the solution.

14.3.2 Acceleration by Error Bounds

Another simple trick makes use of some sharp "error bounds" that can be derived for VFI algorithm. To describe the method, it is useful to adopt the discrete state representation for dynamic programs given in Section 13.6. The following theorem defines these bounds.

Theorem 14.1. [MacQueen-Porteus Error Bounds] In the discrete state dynamic programming problem,

$$V(x_i) = \max_{y \in \Gamma(x_i)} \left[U(x_i, y) + \beta \sum_{j=1}^{J} \pi_{ij}(y) V(x_j) \right], \tag{14.3.2}$$

define

$$\underline{c}_n = \frac{\beta}{1-\beta} \times \min \left[T^n V_0 - T^{n-1} V_0 \right] \tag{14.3.3}$$

$$\bar{c}_n = \frac{\beta}{1-\beta} \times \max \left[T^n V_0 - T^{n-1} V_0 \right] \tag{14.3.4}$$

[17] Denardo (1967) appears to be the first one to propose this algorithm. It has been studied more extensively by van Nunen (1976) and especially Puterman and Shin (1978) (which also explores several variations of it) and Puterman and Brumelle (1979) who establish its equivalence to the Newton-Kantorovich iteration for solving nonlinear equations.

Then, for all $\overline{x} \in X$, we have:

$$T^n V_0(\overline{x}) + \underline{c}_n \leq V^*(\overline{x}) \leq T^n V_0(\overline{x}) + \overline{c}_n. \tag{14.3.5}$$

Furthermore, with each iteration, the two bounds approach the true solution monotonically.

The theorem has several substantive implications. First, these bounds can be quite tight. For example, suppose that in the kth iteration, the value function changes by a constant amount across all values of $\overline{x}$, that is, $T^n V_0(\overline{x}) - T^{n-1} V_0(\overline{x}) = \alpha$ for all $\overline{x}$. Then equations (14.3.3) and (14.3.4) imply that $\underline{c}_n = \overline{c}_n$, which the then implies

$$\frac{\alpha\beta}{1-\beta} = V^*(\overline{x}) - T^n V_0(\overline{x}) = \frac{\alpha\beta}{1-\beta}.$$

Therefore, $V^*(\overline{x}) = T^n V_0(\overline{x}) + \frac{\alpha\beta}{1-\beta}$, and we are done! Although this is a special example that is not likely to arise in real applications, it illustrates the power of these bounds more generally.

Second, it is also useful to contrast these bounds to the simpler bounds derived from the simple contraction argument, given in the previous chapter. This was

$$\|V^* - T^n V_0\|_\infty \leq \frac{1}{1-\beta} \left\|T^n V_0 - T^{n-1} V_0\right\|_\infty = \frac{\alpha}{1-\beta}. \tag{14.3.6}$$

First, unless $\beta = 1$, this bound is looser than the MacQueen-Porteus bound. But second and more importantly, it provides an upper bound in the supnorm, whereas (14.3.5) provides both an upper and a lower bound. For example, in the usual VFI, we would iterate until the upper bound in (14.3.6) is smaller than a small tolerance value. However, with Theorem 14.1 we would stop when $\overline{c}_n - \underline{c}_n$ is smaller than a small tolerance. In the example just given, $\overline{c}_n = \frac{\alpha\beta}{1-\beta}$ and this value might be a large number, say 100, but we would still stop because $\overline{c}_n - \underline{c}_n = 100 - 100 = 0$, whereas we would keep iterating under VFI with regular bounds since the upper bound is still high.

Therefore, an improved algorithm replaces the last step of the VFI algorithm as follows:

This algorithm deserves attention for several reasons. First, it is really simple to implement. It only requires the computation of the maximum and minimum deviation, which can be performed quite easily and quickly. Second, its convergence is guaranteed under the same assumptions about the convergence property of the contraction mapping theorem. Third, its convergence rate can be quite fast in certain situation. In particular, using the discrete state Markov decision problem framework presented in Section 13.6, Bertsekas (2001) proves that the convergence

Algorithm 10: VFI WITH MACQUEEN-PORTEUS ERROR BOUNDS

Step 2': Stop when $\overline{c}_n - \underline{c}_n <$ `toler`. Then take the final estimate of V^* to be either the median

$$\tilde{V} = T^n V_0 + \left(\frac{\overline{c}_n + \underline{c}_n}{2} \right)$$

or the mean (i.e., average error bound across states):

$$\hat{V} = T^n V_0 + \frac{\beta}{n(1-\beta)} \sum_{i=1}^{n} \left(T^n V_0(\overline{x}_i) - T^{n-1} V_0(\overline{x}_i) \right).$$

TABLE I – Mc-Queen Porteus Bounds and Policy Iteration: Speed Improvements

$\beta \rightarrow$	0.95			0.99			0.999		
$MQP/N_{max} \rightarrow$ ↓	0	50	500	0	50	500	0	50	500
	$\gamma = 1$								
no	14.99	1.07	1.00	26.48	1.28	1.00	33.29	1.41	1.00
yes	0.32	0.60	0.79	0.10	0.23	0.27	0.01	0.03	0.04
	$\gamma = 5$								
no	13.03	0.96	1.00	26.77	1.28	1.00	33.37	1.45	1.00
yes	0.67	0.67	0.69	0.14	0.24	0.30	0.02	0.04	0.06

Note: N_{max} indicates the maximum number iterations in the modified policy algorithm. Results remain unaffected if we stop at $N_{max} = 100$ or for intermediate values of utility curvature. The timing for $N_{max} = 500$ and no MacQueen-Porteus bounds is normalized to 1.0 for each (β, u) combination to show relative speed gains. The second row in each panel (marked with MPQ=yes) report the fraction of time it takes the computation to finish when MacQueen-Porteus bounds are applied relative to the cell immediately above it (with the same number of iterations in policy stage).

rate of this algorithm (for both $\tilde{V}$ and $\hat{V}$) is governed by the modulus of the *subdominant* eigenvalue of the transition matrix $\pi_{ij}(y^*)$, that is, the eigenvalue with the second largest modulus.[18] Recall that if the transition matrix has a unique ergodic distribution, then its largest eigenvalue is 1 and when multiplied with the discount factor, this yields a convergence at rate β. When β is close to 1, the VFI algorithm becomes extremely slow. Bertsekas' result shows that under the right conditions, we can improve from the largest eigenvalue to the second largest. As long as this is less than 1, the convergence rate can be faster and often much faster. In the important special case where a Markov chain is constructed by discretizing an AR(1) process, the subdominant eigenvalue is given by the persistence parameter ρ. Therefore, the less persistent the stochastic process is the more MacQueen-Porteus bounds will help accelerate convergence.

[18] In Bertsekas' formulation the state space is discrete but can be infinite. Notice that $\pi_{ij}(y^*)$ is the transition matrix induced by the optimal policy.

14.4 VFI with Endogenous Grid Method

As noted earlier, the different versions of the VFI algorithm described so far all face the same bottleneck: the maximization step of the algorithm is slow and accounts for a substantial fraction of the computational time of each algorithm. The endogenous grid method (EGM) introduces two key modifications that can significantly speed up the maximization step and, consequently, the VFI algorithm itself.

First, we construct the capital grid based on tomorrow's values and let today's capital grid be determined endogenously. Second, we adopt a slightly different timing for selecting the state variables. In particular, we write the value function as a function of the *end-of-period* total resources, rather than the usual choice of *beginning-of-period* capital.

To understand how these modifications help, first recall that the VFI method presented in the previous section proceeds by backward iteration and maximizes the right-hand-side of the Bellman equation. Assuming an interior solution, the FOC is:

$$c^{-\gamma} = \beta \mathbb{E}\left(V_k\left(k', z'\right)|z_j\right),$$

where V_k denotes the derivative of V with respect to capital. This equation can be rewritten (by substituting out consumption using the budget constraint) as

$$\left(z_j k_i^{\alpha} + (1-\delta)k_i - k'\right)^{-\gamma} = \beta \mathbb{E}\left(V_k\left(k', z'\right)|z_j\right), \tag{14.4.1}$$

In the standard VFI, we solve this equation for k' for each value of k_i and z_i. This is a costly computation for three reasons. First, this is a *non-linear* equation in k'. Second, when searching for a solution over values of k', we need to repeatedly evaluate the conditional expectation on the right hand side for every trial value of k' (since it appears inside the expectation). Third, in standard VFI, the value function is stored at grid points defined over k, so for every trial value of k', we need to interpolate (or, even worse, extrapolate!) the tabulated values $\{V(k_i, z_j), i = 1, ..., I; j = 1, ..., J\}$ to obtain off-grid values $V(k', z'_j)$ for each z'_j.[19] Each one of these steps add significantly to the cost of solving equation (14.4.1).

In the endogenous grid method (hereafter, EGM), we take the same equation but use it differently. The idea is to construct a grid for tomorrow's capital, k', and today stochastic state z. We then solve for c and k as a function of k'_i and z_j only

[19] Of course, if we proceed this way, at the very least we should construct the interpolating function, such as a spline, and compute its coefficients once and store them. Then for every choice of k_{t+1} we can simply use the stored coefficients to evaluate the interpolant at new points, which is much faster than re-computing coefficients (which is redundant).

on these grid points. That is, we view the problem as:

$$\begin{aligned} V(k, z_j) &= \max_{c, k_i'} \left\{ \frac{c^{1-\gamma}}{1-\gamma} + \beta \mathbb{E}\left(V(k_i', z') \,|z_j\right) \right\} \\ \text{s.t} \quad c + k_i' &= z_j k^{\alpha} + (1-\delta)k \\ \ln z' &= \rho \ln z_j + \eta', \\ k' &\geq k_{\min}. \end{aligned} \tag{P3}$$

Comparing this equation to problem P2 above, notice the subtle change in the placement of the subscript i from today's capital stock previously, to tomorrow's capital here. (I also switched to a discrete shock space, which will come in handy in a moment). To see how this simplifies things, notice that the problem is going to be solved for capital values, k_i, on the grid tomorrow. This step involves no interpolation since k_i' is on the grid and, as well shall see, we will compute this once. Now rearrange the right hand side of (14.4.1) to obtain:

$$z_j k^{\alpha} + (1-\delta)k = \left[\beta \mathbb{E}\left(V_k(k_i', z') \,|z_j\right)\right]^{-1/\gamma} + k_i'.$$

Solving this equation for k is easier, because—even though we still need to solve a nonlinear equation—the conditional expectation needs to be computed once, unlike what we had to do before (as we searched over values of k').

The second clever idea in the EGM method is that we can avoid even solving this nonlinear equation by carefully changing the state variable from the beginning-of-period capital stock, k_t, to end-of-period total cash-on-hand, or equivalently, output plus undepreciated capital. Define

$$Y \equiv z k^{\alpha} + (1-\delta)k \tag{14.4.2}$$

and rewrite the Bellman equation as:

$$\begin{aligned} \mathcal{V}(Y, z) &= \max_{k'} \left\{ \frac{(Y - k')^{1-\gamma}}{1-\gamma} + \beta \mathbb{E}\left(\mathcal{V}(Y', z') \,|z_j\right) \right\} \\ \text{s.t} \quad \ln z' &= \rho \ln z_j + \eta'. \\ Y' &= z' k'^{\alpha} + (1-\delta)k'. \end{aligned}$$

The key observation is that Y' is only a function of k_i' and z', so we can write the conditional expectation on the right hand side as:

$$\mathbb{V}(k_i', z_j) \equiv \beta \mathbb{E}\left(\mathcal{V}(Y'(k_i', z'), z') \,|z_j\right).$$

The problem can then be written as:

$$\mathcal{V}(Y,z) = \max_{k'} \left\{ \frac{(Y-k')^{1-\gamma}}{1-\gamma} + \mathbb{V}(k'_i, z_j) \right\}$$

Now the FOC of this new problem becomes:

$$c^*(k'_i, z_j)^{-\gamma} = \mathbb{V}_{k'}(k'_i, z_j). \tag{14.4.3}$$

Having obtained $c^*(k'_i, z_j)$ from this expression, we use the resource constraint to compute today's end-of-period resources, $Y^*(k'_i, z_j) = c^*(k'_i, z_j) + k'_i$, as well as

$$\mathcal{V}(Y^*(k'_i, z_j), z_j) = \frac{(c^*(k'_i, z_j))^{1-\gamma}}{1-\gamma} + \mathbb{V}(k'_i, z_j)$$

on the left-hand-side. Therefore, given tomorrow's grid, we are able to find today's state variable, Y^*, and choice variable without requiring a non-linear solver. However, the values of $\mathcal{V}$ we obtained are on a discrete grid of values given by $(Y^*(k'_i, z_j), z_j)$ which is not on the (k'_i, z_j) grid, since $Y^*$is not on this grid (necessarily). But this is easy to fix. We can obtain the values of $\mathcal{V}$ on the latter exogenous grid by interpolation. Then we can plug the updated and interpolated value function into the RHS and iterate until convergence.

Notice that we have not solved for the *beginning-of-period* capital stock, k. This can be obtained by solving (14.4.2) for k, given grid points of $Y(i,j)$. This is a non-linear equation, but needs to be solved only once, after the Bellman equation has been solved. Here is the full algorithm.

14.4.1 EGM With CES Curvature Reduction Formulation

WORK in PROGRESS....

14.4.2 Other Advantages/Disadvantages of EGM

➢

14.4.3 EGM with Labor Supply

➢ Barillas and Fernández-Villaverde (2007). Maliar and Maliar (2013) paper proposes another approach to deal with labor supply.

Algorithm 11: Endogenous Grid Method

Step 0: Set $n = 0$. Construct a grid for tomorrow's capital and today's shock: (k'_i, z_j). Choose an initial guess $\mathbb{V}^0(k'_i, z_j)$.

Step 1: For all i, j, obtain

$$c^*(k'_i, z_j) = \left(\mathbb{V}_k^n(k'_i, z_j)\right)^{-1/\gamma}.$$

Step 2: Obtain today's end-of-period resources as a function of tomorrow's capital and today's shock:

$$Y^*(k'_i, z_j) = c^*(k'_i, z_j) + k'_i,$$

and today's updated value function,

$$\mathcal{V}^{n+1}\left(Y^*(k'_i, z_j), z_j\right) = \frac{\left(c^*(k'_i, z_j)\right)^{1-\gamma}}{1-\gamma} + \mathbb{V}^n(k'_i, z_j)$$

by plugging in consumption decision into the RHS.

Step 3: Interpolate $\mathcal{V}^{n+1}$ to obtain its values on a grid of tomorrow's end-of-period resources: $Y' = z'(k'_i)^\alpha + (1-\delta)k'_i$.

Step 4: Obtain

$$\mathbb{V}^{n+1}(k'_i, z_j) = \beta \mathbb{E}\left(\mathcal{V}^{n+1}\left(Y'(k'_i, z'), z'\right) | z_j\right).$$

Step 5: Stop if convergence criterion is satisfied and obtain beginning-of-period capital, k, by solving the nonlinear equation $Y^{n*}(i, j) \equiv z_j k^\alpha + (1-\delta)k$, for all i, j. Otherwise, go to step 1.

14.4.4 EGM with Non-Differentiable Value Functions

➢ Fella (2014)

14.5 Euler Equation Method

$$1 = \mathbb{E}\left[\beta \frac{U'(c^*(z^{t+1}))}{U'(c^*(z^t))}(1 + \tilde{r}(z^{t+1}))\right].$$

Let $F(k) = zk^\alpha + (1-\delta)k$

$$U'(c_{n+1}(k, z)) = \mathbb{E}\left[\beta U'(c_n(F(k, z) - c_{n+1}(k, z), z')) \times F'(F(k, z) - c_{n+1}(k, z), z')\right].$$

Algorithm 12: Endogenous Grid Method with CES Trick

Step 0: Set $n = 0$. Construct a grid for tomorrow's capital and today's shock: (k'_i, z_j). Choose an initial guess $\mathcal{V}^0(Y', z_j)$ and compute $\mathbb{V}^0$.

Step 1: For all i, j, obtain

$$c^*(k'_i, z_j) = \mathbb{E}\left((1-\gamma)\frac{\partial \mathcal{V}^n(Y', z_j)}{\partial Y'}\frac{\partial Y'(k_i, z')}{\partial k'_i}\left(\mathcal{V}^n(Y', z_j)\right)^{1-\gamma}\right)^{-1/\gamma}.$$

Step 2: Obtain today's end-of-period resources as a function of tomorrow's capital and today's shock:

$$Y^*(k'_i, z_j) = c^*(k'_i, z_j) + k'_i,$$

and today's updated value function,

$$\mathcal{V}^{n+1}\left(Y^*(k'_i, z_j), z_j\right) = \left[\frac{\left(c^*(k'_i, z_j)\right)^{1-\gamma}}{1-\gamma} + \mathbb{V}^n(k'_i, z_j)\right]^{1/(1-\gamma)}$$

by plugging in consumption decision into the RHS.

Step 3: Interpolate $\mathcal{V}^{n+1}$ to obtain its values on a grid of tomorrow's end-of-period resources: $Y' = z'(k'_i)^\alpha + (1-\delta)k'_i$.

Step 4: Obtain

$$\mathbb{V}^{n+1}(k'_i, z_j) = \beta\mathbb{E}\left(\mathcal{V}^{n+1}\left(Y'(k'_i, z'), z'\right)|z_j\right).$$

Step 5: Stop if convergence criterion is satisfied and obtain beginning-of-period capital, k, by solving the nonlinear equation $Y^{n*}(i,j) \equiv z_j k^\alpha + (1-\delta)k$, for all i, j. Otherwise, go to step 1.

$$U'(c_{n+1}(k_i, z_j)) = \mathbb{E}\left[\beta U'(c_n(F(k_i, z_j) - c_{n+1}(k_i, z_j), z')) \times (1 + R(z'))\right].$$

$c = zk^\alpha + (1-\delta)k - k'$

At $t = T$, we know that $c(k'_i, z'_j) = k'_i$

Guess and Verify (Policy Functions)

Let the policy rule for savings be: $k' = g(k)$. The Euler equation is:

$$\frac{1}{Ak^\alpha - g(k)} - \frac{\beta\alpha A\left(g(k)^{\alpha-1}\right)}{A\left(g(k)^\alpha - g(g(k))\right)} = 0 \quad \text{for all } k.$$

which is a functional equation in $g(k)$. Guess $g(k) = sAk^{\alpha}$, and substitute above:

$$\frac{1}{(1-s)\,Ak^{\alpha}} = \frac{\beta\alpha A\left(sAk^{\alpha}\right)^{\alpha-1}}{A\left(\left(sAk^{\alpha}\right)^{\alpha} - sA\left(aAk^{\alpha}\right)^{\alpha}\right)}$$

As can be seen, k cancels out, and we get $s = \alpha\beta$. By using a very flexible choice of g, this method too can be used for solving very general models.

➢ We need Euler equation to be sufficient, so concave utility and convex choice set. Not always possible.

14.6 Taking Stock

Today, the state-of-the-art for solving dynamic programming problems is the endogenous grid method. It dominates the other methods discussed in this chapter (as well as others not covered here) in terms of both computational speed and programming time. Furthermore, it has been extended to models with non-concave objectives, problems with endogenous labor supply and is able to deal easily with certain types of constraints commonly encountered in economic applications. So, when it is applicable it should be your first choice. The problem is that it is not always applicable, perhaps most importantly, when the problem features more than one endogenous state variable. And these are the problems that are especially challenging in terms of overall speed requirements. In such cases, the modified policy iteration algorithm and the MacQueen-Porteus error bounds can be especially helpful in speeding up the solution. The bottom line is that all three of these methods should be part of your computational toolbox.

Policy iteration was first proposed Bellman (1957)'s seminal book (which he called "iteration in policy space"), but was substantially generalized by Howard (1960) and later by Denardo (1967), who also proved convergence results in increasingly more general frameworks.

Talk about Powell's and Karatzas and Shreve's methods that also rely on a different timing for the Bellman equation.

The idea of parameterizing the conditional expectation as a function of tomorrow's state variables goes back to Wright and Williams (1984), who used a parameterized expectations approach with current period decision (and next period's state). The endogenous grid method is originally due to Carroll (2006) and makes use of a similar idea but in a very effective way.

Chapter 15

Putting The Algorithms to Work

In this chapter, you will start picking the first fruits of the investments you have made since Chapter 8, by combining the tools and algorithms that have been covered so far. In particular, we will solve a canonical income fluctuations problem under a broad range of assumptions and parameterizations. To illustrate the pros and cons of different tools and methods, we will build our algorithm in a modular fashion and implement each piece by taking turns between several alternative tools. In doing so, we will also tie many loose ends regarding implementation.

The reader will notice that this chapter has a different tone from those before. If previous chapters were regular lectures, this one is intended to be a recitation session and office hours. As such, it will go deeper into details of the implementation and dispense practical advice on each and every step of the programming process. While many of these details are already well understood by experienced programmers, my goal here is to make sure first timers are also on board and can avoid costly mistakes (or inefficiencies).

15.1 The Problem and Its Parameterization

We solve the decision problem (income fluctuation problem) corresponding to the stochastic neoclassical growth model studied in the previous chapter. For complete-

ness, here it is once again:

$$V(k_i, z_j) = \max_{c,k'} \left\{ \frac{c^{1-\gamma}}{1-\gamma} + \beta \mathbb{E}\left(V(k', z') \,|z_j\right) \right\} \tag{15.1.1}$$
$$\text{s.t} \quad c + k' = (1+R)k + z$$
$$\ln z' = \rho \ln z_j + \eta', \qquad k' \geq k_{\min}.$$

15.1.1 Model Parameterization

The parameter choices for the model and the numerical algorithm are summarized in Table I.

Risk Aversion. We consider two values of the curvature parameter, γ, of 2 and 10. The former is in the ballpark of values often chosen in macroeconomic applications and the latter is a more common value for problems in which risk premia of various kinds are the focus. Furthermore, the latter also poses greater challenges for solution methods resulting from the very high curvature, so we would like to see how each method copes with those challenges.

Earnings Process. We assume that η (and consequently z) is normally distributed. Two values for ρ are considered: 0.9 and 0.98 and the unconditional standard deviation of $\ln z$ is set to 0.20 in both cases. We use the Rouwenhorst method for discretization over an 11-state grid (for z).

Borrowing Limit. The borrowing limit, $k_{\min}$, is set to a fraction ψ of the natural borrowing limit, which is defined as the loosest limit that still ensures full repayment of debt in the lowest earnings state $z_{\min}$. The resulting limit is $k_{\min} = -\psi z_{\min}/R$. We use $\psi = 0.6$ which allows up debt up to 60% of the natural limit. We also study $\psi = 0.1$ as another example.

Time Discount Factor and the Interest Rate. As this is a decision problem, we treat β as given and choose R such that the parameterized model generates a reasonable average wealth-to-income ratio. Average capital, $\bar{k}$, is obtained by simulating the model for 1.1 million periods using the derived policy function, and averaging the last one million periods.

15.1.2 Implementing the Solution Algorithms

We want to examine how a variety of choices impact the speed of convergence and accuracy of the numerical solution. Many of these choices and issues related to

TABLE I – Parameters and Choices for the Model and Numerical Methods

	Model Parameters		
Parameter	Description	Value(s)	Comment
γ	RRA	2 and 10	
ρ	Persistence of earnings	0.90 and 0.98	
σ_z	Uncond. standard dev. of z	0.20	
σ_η	Innovation standard dev.	$0.20 \times \sqrt{1-\rho^2}$	Gaussian
β	Time discount factor	0.95 and 0.99	
R	Interest rate	Calibrated for $\overline{k}/\overline{z} = 5$	
ψ	Borrowing limit (% of natural limit)	0.6 and 0.1	
	Choices for Numerical Solution		
N	Number of grid points	Experiment from 20 up	
k_0	Lower bound of wealth grid	$k_{\min} = -(0.6/R)z_{\min}$	
k_N	Upper bound of wealth grid	500	
Support of z	Discrete grid	11-states	Rouwenhorst
θ	Expansion exponent	3, 1.3, 1	
	Optimization method	Brent	
	Interpolation	Cubic-spline or piecewise-linear	
V_0	Initial guess for value function	$V_0(k_i, z_j) = U((1+R)k_i + z_j)$	
Stopping criterion	$\max_i\left(\left(V_i^{n-1} - V_i^n\right)/(1+\text{abs}(V_i^n)\right)$	$< 10^{-7}$	i indexes grid points

their impact have been previously discussed in detail in previous chapters. Here we review them briefly to refresh our memory and highlight particular aspects relevant for the problem at hand. The following options are what we consider.

Capital Grid. The lower bound for capital is almost always specified as part of the problem. The capital grid, which we shall denote G_k, is a sequence of $N+1$ monotonically increasing points: $G_k \equiv \{k_0, k_1, \ldots, k_N\}$. When considering k_N, there are two potential strategies. First, we can choose it such that $g(k_N, z) < k_N$ is guaranteed for all z. This may be desired if we wish to simulate the policy function and there is some positive probability that the maximum level of k may be reached. The problem with this approach is that it may require a large number of grid points to cover the entire range so as to maintain accuracy of the policy function. Alternatively, we could choose $\bar{k}$ such that $\exists z$ st $g(k_N, z) > k_N$, but where the probability of accumulating k_N is very low. This allows higher accuracy for the policy function at the more likely capital points, but might be problematic during the actual value function iteration steps. The reason is that if $g(k, z) > k_N$, then we have not evaluated $V(g(k, z))$, and have to extrapolate from the current data. Extrapolation can often lead to very bad estimates, preventing convergence.

Expanding Grid. An expanding grid is constructed as described in Chapter 8.4. We consider values for the expansion exponent, θ, of 3 (aggressive concentration near low values and expansion at higher end), 1.3 (moderate), and 1 (uniformly-spaced grid).[1]

[1] Selecting an appropriate curvature is closely related to the number of grid points and the range of the grid. For the examples in this chapter, with 300 grid points and $\theta = 3$, all cases converged. However, as the number of grid points increased, the curvature had to be reduced. The reason be-

Utility Function: CRRA vs CES As discussed in Chapter 8.5, an alternative way to write the standard CRRA utility function is in its CES form, which yields identical decision rules, but implies a value function that is (almost) linear even when markets are not complete. Using this alternative formulation, the Bellman objective in (15.1.1) would read

$$\mathcal{V}(k,z) = \max_{c} \left[(1-\beta) c^{1-\gamma} + \beta \mathbb{E} \left(\mathcal{V}(k',z')^{1-\gamma} \right) \right]^{1/(1-\gamma)}.$$

Unlike V, the new utility function, $\mathcal{V}$, is linear in k when the Merton-Samuelson theorem (Theorem 3.3) holds and very close to linear even when it does not hold, because it is effectively a transformation $\mathcal{V} = V^{1/(1-\gamma)}$. Previously we have shown this curvature reduction effect and argued that it makes interpolation very accurate even with few points. Therefore, in principle we can obtain a very accurate solution to the Bellman equation using a very sparse grid, which typically results in large speed gains.[2]

One goal of this chapter is to explore the trade-offs between the two formulations and quantify the size of such gains, if any, and the conditions that determine its effectiveness. Therefore, we shall solve the baseline model with both formulations and compare the findings.

Interpolation. We consider two ways of interpolating the value function off the grid points: piecewise linear and cubic spline interpolation. Cubic splines generate a smooth, continuously differentiable function, as compared to piecewise linear interpolation which simply generates a continuous function with kinks at the grid (or knot) points. Therefore, splines can potentially achieve convergence with fewer grid points, or considering the same number of grid points, faster and with greater accuracy. However, care must be taken with splines to ensure that grid points are not too far apart or too close together. As discussed in Chapter 8.3, if they are, spline might not generate a nicely monotonic function, and ripples can prevent the program from converging. Finally, because the CES formulation generates almost-linear value function, we would like to explore if it ever makes sense to consider piecewise linear interpolation with that formulation.

---

hind this was that $V^n(k(i)) \approx V^n(k(i+1))$, and it was possible that $V^n(k(i)) > V^n(k(i+1))$. Small ripples like this propagate through the value function for subsequent iterations, and can cause the value function to cycle between a series of values; this prevents convergence.

This issue is more apparent with the endogenous grid method, where it is essential that at each iteration, $V_k^n(k(i)) > V_k^n(k(i+1))$. Without this condition holding, the value function diverges relatively quickly.

[2] Recall that with the CES formulation, the derivative of the right hand side becomes more involved. See, again, Chapter 8.5 and especially footnote 7. For this simpler case $\partial RHS/\partial c = \frac{1}{1-\gamma}(RHS)^{\gamma/(1-\gamma)}[(1-\beta)(1-\gamma)c^{-\gamma} + \beta\mathbb{E}\left((1-\gamma)\mathcal{V}(k',z')^{-\gamma} \times \partial\mathcal{V}/\partial k'\right)]$.

Initialization. Even in this simple consumption-savings problem, the initial choice of V_0 can greatly affect the convergence time of the algorithm (and sometimes even whether or not the algorithm converges at all—e.g., in algorithms like policy iteration). While a surprisingly common choice for the initial value is the zero function, this is rarely a good choice. We set V_0 to the value in the last period of a finite horizon model where the individual spends all his resources at that date. This is a natural choice and incorporates the information of the grid size, value function curvature, among others.[3]

Maximizing the Bellman Objective. We use Brent's method as the optimization routine. Because maximizing the Bellman objective is a constrained optimization problem, choosing the bounds of the choice set is quite critical. These issues have been discussed in great detail in Chapter 12.3, so we do not repeat those considerations here.

Stopping Rules. The stopping criterion for the algorithms is to stop when the maximum percentage change in the value function between two consecutive iterations is 4×10^{-7} or less. Because the value function can get very close to zero (at the high end of the capital grid) we modify the percentage to read

$$\max_{i,j} \frac{\left|V^n(k_i, z_j) - V^{n-1}(k_i, z_j)\right|}{1 + |V^n(k_i, z_j)|} < 4 \times 10^{-7}, \tag{15.1.2}$$

where the $+1$ in the denominator ensures it is never too close to zero.

An often overlooked question is: At which points in the state space for V do we need to verify the convergence criterion? The default answer that is often assumed is that we check it on the grid points. While this usually works, we will see examples in a moment where it can lead to premature stopping and to subtle and hard-to-detect problems. For now, let us keep this point in mind.

Benchmark: The True Solution. To get the true solution for this problem, we solve it using the standard VFI algorithm with 3000 grid points, $\theta = 1$, spline interpolation, and a tighter convergence criterion of 10^{-8}.

One way to reduce this penalty is to use McQueen-Porteus bounds discussed above.

[3] An alternative choice would be to set the V_0 to labor income plus interest income (or payments). This corresponds to the case where the net savings rate is zero, whereas the benchmark formulation takes net savings rate to be –100%.

TABLE II – Interest rates and Borrowing Constraint

		Interest rate		$k_{\min}$		min(c)/mean(z)
ρ	β	2	10	2	10	2 and 10
0.9	0.95	0.050	0.030	–6.34	–10.48	0.212
	0.99	0.010	0.008	–32.56	–42.00	0.212
0.98	0.95	0.051	0.027	–6.29	–11.76	0.212
	0.99	0.009	0.005	–33.91	–61.42	0.212

Calibration with $K/W = 5$

15.1.3 Choice of VFI Algorithms

A key goal of the benchmarking exercise in this Chapter is to provide an evaluation of the relative strengths and weaknesses of different dynamic programming algorithms introduced in Chapter 14. In particular, we evaluate and compare: (i) the plain vanilla VFI method, (ii) VFI + Modified Policy Iteration Algorithm (MPIA), (iii) VFI + MacQueen-Porteus Bounds (MQP), and (iv) Endogenous Grid Method (EGM). Once we analyze these four methods, we shall also consider combinations of these methods

15.1.4 Check the Curvature

Before launching into the choice of numerical methods, let us take a look at some of the key parameters implied by this calibration. Table II reports the calibrated values of R for each parameterization, along with the implied borrowing limit, and $\tilde{c}_{\min}$, the *maximum consumption feasible in the worst state,* $(k_{\min}, z_{\min})$. It is easy to see that $\tilde{c}_{\min}$ is simply $(Rk_{\min} + z_{\min})$ and Table II reports it as a fraction of average earnings: $(Rk_{\min} + z_{\min})/\overline{z}$.[4] This consumption level corresponds to the case where the individual earns the lowest income level $z_{\min}$, is at the borrowing limit, and is rolling over the debt (i.e., he is only paying the interest, not the principal). Of course, the individual might actually consider *reducing* his debt at this state, which would imply even lower consumption. So, when we maximize the Bellman objective we will be evaluating $U(c)$ at consumption levels *below* $\tilde{c}_{\min}$. For example, for our parameter choices, the ratio of $\tilde{c}_{\min}$ to $\overline{z}$ is 0.21. This means if we normalize $\overline{z}$ to 1, and $U(c)$ is a CRRA utility function with $\gamma = 10$, our objective function will have a utility term that is around $-(0.21)^{-9}/9 \approx 140,000$. But suppose that we are not careful and search over a range of consumption values between 0.21 and, say, 0.05. The utility term at $c = 0.05$ will be 56.9 *Billion*. The slope at that consumption level will be $(0.05)^{-10} \approx 10.2$ Trillion! These are simply enormous numbers that

[4] We do this because these models are inherently scale free: we can always multiply everything with a positive scalar and scale the magnitudes up or down with no substantive changes. Such changes can easily have effects on the numerical performance as numbers we use in computers do have a scale (e.g., double precision, etc.).

will give any optimizer a hard time and make it very hard to keep significant digits intact and interpolations accurate.

So what to do? It turns out that the optimal choice of consumption at $(k_{\min}, z_{\min})$ is almost always to remain very close to the constraint, so we will rarely ever see an optimal consumption level lower than this. So when we are maximizing the Bellman objective, we should first check if this constraint $k' = k_{\min}$ is the optimum. If it is, then we are golden. We do not have to interpolate in the range of values with very large objective values, slopes, and curvature. This is very often the case, so we make sure to check this carefully first.

If, however, the corner solution is not optimal, then we should take a moderate sized step first—and always compute the magnitudes described above to see how big the numbers we are dealing with are. For example, with our calibration, it makes sense to search between say 0.1 and 0.212 before going even lower. It is important to never jump and try values $k' \gg k_{\min}$ as computations can easily get inaccurate in that range. A more detailed analysis of consumption choice in this model is presented in Chapter 18.1.

The main takeaway from this extended discussion is that we should always be aware of the various magnitudes implied by our parameter choices. The lowest end of the state space can pose great challenges unless we pay close attention to its implications. If, for example $\tilde{c}_{\min}$ is very low (say 0.05 or lower), it means all consumption values in the feasible range will imply a lot of curvature in the objective. Consequently, such models would require even more careful handling to obtain accurate solutions.

15.2 Ensuring Accuracy

Now that our algorithms have stopped with the desired maximum tolerance, can we go ahead and use the resulting solution—the decision rules and value functions—to address the substantive questions of our paper? Not so fast. We need to first ensure that the table of numbers that our computer just spit out actually is the true solution to our model. How do we do this?

The first lesson to learn about ensuring the accuracy is that *there is no silver bullet.* Checking for accuracy requires lots of careful detective work. And like in any good detective story, there are often lots of twists and turns: it is quite common for a solution that looks accurate from one angle to actually be wildly inaccurate for things that count for your analysis, and just as common for inaccurate-looking solutions to actually be more than adequate in other cases. The only way to ensure accuracy is to use your best judgement and not be swayed by results that look "good" and stop investigating alternative checks for accuracy. The better we appreciate this fact, the more likely our solutions can be ensured to be accurate.

Although there is no silver bullet, there are a number of good ideas that have been found to reveal commonly encountered types of inaccuracies. When speaking of accuracy it is useful to think of two classes of problems, with some overlap. The first class comprises the dynamic program and the accuracy of its solution; the second class covers general equilibrium models that we will cover in Part IV. While most of the methods discussed in this section apply to both types of problems, the latter raises some additional accuracy challenges that do not arise in partial equilibrium dynamic programming problems. Such issues are postponed until Chapter 19.3.

15.2.1 Essential Tests to Perform

I begin with a series of accuracy checks that are simple, yet very informative, and are therefore *absolutely required* before any proposed solutions to a dynamic economic model can be taken seriously.

I. How to Check for Convergence?

This sounds like a trivial question. After all we have already defined how we check for convergence. But as always the devil is in the details. When the state space is continuous, at which points do we check the convergence conditions? A natural answer seems to be: at the grid points. But this approach can yield very misleading and premature declarations of convergence. We will see a stark example of this in a moment. A better approach is to pre-specify a grid that is sufficiently fine and check convergence on all points in this finer grid. Of course, this means extra computational time. More on this in a moment.

II. Plot, Plot, Plot: A Picture is Worth A Thousand Words

There is no better way to do debugging. To put it bluntly, if somebody tells me that they solved a model and present a bunch of numbers but when asked say that they have not plotted the value functions and decision rules over their entire state space (taking slices when needed), I won't believe any of the numbers they present. There is no better way to verify that the solution is correct than to actually see it with your eyes. Some problems are immediately obvious once you *see* them.

For example, the theoretical results on dynamic programming covered in Chapter 13 (and some other results that may apply more directly to the particular problem at hand) establish qualitative properties of the solution. These may include the (global) strict monotonicity and strict concavity of the value function, the strict monotonicity of the decision rule, the differentiability of one or the other, among

others.[5] Therefore, the first thing to look for is to see if the numerical solution is visually consistent with these implications.

- ➢ First, plot the value function and decision rules at the grid points. If you see any signs of non-monotonicity of non-concavity for function values at the grid points, that already reveals a major problem. Second, using the same interpolation method as in your solution algorithm, find and plot the interpolated function values on a much finer grid. For example, if you are solving the model on a capital grid with 50 points, you should resample and interpolate at 1000 points. Your solution algorithm does precisely this resampling when maximizing the Bellman objective. Do you see any signs of non-concavity on the resampled functions? If you do, beware that your solution algorithm also saw the same strange pattern during the solution. Do not settle on one perspective.

- ➢ Because of scaling—for example, the value function might extend over a wide range of values—some problems might not be apparent at first blush. Therefore, it is useful to plot the functions both on a linear scale, but also on a log-log plot (log wealth versus log utility function, with the sign flipped if necessary) or on a semi-log plot. These plots could allow one to see more clearly what is going on.

- ➢ Furthermore, sometimes, wiggles and imperfections that are not visible to naked eye can be made visible by plotting the derivative of the function instead. (If you are wondering why we might care about such small imperfections, it will become clearer in Part V. For example, if the derivative is very jagged, this might prevent derivative-based optimizers from working well when we calibrate the model you solved numerically.)

Having said all of this, plotting the solution is a necessary step but is nowhere near sufficient. It will reveal obvious problems when present but many sinister problems could easily fly under the radar but can matter greatly for the final results. We shall see examples of this in a moment.

Finally, we can incorporate the verification of these properties—especially of monotonicity and concavity—into the convergence criteria of our VFI algorithm. This can often allow us to catch sneaky problems that may otherwise escape visual detection, but can create big problems for the properties of the solution. Examples of this will be studied momentarily.

[5] Of course, the computational method we choose, including the nature of the approximation techniques may invalidate some of these features. For example, using linear interpolation will lead to solutions that are non-differentiable at knot points. In such cases, those ramifications should be accounted for when interpreting the plotted numerical solutions.

III. Increase Order of Approximation(s); Recompute.

Perhaps the most effective approach to verifying accuracy is the following: once a solution is obtained with your ideal choice of approximation order—which may include the number of grid points used for endogenous and exogenous state variables, the accuracy used to compute the integration, the stopping tolerance for the optimizer, etc.—you should increase the order and re-solve the model. For example, if you used a 100-point grid on wealth direction, increase it to 300, if you used a 9-state process for shocks, increase it to 19 or 29, etc., and obtain a "finer" solution. If your original choice was sufficiently accurate, this finer solution should not produce any statistics (that you care about) that is appreciably different than the original one.

The advantages of this test—relative to simply plotting, for example—are twofold. First, it allows you to check for accuracy of the solution along dimensions you care about and in a way that you can easily quantify. For example, if your model is about consumption volatility and the original (coarse) solution yields a volatility of 5%, whereas the finer solution yields a volatility of 5.1%, as an economist you can judge if this discrepancy is acceptable or not. Some other tests do not always allow for such meaningful quantification.

A second advantage of this approach relates to a crucial difficulty in tests of accuracy. The difficulty is that when we check for accuracy either by plotting or through some of the more sophisticated methods outlined in a moment, it is not always clear if the inaccuracy matters, because it is not often clear if the part of the state space where the inaccurate solution lies is visited sufficiently often to matter. Experience has shown that some inaccurate looking solutions generate results that are indistinguishable from the true solution, because the inaccuracy happens to be in a region which is never visited.

But then this argument seems to suggest a simple remedy: check to see the probability that those parts of the state space are visited. As reasonable as this suggestion sounds, it often does not work reliably (and could give a false sense of security). The reason is that the distribution of probabilities across states is itself endogenous and depends on the decision rules themselves! For example, imagine that our solution for a consumption-savings problem is inaccurate such that it overstates the disutility of very low wealth levels. With this numerical solution, we are going to find that agents will avoid low wealth levels (more so than under the true solution) and we will not see too many agents with low levels. Then the argument above will lead us to conclude that the inaccuracy of the solution at the low end is not a problem because those states are not visited! If however, we were to obtain the true solution, those states would have been visited more often.

Increasing the order of accuracy and checking gets around this problem because if the original order was very low, the finer solution will generate a different probability

TABLE III – Convergence Time of VFI Algorithm, Four Methods

Method:	RRA	
	2	10
CES-spline	8.6	15.9
CES-linear	7.8	14.2
CRRA-spline	12.7	55.1(60)
CRRA-linear	14.8	20.4

Note: The baseline number of grid points is 20. If convergence is not obtained for this specification we increase grid points by 20 up to 100. CES formulation has integer exponent.

distribution over states, which could be caught in the statistics we compute. Here, again, we should plot some more. This time, we should generate a very long time series from the original solution and the finer one. Then we should inspect to see if they overlap (almost) perfectly, and if not if the discrepancy is ever large enough that might care about.

While this step is a necessary check of accuracy (without which no solution should be accepted as accurate), it only checks for accuracy issues arising from choosing too coarse a grid or an approximation with less than ideal accuracy. Although these issues are very common, there are some other types of problems that this approach may not detect.

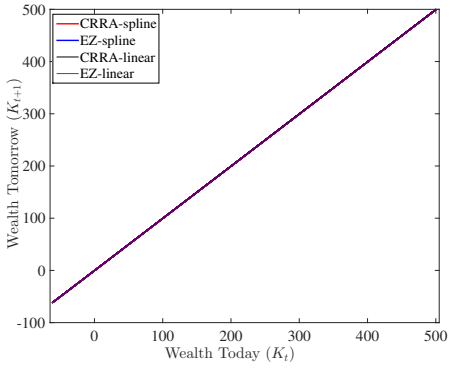
15.2.2 Checking Accuracy: An Application

Now let us apply the recipe detailed in the previous section to the solution of the canonical problem. In particular, we use the following parameters: $\rho = 0.98$, $\beta = 0.99$. For a quarterly model these numbers are typical for a persistence process like TFP or income. We choose the interest rate so as to generate an average $\overline{K}/\overline{W}$ of 5. We also set $k_{max} = 500$. We will test four set of methods for solving this problem: (i) CRRA specification, spline interpolation; (ii) CRRA specification, linear interpolation, (iii) CES specification, spline interpolation; and (iv) CES specification, linear interpolation. Now encouraged by the effectiveness of an expanding grid discussed earlier we set $\theta = 3$ and start with a grid that has only 20 points. If convergence is not obtained with this grid, we increase the number by 20 until convergence criterion (15.1.2) is satisfied.

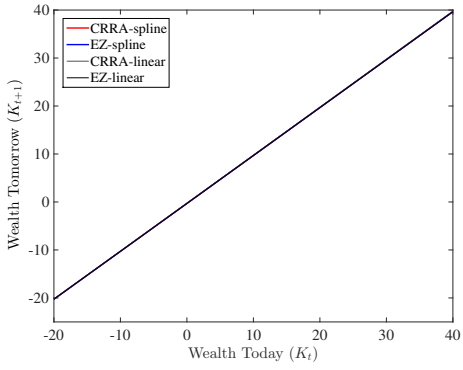
Convergence vs. "Convergence" According to the results in Table III, CRRA-linear converges with only 20 grid points in a comparable time to CES-linear, and slightly ahead of CES-spline. All three cases converge with only 20 grid points. CRRA spline on the other hand is the laggard and takes 60 grid points to converge at all, and takes about three times longer.

FIGURE 15.2.1 – Savings Decision Generated by Four Solutions of the Same Model

(A) RRA=10



(B) RRA=2, Zoomed in



How did CRRA-linear pull off this remarkable feat? This result, if confirmed, goes against much of the discussion in the previous sections that the CRRA formulation implies more curvature in the value function and thus requires more powerful interpolation—spline with more points—than the CES formulation. Maybe this is too good to be true, so let us check a bit more.

As suggested in the previous section, we begin by plotting the decision rules. Plotting $k'(k, z)$ for each value of z is what we want, but because there are 4 methods and 11 values for the z grid, plotting all 44 functions would take too much space. So, after reviewing all these 44 cases, in Figure 15.2.1a, I plot the case with the with the highest deviation between the decision rules across the four methods, which happens to be for $z = z_1$ (lowest income level). As you can see here, there is what appears to be a single linear decision rule, even though the figure plots all four decision rules. So perhaps we would be inclined to look at this figure and stop here, thinking that all models produce virtually identical decision rules. But maybe we are the more meticulous type and realize that the range of wealth values are very wide, so perhaps we should expand the graph by zooming in near the average capital level and take $\pm\sigma$ of the range in which assets fluctuate over time.[6] The wealth grid we are looking at now is one-tenth of the full range displayed in the left panel. In order to avoid plotting too many graphs, this time in Figure 15.2.1b I am plotting the decision rules for the $RRA = 2$ case, which should be more favorable to the CRRA-linear case since it poses much lower curvature than the case in the left panel. I have zoomed in to the asset range between -20 and 40. Again decision rules look indistinguishable.[7]

[6] Of course, we do not yet know σ since we have not simulated the model yet. But suppose we have a good guess.

[7] In my experience, when this question is assigned as a homework problem many student stop here and report back that the decision rule is very accurate because it is indistinguishable from more precise solutions.

But suppose we are the really picky type so we decide to take an even smaller interval, between $k \in [1, 10]$ and now plot tomorrow's asset holdings on a log scale, as in Figure 15.2.2, so as to express possible deviations in percentage deviation form. Now we see the first sign of, what appears to be, a small deviation. That is, for low asset holdings today ($a < 5$), the CRRA-linear solution diverges slightly from the other three—who still look indistinguishable from each other. One difficulty here—and with plotting decision rules in general—is that unless the problem is very obvious (such as oscillations, nonmonotonicity, and so on) it is not clear if the deviation seen in this figure is big or small.

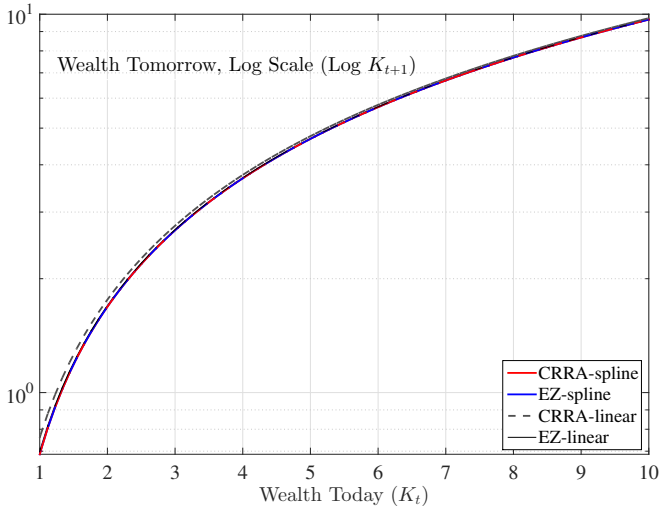
There is no room for uncertainty when we decide to accept a solution as accurate. Until we can prove otherwise, the default assumption about any proposed solution is that it is not accurate. With this in mind, we now move on to the next and more informative accuracy check.

We will now compare some key statistics implied by each solution to the true solution, obtained by solving the same problem with the minimal set of shortcuts: a 1000-point grid, spline interpolation, and very high required accuracy for stopping rule. We simulate each solution for 110,000 periods and discard the first 10,000 periods so as to isolate the potential effects of initial conditions. We use the last 100,000 periods to compute various statistics. Table IV reports the mean, standard deviation, min, and max of capital, consumption, and net savings ($K_{t+1} - K_t$). The first two rows of each panel reports the results with CES and CRRA formulations when spline interpolation is used. It is remarkable that even with 20 grid points, both solutions are very accurate—all statistics are within 1–2% of the true solution. This is thanks to the aggressive concentration of grid points with $\theta = 3$ and the high accuracy of spline interpolation.

The last two rows report the same two formulations but now with linear interpolation. The CES formulation overstates the average capital stock (13.34 vs 11.26 under the true solution), but otherwise generates all other statistics very accurately. This is quite remarkable given how few points are used with a simple linear interpolation. Clearly, this is due to the fact that the value function has little curvature with the CES formulation and most of that curvature is concentrated at the very low end, so concentrating points in that area allows us to capture that curvature without needing higher order interpolation.

Finally, we turn to the CRRA formulation with linear interpolation, reported in the last row of each panel of Table IV. As seen here, we were right to be suspicious: the solution is extremely inaccurate, with an average capital stock of 31.83 (almost three times its true value) and an average consumption of 1.33 (16% higher than the truth). Furthermore, the maximum capital is 80.78, compared with about the true maximum of 100, and the volatility of $K_{t+1} - K_t$ is about half the truth. Clearly, such a solution is not even close to being acceptable. So the small deviations

FIGURE 15.2.2 – RRA=2, Zoomed in Further, Semi-log Plot



we noticed in the decision rules turns out to matter greatly for the statistics we care about. Having said that, notice also that some other statistics, especially the standard deviation of consumption and capital are very accurate. Thus, if we had just focused on a few arbitrary statistics, there would have been a chance that we would accept the solution as accurate. This is why it is essential to compare a broad set of statistics under the true solution and the proposed one.

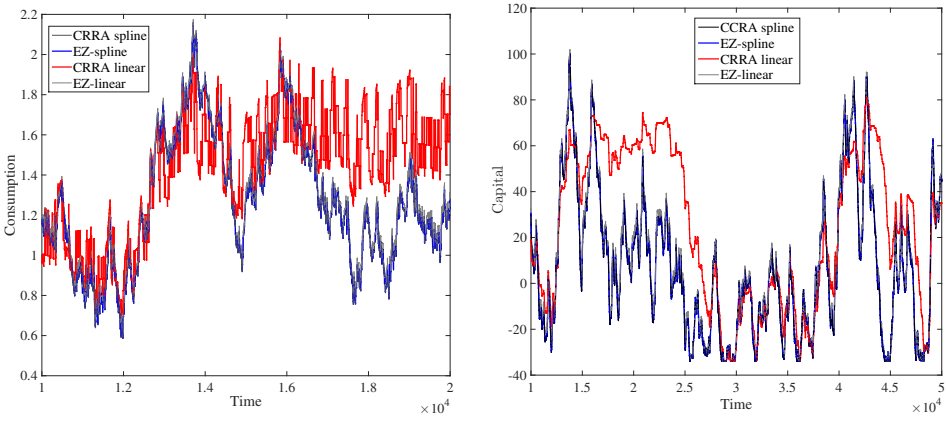
While these statistics are very useful and allowed us to weed out CRRA-L as an unacceptable solution, it is also useful to look at the time series of the data we simulated. Figure 15.2.3 plots the simulated time series of consumption and capital under the four solution methods. The first three solutions track each other closely. In fact, CES-S and CRRA-S overlap so well that they are not separately visible. CES-L is also very close, although at the very edges of the graph the thin grey line separates ever so slightly from the other two solutions. As for the CRRA-L, the nature of inaccuracy is quite interesting. In the left panel, the consumption path is not too far off the true solution between periods 10,000 and 13,800. However, after that period, the two solutions diverge and remain very different for long stretches.

However, this is not a permanent divergence, as the right panel makes clear. Here we plot the capital values but all the way up to period 50,000. As seen here, the capital paths completely diverge between periods 16,000 and 28,000 but then converge again from 28,000 all the way to period 40,000. The takeaway is that the CRRA-L solution has some inaccuracy in part of the state space that is not visited with high probability. But once that region is visited, the solution diverges and state variables seem to get "trapped" in that region that implies a capital stock much higher than the truth. This generates an excessively high capital stock and

TABLE IV – Statistics of Simulated Data Generated by Different Solution Methods

	Mean	Std.Dev.	Max	Min
		Capital		
CES-S	11.26	28.81	100.24	–33.912
CRRA-S	11.18	28.77	99.76	–33.912
CES-L	13.34	28.90	101.86	–33.912
CRRA-L	**31.83**	*28.66*	**80.78**	–33.912
		Consumption		
CES-S	1.14	0.325	2.16	0.213
CRRA-S	1.13	0.325	2.16	0.213
CES-L	1.15	0.326	2.17	0.213
CRRA-L	**1.33**	*0.326*	*2.14*	0.213
		$K_{t+1} - K_t$		
CES-S	0.001	0.132	0.611	–0.324
CRRA-S	0.001	0.132	0.613	–0.324
CES-L	0.001	0.130	0.609	–0.318
CRRA-L	*0.001*	**0.075**	**0.566**	**–0.233**

Note: The percentage of periods where k is at its lower bound under the three solutions is 0.029,0.029, and 0.01 respectively.



(A) Consumption Path (B) Capital Path

FIGURE 15.2.3 – Simulated Paths Generated by Four Solutions Methods, RRA=2

therefore a high consumption level. But later, again with low probability, it seems the solution leaves that region and moves again to parts of the state space where the solution is accurate.[8]

[8]How about the RRA=10 case? CRRA-L produced an average asset level of 169.1 compared with 11.41 under the true solution. The maximum asset level is also 217.99 compared with only 103.80 under the true solution. Consequently, consumption is also much higher, with an average of 1.91 compared with 1.09 under the true solution. Finally, net savings, $K_{t+1} - K_t$ is much less volatile, with a standard deviation of 0.067 compared with 0.154 under the truth.

There are several takeaways from this discussion. First, perseverance pays off. Relying on "convergence" of subsequent value function iterations is not close to being sufficient for ensuring accuracy. Plotting the solutions helps a little bit in this case, but only after we zoomed into various parts and saw what appeared to be small signs of divergence. Only after simulating data and comparing to the true solution were we able to tell that one of the solutions was, in fact, wildly inaccurate. Second, we do not always need to go as extreme with the true solution as we did in this example. Even if we had switched from piece-wise linear to spline interpolation, we would have immediately noticed that the statistics from the model changed dramatically and that would have led us to explore further. Third, in models with high persistence (here with $\rho = 0.98$ and $\beta = 0.99$) we must make sure to simulate for a long enough period. Here we did 100,000 periods, but that one could go even higher. If we had simulated for 5,000 or 10,000 periods, which just had coincided with the periods where CRRA-L happened to be accurate, again, we would not have realized how inaccurate the solution could be. Fourth, this analysis also speaks strongly in favor of the CES formulation, which is largely accurate even with linear interpolation. However, spline allows us to use so few grid points and at that level, spline is not much slower than linear interpolation. The slight improvement in speed (see Table III) is typically not worth the loss of accuracy and differentiability. Therefore, it is generally advisable to use fewer grid points concentrated at the areas with high curvature and apply spline interpolation, which is not much slower than piece-wise linear interpolation, but is much more accurate. The downside, of course, is the possibility of oscillations which we need to pay attention to.

15.3 Benchmarking VFI Algorithms

A good solution algorithm has three desirable features: speed, accuracy, and robustness. Different algorithms make trade-offs along these three dimensions and our goal is to find the one(s) that is (are) good along all three dimensions.[9]

In this section, we examine the computational speed of various alternative choices that make up the VFI algorithm. The analysis of accuracy in the previous section taught us some important lessons, the most important of which is that *we can not rely on value function convergence on a discrete grid as evidence of an acceptable numerical solution.* Therefore, the timing benchmarks in this section pays close attention to not only the time it takes for value function convergence but also to the time it takes to obtain an accurate and acceptable solution. To this end,

[9] A fourth key feature is the easy of coding/implementation, but the particular choices we consider in this section are relatively similar and all rather straightforward to implement. We shall focus on this aspect more in Part III for General Equilibrium applications where there are larger differences.

TABLE V – Convergence Time of VFI Algorithm

		$\gamma = 2$		$\gamma = 10$	
		CES	CRRA	CES	CRRA
		PANEL A: $\rho = 0.90$ and $\beta = 0.95$			
$\theta =$	1.0	$1.9^a/50.2^b$	6.5/68.8	3.6/66.7	191.4/191.4
	1.3	1.9/4.1	3.6/7.7	3.6/11.7	52.6/69.2
	3.0	1.9/1.9	1.8/3.6	3.4/3.4	7.4/15.7
		PANEL B: $\rho = 0.98$ and $\beta = 0.99$			
$\theta =$	1.0	9.3/143.8	9.6/187.0	17.4/326.0	454.4/454.4
	1.3	9.3/19.7	9.7/31.8	17.1/56.8	54.9/165.0
	3.0	8.6/8.6	9.3/9.3	15.9/15.9	38.1/38.1

Note: Convergence times are reported as a/b where a is the time when convergence is defined by value function discrepancy condition (15.1.2) being satisfied, and the final value function is monotonic in wealth; b is defined if in addition to the conditions in (a), the mean, standard deviation, and maximum of simulated paths for consumption, capital stock and savings is within 1% of the true solution for the given parameterization. The baseline number of grid points is 20. If convergence is not obtained for this specification we increase grid points: 20, 40, 60, 80, 100, 125, 150, 200, 250, 300, 400, 500, 600, 700, 800, 900, 1000. Both formulations have integer exponent.

we proceed in two steps. First, we report the timing for value function convergence for the following 4 parameterizations and 6 algorithmic choices obtained as follows:

Parameterizations: $((\rho, \beta), RRA) = \{\{0.9, 0.95\}, \{0.98, 0.99\}\} \times \{2, 10\}$

Methods: $(\theta, V) = \{1, 1.3, 3.0\} \times \{\text{CES, CRRA}\}$

In the following subsections, we shall compare the four main types of VFI algorithms introduced in Chapter 14: (i) the plain vanilla VFI method, (ii) VFI + Modified Policy Iteration Algorithm (MPIA), (iii) VFI + MacQueen-Porteus Bounds (MQP), and (iv) Endogenous Grid Method (EGM).

15.3.1 Case 1: Plain VFI

We begin with the plain vanilla VFI algorithm. Table **??** reports the various timing benchmarks organized as follows. Columns (1) and (2) display the case with low curvature, $\gamma = 2$, the columns (3) and (4) show the case with high curvature, $\gamma = 10$. For each parameterization, the table reports the results with CES curvature reduction and with standard CRRA formulation. Finally, each of the three rows in a given panel shows how the concentration parameter, θ, affects the timing and accuracy: from top to bottom, going from $\theta = 1$ to 1.3 and then to 3.0.

Each cell in Table V reports two different convergence times for a particular configuration. The one on the left is the time it takes when convergence is defined by two conditions: (i) the value function converges according to (15.1.2) and (ii) the

final value function is monotonically increasing in k. The timing on the right in each cell defines convergence by (i) and (ii) plus a third condition: (iii) we simulate paths for consumption, capital stock, and savings for 110,000 period and ensure that the mean, standard deviation and maximum of these three variables during the simulated path is within 1% of the true solution for the given parameterization. Therefore, the numbers on the right ensures not only that the value function converges but also that the implied behavior is accurate as compared with the true solution. Hence, the timing figure on the right in each cell is the relevant value and the figure on the left is provided for comparison (and illustration of the potential pitfalls involved in just checking conditions (i) and (ii)).

Here are several key takeaways from Table V:

A Concentrating grid point has an immense effect on the speed and accuracy of the VFI algorithm. Going from an equally-spaced grid ($\theta = 1$) to one with high concentration at the lower end ($\theta = 3$) results in an order-of-magnitude improvement (typically a 15- to 20-fold reduction in solution time) across all configurations. This gain is just as big when the curvature is low ($\gamma = 2$) as when it is high ($\gamma = 10$). Furthermore, even a very mild concentration ($\theta = 1.3$) helps substantially: there is a 5- to 10-fold gain for the CES formulation and 3- to 6-fold for the CRRA formulation. Because the CES formulation generates value functions that are very flat everywhere except the lower end, the improvement is larger in this case.

➢ The bottom line is concentrating grid points where there is high curvature is an essential element in VFI algorithm. However, I do not recommend going too far beyond $\theta = 3$, because this essentially entails grid points at the lower end that are too close to each other. This might cause some problems if the maximization tolerance is not very high, and might lead to non-convergence.

B The CES curvature reduction formulation is faster than the standard CRRA formulation for every parameter configuration and choices for θ. The speed up ranges from about 60% when the curvature is small ($\gamma = 2$) up to 3-fold when the curvature is high ($\gamma = 10$). For some other parameter configurations not included in Table V, we have seen speed ups as high as 10-fold for the CES formulation. Furthermore, even when we use too few points for the CES formulation to generate to the true solution, our experience has been that it is almost always significantly more accurate than the CRRA formulation.

➢ Given the ease of implementation of CES trick, it should be the default choice of modeling anytime it is feasible.

C With $\theta = 3$, the CES solution delivers very accurate solutions even when we only check conditions (i) and (ii) above. That is, when the value function converges according to condition (15.1.2) and is monotonic, the solution is also accurate as compared to the true solution. This is often not the case for the CRRA formulation (compare the left/right timing numbers to see this).

D What does checking for monotonicity do? To see this, let us define convergence by just (i)—condition (15.1.2). In my experience, it is very common for inexperienced researchers to simply check this condition to decide whether they have obtained the solution. But relaxing monotonicity can have a dramatic negative impact. For example, in the first row ($\theta = 1$) of top panel with $\gamma = 10$, the convergence time for the CRRA is 191.4 both when we check (i) and (ii), and when we add (iii). This solution requires 1000 grid points. However, if we were to just check (i), convergence would be declared with 20 grid points and in 3.9 seconds! Inspecting the value function we saw that it was very non-monotonic and in fact *positive* in some range of capital values. This is a completely nonsensical solution, but it does happen more often than one might expect. Many of the other cells in rows 2 and 3 declare premature convergence when monotonicity is not checked explicitly.

➢ Therefore, it is essential to check for monotonicity and concavity (which is weaker) of the value function and monotonicity of the decision rule (assuming theoretically we know this to be the case. To save time, you may want to not check for this in every iteration; instead check for condition (15.1.2) in each iteration, and when it is satisfied we can also check for monotonicity/concavity

15.3.2 Case 2: VFI with MPIA

Recall from Chapter 14.3 that policy iteration (and its modified variant) is equivalent to Newton-Kantorovich iteration (Puterman and Brumelle (1979)) and therefore inherits its somewhat temperamental global convergence properties. In this section, we explore the benefits and drawbacks of this method by applying to the same parameterizations considered in the plain vanilla VFI algorithm of the previous section. Based on the findings in the previous section, in what follows, we focus on the case with concentrated grid ($\theta = 3$) and spline interpolation. We fix the number of grid points to the values that delivered an accurate solution in Table V. We vary the number of modified policy iteration steps from 0 (corresponding to the plain vanilla VFI case as a benchmark) all the way up to 500.

Table VI reports the results. MPIA can result in substantial improvements—up to 8- to 10-fold—in convergence time. However, there are several catches.

TABLE VI – Convergence Time of VFI with MPIA

PI	$\gamma = 2$				$\gamma = 10$			
	CES		CRRA		CES		CRRA	
	PANEL A: $\rho = 0.90$ and $\beta = 0.95$							
0	1.87	336	3.60[a]	320	3.38	482	15.7	327
1	1.07	177	2.00[a]	168	1.87	253	8.6	171
5	0.46	64	0.90[a]	62	0.82	91	3.6	61
10	0.33	35	0.65[a]	36	0.52	48	2.8	37
25	0.24	18	0.47[a]	18	0.63	38	*	*
50	0.27	12	0.48[a]	12	*	*	*	*
100	0.36	10	0.67[b]	10	1.18	27	*	*
200	0.55	9	1.24[b]	10	1.41	18	*	*
300	0.81	9	1.79[b]	10	2.05	18	*	*
400	1.08	9	2.35[b]	10	2.69	18	*	*
500	1.32	9	2.91[b]	10	3.56	19	*	*
	PANEL B: $\rho = 0.98$ and $\beta = 0.99$							
0	8.60	1420	9.32	1634	15.88	2072	38.01[a]	1708
1	4.81	747	5.11	853	8.81	1092	20.86[a]	888
5	2.12	270	2.24	304	3.81	393	8.83[a]	321
10	1.48	154	1.54[a]	173	2.59	222	6.54[a]	201
25	1.04	70	1.08	78	1.97	112	4.71[c]	100
50	.88[a]	38	0.88[a]	42	*	*	8.38[d]	78
100	.82[a]	21	0.86[a]	24	*	*	10.49[d]	84
200	.96[a]	14	.99[a]	16	*	*	17.75[d]	78
300	1.04[a]	11	1.12[a]	13	*	*	25.35[d]	77
400	1.34	11	1.38	12	*	*	35.67[e]	82
500	1.51[c]	10	1.7	12	*	*	41.47[d]	77

Note: $\theta = 3$. The baseline number of grid points is the same as under plain VFI shown in the previous table. [a]Decision rule converges only to 10^{-6} accuracy. [b]Decision rule converges only to 10^{-5} accuracy. [c]Decision rule converges only to 10^{-4} accuracy. [d]Decision rule converges only to 10^{-3} accuracy. [e]Decision rule converges only to 10^{-2} accuracy.

A First, the improvement is U-shaped: the convergence time improves rather dramatically as we increase the PI iterations from 0 to 10; it continues to improve but at a much slower rate from PI iterations of 10 up to 25 to 50; convergence time starts to rise as we further increase the iteration number from 50 to 500. Notice that the number of VFI iterations (reported in the next column) declines monotonically, so the U-shape in overall timing is resulting from the fact that each PI step has its own overhead cost.

B The sweet spot for the PI steps is a bit higher in Panel B (when ρ and β are higher) and is a bit lower when the curvature, γ, is higher. For example, the CRRA formulation in the top panel and when $\gamma = 10$ crashes with PI steps

are 25 and higher. Similarly, the CES formulation crashes in the bottom panel when $\gamma = 10$ when PI is 50 or higher. Also notice that even though the corresponding CRRA formulation technically converges, the policy rule never converges to a very high accuracy (not exceeding 10^{-3} in some cases). This suggests that PI steps in excess of 25 should be viewed with lots of caution.

C The bottom line is that prudent choices for PI iterations are typically below 10 and most certainly below 25. It is possible to improve upon the figures reported here by implementing more sophisticated variants whereby the PI steps is small early on in the VFI iterations and is then increased as the value function starts to converge and is then again reduced to achieve higher accuracy very near the true solution. In my experience such algorithms require quite a bit of baby sitting and are not usually worth the additional headache.

To sum up, MPIA is a simple but critically important addition to the VFI algorithm. It almost never makes sense to use the plain vanilla VFI algorithm given MPIA is available. Even using 5 to 10 PI steps can result in an order of magnitude improvement in convergence time.

15.4 Checking Accuracy via Euler Equation Errors

Recall that the optimal solution do a dynamic programming problem must satisfy an appropriate Euler equation (or inequality)—although satisfying this condition is not always sufficient to be an optimum. However, the fact that this equation is necessary provides a general way to check if a proposed numerical solution is accurate, by checking if it satisfies various implications derived from the Euler equation. I first discuss the case where constraints are not present or not binding so that the Euler condition must hold as an equality, and then discuss the ramifications of binding constraints.

The simplest version of the Euler equation errors can be explained with the following example. A standard consumption-savings problem yields the following Euler equation:

$$1 = \mathbb{E}\left[\beta\left(\frac{c_{t+1}}{c_t}\right)^{-\gamma}\tilde{R}_{t+1}|I_t\right], \tag{15.4.1}$$

where $\tilde{R}_{t+1}$ is the gross (possibly stochastic) return on the asset that the consumer has access to. For the exact solution, this equation must hold exactly. Our numerical solution is then accurate to the extent it satisfies this condition. So the very first condition we can check is the unconditional version of (15.4.1). That is, simulate a

time-series of consumption (and returns if stochastic) and compute

$$\frac{1}{N}\sum_{n=1}^{N}\beta\left(\frac{c_{t+1}}{c_t}\right)^{-\gamma}\tilde{R}_{t+1}\approx 1.$$

Simulation size N should be large enough that the law of large numbers can be assumed to hold. With today's computers this is no longer a problem, so choose it liberally (at least 100,000 periods) and discard the first 10% or so of observations to prevent initial conditions from imposing undue influence. Although this is a quick and easy check, the deviations from 1 is not straightforward to interpret. In other words, how small a deviation means that the solution is accurate?

15.4.1 Euler Equation Errors

To get a scale independent measure, we can proceed as follows. Suppose that we have solved our model and obtained a savings rule $k' = \hat{g}(k, z)$. The Euler equation can be written more generally in terms of savings rule and constraints as follows:

$$U'(\tilde{c}(k,z)) = \mathbb{E}_z\left[\beta(U'((1+R)\hat{g}(k,z)+z'-\hat{g}(\hat{g}(k,z),z'))R\right]. \tag{15.4.2}$$

Thus, given a proposed solution $\hat{g}(k)$ we can obtain the consumption function in two ways. First, the budget constraint implies that

$$\hat{c}(k,z) \equiv (1+k)R(z) - \hat{g}(k,z).$$

Second, plugging $\hat{g}(k, z)$ into the Euler equation (15.4.2) delivers another solution for consumption, $\tilde{c}(k, z)$, on the left hand side. If the proposed solution is perfect the two consumption functions should coincide.[10] If not, the discrepancy gives us an error measure for the accuracy of the solution:

$$\mathcal{R}(k,z) \equiv \frac{\tilde{c}(k,z)}{\hat{c}(k,z)} - 1.$$

This measure is interpreted as the percentage change in consumption we need in state (k, z) to have our solution satisfy the Euler equation exactly. With this definition at hand, we can evaluate this measure of error on a fine grid (k_i, z_j) to ensure it is small everywhere.

While this exercise is useful, it might paint a picture that is too bleak. This is because the solution might be inaccurate in some parts of the state space which are rarely visited. Therefore, another measure would take the two measures of

[10] The idea of expressing Euler equation errors in consumption equivalent terms first appears in Christiano and Fisher (2000).

consumption and simulate a time series of Err for a long period of time. This alternative look would tell us whether the solution is accurate along the simulated path. We can then examine the statistics (min, max, standard deviation, and so on, of Err) to verify the accuracy. Having said that, recall our previous discussion that following the simulated path runs the risk that the path itself is endogenous and depends on the very solution whose accuracy we are trying to verify. Thus, ideally, we would require the verification over the entire state space to be satisfied.

15.4.2 Marcet-Den Haan Test

It is possible to push the idea of the previous accuracy test one step further. To see how, suppose we simulate a long time series from our model and denote it with $\{\hat{c}_t, \hat{k}_t\}_{t=1}^T$. Then, if our solution is accurate, this simulated sequence must satisfy the Euler equation (15.4.2), which can be written without the expectation term by introducing the forecast error, denote it with ε_{t+1}, to read:

$$U'(\hat{c}_t) = \beta R \times U'((1+R)\hat{k}_t + z_{t+1} - \hat{k}_{t+1}) + \varepsilon_{t+1}. \tag{15.4.3}$$

In this formulation, there is no presumption that $\varepsilon_{t+1} \equiv 0$ for the correct solution, because there is no longer an expectation term. This forecast error will be nonzero as long as there are unforeseeable "shocks" as of period t. This is where the stricter test comes into the picture. Rational expectations imply that if the solution is consistent with the Euler equation, then the forecast error at $t+1$ must be orthogonal (uncorrelated) with all information known in period t and before. Thus, we can use equation (15.4.3) to generate a path for $\{\varepsilon_{t+1}\}_{t=1}^T$ and then run the following regression:

$$\hat{\varepsilon}_{t+1} = a_0 + a_1 x_{1t} + a_2 x_{2t} + ...a_n x_{nt} + \nu_{t+1},$$

where $\{x_{1t}, x_{2t}, ..., x_{nt}\}$ are variables that are known as of period t. The unforecastability condition implies that all coefficients are jointly zero: $a_0 = a_1 = ... = a_n \equiv 0$. If this condition is violated then the solution violates the Euler equation, indicating an inaccurate solution. Den Haan and Marcet (1994) have proposed this as a stricter version of the accuracy test just described. **[[Elaborate on how to implement this with a finite sample]]**

15.5 Tips: Programming Practices

Exponents: Use Integer Values Whenever You Can. Table VII shows why.

TABLE VII – Convergence Time of VFI Algorithm, CES with Real or Integer Exponents

Exponent:		2.0/2	10.0/10
ρ	β		
0.9	0.95	1.31	1.97
	0.99	1.02	1.51
0.98	0.95	1.23	1.86
	0.99	1.06	1.94

Note: The baseline number of grid points is 20, and $\theta = 3$. Each cell report the relative time it takes for VFI to converge when the exponent is a real number relative to the baseline where it is the same number but coded as integer.

Avoid using hard numbers. Your code should be written such that if a coauthor asked you to solve the model again with two times the number of grid points for a given state variable, or twice the number of ages in a lifecycle model, or twice the number of shock values in your Markov chain, you should be able to accomplish this by changing one number in only one place in your entire code. This is very often possible if you adopt good coding practices.

Version tracking is crucial. This is a seemingly trivial point, but so often new programmers struggle to find an earlier version that worked perfectly when the new version gives a headache and unable to do so. So when you make changes to your code, save every new version of the code that is substantially changed under a different file name—*do not overwrite existing codes continually.* Even if changes are not substantial, I make a habit of saving the code at the end of the day under a new name. Even better, use a code management repository, such as Github or Subversion, which allows automatic version tracking and revision history.