Dynamic Programming

Alexis Anagnostopoulos
Stony Brook University

1 Introduction

We have seen how to obtain conditions for optimality in an infinite horizon maximization problem, specifically conditions to solve for infinite sequences of consumption and capital in the Cass-Koopmans model. But we have not yet discussed solution methods that can be used to solve those conditions in order to obtain the maximizing choices. One approach is to solve that sequence problem by directly attacking the difference equations. In some cases (see assignments), a trick can be used to solve a non-linear difference equation analytically. Typically, however, non-linear difference equations will not admit a closed form solution and we will need to resort to approximations. Often, researchers will obtain a linear approximation to the non-linear difference equations and then use well-known methodologies for solving the resulting linear difference equations. We will discuss this approach later in the course. An alternative approach is to obtain a completely different formulation of the original maximization problem and use dynamic programming. Fundamentally, this will require writing the original maximization problem in a recursive form and switching from searching for optimal sequences to searching for optimal functions. This is the focus of this part of the course.

Dynamic programming will give us a method to solve for a wide range of dynamic problems in economics. It also allows us to obtain powerful results regarding the properties of the solution, including existence and uniqueness. But the recursive approach does not only provide a solution method. It will provide a way of thinking about dynamic problems that is quite intuitive. Indeed the vast majority of macroeconomists use these ideas to understand and explain their work. In addition, recursive methods will provide a more succinct way to present an economic model

and, importantly, will make it much easier to extend our models to the inclusion of uncertainty. Finally, for the typical cases where closed form solutions will not be available, a recursive formulation will allow us to numerically solve for the optimal functions on a computer.

We will focus mostly on the ideas and the practical implementation and leave the rigorous proofs aside.[1]

1.1 An Analogy with Decision Theory

Before delving into dynamic programming and it's application to the infinite horizon Cass-Koopmans model, it is useful to recall the backward solution of problems involving sequential decisions from your undergraduate decision theory (or game theory) classes. A very simple example of a three-period decision tree is provided in Figure 1.

The decision maker is sitting at the $t = 0$ node (A) and has to decide on the whole current and future plan of up (U) or down (D) decisions so as to maximize the sum of the one period returns (the numbers above each segment) resulting from those decisions. An optimal plan will be a sequence of choices for this three period problem, for example $\{U, U, D\}$. This is analogous to the optimal choice in a three period Cass-Koopmans model which will look like $\{k_1, k_2, k_3\}$.[2] Similarly, the objective is analogous to the three period Cass-Koopmans model where the 'one period returns' were obtained by plugging the choices into the discounted period utility $\beta^t u(c_t) = \beta^t u(f(k_t) + (1-\delta)k_t - k_{t+1})$ and then summed up over time to yield the overall objective of the planner. The simplified example given here differs from the Cass-Koopmans model in several dimensions. For example, here the decision maker can choose out of a discrete choice set (U or D) at each node, whereas in the Cass-Koopmans they could choose out of an interval of feasible k_{t+1}'s. Moreover, the possible choices here are always the same (U or D) whereas in the Cass-Koopmans model the choice set depended on the available capital k_t so it was different potentially at every node of the tree. But these differences are not central to the main analogy being built in this section.

One option for solving the decision problem is to obtain a complete enumeration of the objective (the sum of period returns) for every possible path and then simply choose the path that yields the maximum

[1] The classic reference for dynamic programming is "Recursive Methods in Economics Dynamics" by Stokey and Lucas. Section 5 below provides a guide through Chapters 3 and 4 of that textbook which are the relevant ones for this part of the course. For students interested in pursuing a PhD, it is strongly suggested that you read these carefully.

[2] For convenience, I am assuming we have substituted out consumption and investment so that the only choice left is capital.

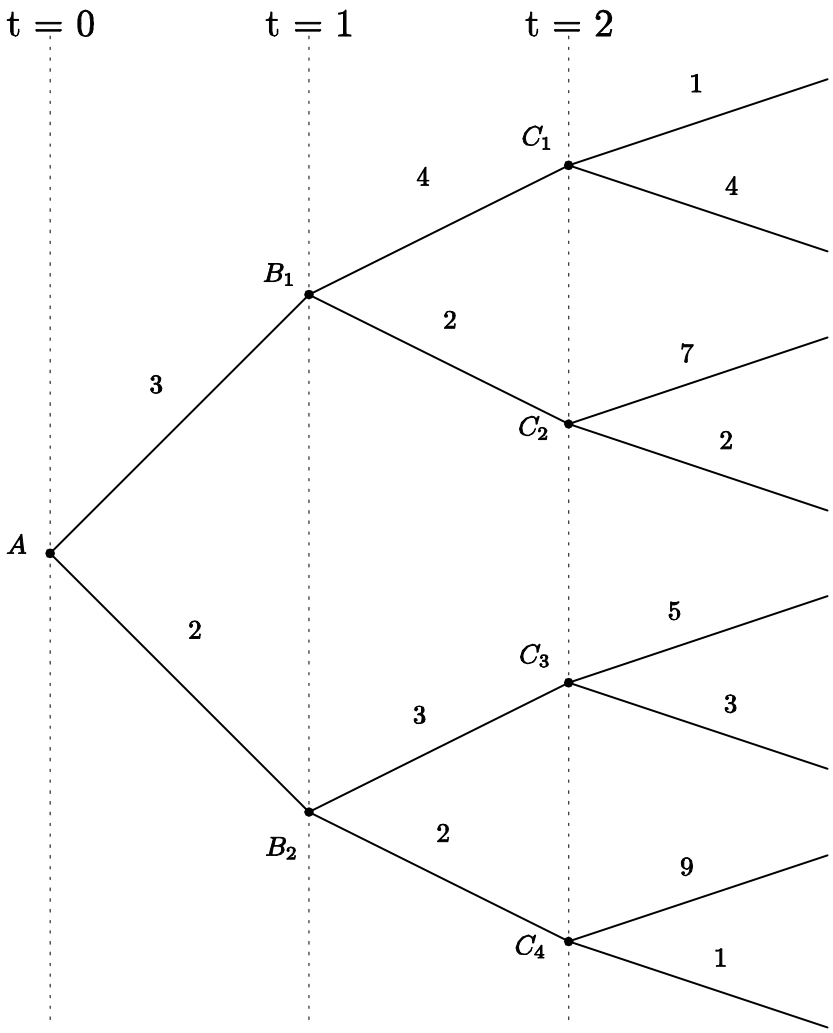

Figure 1: A decision tree example.

value of the objective. For example, choosing $\{U, U, U\}$ would yield $3 + 4 + 1 = 8$, choosing $\{U, U, D\}$ would yield $3 + 4 + 4 = 11$ etc. Although this is straightforward in the simple example, this option is not available in the Cass-Koopmans model for several reasons. To start with, the choice set for k_{t+1} is not just a collection of discrete values, rather it is an interval so complete enumeration of all choices is not possible. More importantly, this will not be helpful when we move to considering the infinite horizon version of the problem.

An alternative approach is to work backwards. Starting in the last period, choose *at every node* either U or D depending on the one period return resulting from that. So at C_1 the choice would be D (since 4>1), at C_2 the choice would be U (since 7>2) and so on. This defines the maximum return that the decision maker can obtain from that node onwards. Call this V_1 (because it is the max return to a one-period problem) and note that there are 4 nodes at $t = 2$ so there are 4 values V_1^i, $i = 1, 2, 3, 4$ one for each node. Clearly $V_1^1 = 4$, $V_1^2 = 7$, $V_1^3 = 5$ and $V_1^4 = 9$. We can then move backwards to the B nodes at $t = 1$. We can evaluate again, at every node at $t = 1$, whether U or D is the best option by adding the one period return to the 'continuation value' V_1^i and choosing the option with the highest sum. Again this defines the maximum remaining return that the decision maker can obtain from that node onwards, which we call V_2^j (because it is the return to a two period problem), $j = 1, 2$. For example, at B_1 choosing U would yield $4 + V_1^1 = 8$, choosing D would yield $2 + V_1^2 = 9$ and therefore it's best to choose D and the corresponding value at B_1 is $V_2^1 = 9$. Similarly, at B_2 the best choice is D and $V_2^2 = 2 + V_1^4 = 11$. Finally, we can do the same at the initial node, i.e. evaluate the two options (U or D) at node A on the basis of the sum of the one period return plus the continuation value V_2^j. At node A, choosing U would yield $2 + V_2^1 = 12$ and choosing D would yield $2 + V_2^2 = 13$. Therefore the optimal choice at $t = 0$ is to choose D. In this way, we have reduced the problem of choosing the optimal sequence to a number of simple up or down choices and have obtained the optimal sequence ($\{D, D, U\}$ and the overall maximized value $V_3 = 14$ for the three-period problem facing the decision maker at the time of maximization $t = 0$.[3]

This backward induction approach gives the correct answer because

[3]Notice that going backwards is crucial for this reduction to many simple problems; going forward wouldn't work. At $t = 0$, we would have the one period returns from U or D but would not know the 'continuation value' unless we went ahead and considered all possible future paths too. That is, we would not be able to break down the problem to a series of simple choices and would just need to follow the complete enumeration approach discussed before.

of a version of Bellman's *principle of optimality*, which states

An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.[4]

The idea is the following: Take the optimal sequence and consider following its path up to any of the nodes of the tree. Could it be you can improve upon that choice by changing the decisions from then on? Consider for example any of the C nodes at $t = 2$. Could it be that in the last U or D decision the optimal sequence chooses the option with the lower one period return? The answer is no, because then it would not be the optimal one, you could always increase the maximized value of the objective by choosing the option with the highest one period return. This logic applies to any node visited by the optimal path. From a mathematical perspective, the idea is simply that

$$\max_{x,y} f(x,y) = \max_{x}\{\max_{y} f(x,y)\}$$

i.e. you can solve a maximization problem with two (or more) variables x, y by first choosing one of them (y) for *any* value of the other (x) and then taking the resulting function of x (in this case $V(x) = \max_y f(x,y)$) and maximizing by choice of that variable.[5]

2 Finite Horizon Cass-Koopmans

Let us apply the principle of optimality and use it to solve backwards the two period Cass-Koopmans model. The sequence problem at time zero is as follows

$$\max_{\{c_0,c_1,k_1,k_2\}} [u(c_0) + \beta u(c_1)]$$

s.t.

$$\begin{aligned} c_0 + k_1 - (1-\delta)k_0 &\leq f(k_0) \\ c_1 + k_2 - (1-\delta)k_1 &\leq f(k_1) \\ k_1 \geq 0, k_2 \geq 0, c_0 \geq 0, c_1 &\geq 0 \\ k_0 \text{ given} & \end{aligned}$$

Suppose we approach this using the backward solution algorithm. The first step is to find the optimal choice in the last period, given any amount of capital left over from period $t = 0$, i.e. given any k_1. In the

[4] See Bellman, 1957, Chap. III.3.

[5] This is true under conditions which are very mild, essentially boiling down to the maxima being well defined, see the discussion in section 5 and the corresponding parts of the Stokey and Lucas textbook referred to there.

last period, the planner has to choose c_1, k_2 to maximize $u(c_1)$ subject to the resource constraint

$$V_1(k_1) \equiv \max_{c_1,k_2} u(c_1)$$

s.t.

$$\begin{aligned} c_1 + k_2 - (1-\delta)k_1 &\leq f(k_1) \\ k_2 &\geq 0, c_1 \geq 0 \\ k_1 &\text{ given} \end{aligned}$$

Note that we have defined the maximized value of the problem as $V_1(k_1)$. This is the value of the objective once the optimal choice has been computed and substituted back into the objective. This is why we define $V_1(k_1)$ as equal to the MAX of $u(c_1)$ subject to constraints, we do not define it as equal to $u(c_1)$. We indicate that this maximized value depends on k_1, it is a *function* of k_1.[6] We will call it a *value function* and use the subscript 1 to indicate that this is the value function of a one period problem. This maximization problem will also deliver optimal choices for c_1 and k_2 which will also depend on k_1, i.e. they are also functions. We will call those *policy functions* and denote them by $g_1(k_1)$, $g_1^c(k_1)$ where once again the subscript 1 indicates that these are the policy functions arising from a one-period problem. It is straightforward to use the Lagrangian approach to obtain conditions for maximization and to solve them to obtain the solution for any k_1, i.e. the value function and the policy functions

$$\begin{aligned} k_2 &= g_1(k_1) = 0 \\ c_1 &= g_1^c(k_1) = f(k_1) + (1-\delta)k_1 - g_1(k_1) = f(k_1) + (1-\delta)k_1 \\ V_1(k_1) &= u(f(k_1) + (1-\delta)k_1) \end{aligned}$$

The choice of capital in the last period is optimally equal to zero *for any* k_1, hence the policy function for capital is the zero function. It just happens that it is independent of k_1. The policy function for consumption and the value function both depend on k_1.

As a second step, the planner can move one period back to the $t = 0$ problem. The objective at $t = 0$ is $u(c_0) + \beta u(c_1)$ but we already know how any choice of k_1 today will affect the second component. That is , we can write the objective only in terms of the current choices c_0, k_1

$$V_2(k_0) = \max_{c_0,k_1} [u(c_0) + \beta V_1(k_1)] = \max_{c_0,k_1} [u(c_0) + \beta u(f(k_1) + (1-\delta)k_1)]$$

[6] The maximized value also depends on parameters such as δ as well as any parameters involved in the utility and production function, but we suppress this dependence.

s.t.

$$\begin{aligned} c_0 + k_1 - (1-\delta)k_0 &\leq f(k_0) \\ k_1 \geq 0, c_0 &\geq 0 \\ k_0 \text{ given} & \end{aligned}$$

Once again this problem can be solved using standard methods (obviously we would need specific assumptions on u and f and typically we would not be able to obtain a closed form solution) and it will yield a value function for the two period problem $V_2(k_0)$ as a function of k_0 and policy functions determining $k_1 = g_2(k_0)$ and $c_0 = g_2^c(k_0)$ as functions of k_0. At this point we have solved the problem backwards. The solution is a collection of functions $\{g_1(.), g_2(.), g_1^c(.), g_2^c(.), V_1(.), V_2(.)\}$. We can use those to also obtain the sequence of optimal choices $\{c_0, c_1, k_1, k_2\}$ given any initial k_0 as follows

$$\begin{aligned} k_1 &= g_2(k_0) \\ c_0 &= g_2^c(k_0) \\ k_2 &= g_1(k_1) = g_1(g_2(k_0)) \\ c_1 &= g_1^c(k_1) = g_1^c(g_2(k_0)) \end{aligned}$$

What if we were interested in solving a 3-period model backwards? We could start again from the last period and go backwards step-by step. But note that we have already done most of the work. We know already the solution to a two period model, it is described by the functions $\{g_1(.), g_2(.), g_1^c(.), g_2^c(.), V_1(.), V_2(.)\}$. We can instead use this information to just go one more step backwards. The only thing we need to be careful about is at which point to evaluate all these function. In this case, the planner is facing a two-period problem at $t = 1$ given k_1 (contrast this to before where the planner faced a two-period problem at $t = 0$ given k_0). So we need to simply evaluate $g_2(.), g_2^c(.), V_2(.)$ at k_1, which is the inherited capital at $t = 1$. To solve the problem at $t = 0$ (which is now a 3-period problem) we would do the following

$$V_3(k_0) = \max_{c_0, k_1} [u(c_0) + \beta V_2(k_1)]$$

s.t.

$$\begin{aligned} c_0 + k_1 - (1-\delta)k_0 &\leq f(k_0) \\ k_1 \geq 0, c_0 &\geq 0 \\ k_0 \text{ given} & \end{aligned}$$

More generally, define $V_j(.)$ to be the value function (the value of the objective *at the maximum*) when there are j periods left or, equivalently,

the value function of a j period problem. Also define $g_j(.)$, $g_j^c(.)$ to be the policy functions for capital and consumption respectively when there are j periods left or, equivalently, the policy functions of a j period problem. It should be clear that for any T-period sequence problem, we can write it as

$$V_T(k_0) = \max_{c_0, k_1} [u(c_0) + \beta V_{T-1}(k_1)] \tag{1}$$

s.t.

$$\begin{aligned} c_0 + k_1 - (1-\delta)k_0 &\leq f(k_0) \\ k_1 \geq 0, c_0 &\geq 0 \\ k_0 &\text{ given} \end{aligned}$$

Following this approach, we have reduced the original sequence problem to a collection of simple problems where the choice is between consuming today and saving for tomorrow. These problems are simple to solve, provided we know the function V_{T-1}, which means we will need to start in the last period (where $V_0 = 0$ obviously), find V_1 and work our way backwards recursively.

3 State variables

Value functions and policy functions are written as functions of only the capital stock chosen in the previous period. Why are they not also functions of consumption chosen in the previous period? Or of the capital stock chosen two periods before? The answer is simple: in this model, whether the planner has a lot of capital at some period t because they started (at $t = 0$) with very little and saved a lot until t or because they had a lot of capital to start with at $t = 0$, is irrelevant. The history of how the planner ended up with a particular value of k_t at time t is not relevant for the decisions at t (or from then on). The same is true about past consumption choices, they are not relevant for the current and future decisions. This is because the set of feasible choices for c_t, k_{t+1} only depends on k_t (since available resources are $(1-\delta)k_t + f(k_t)$) and both current utility and future utility do not depend directly on any other past choice like k_{t-1} or c_{t-1}. Another way to state this is that all the information that is needed in order to make the consumption/savings decision at any point in time t is summarized in the inherited capital stock. We say that capital is a *state variable* in this problem. This is, of course, very particular to this specific model. Determining the state variables in any given model is a crucial skill that you will need to develop with some practice.

Actually, it is not true that *all* that is needed is the capital stock in this finite horizon setting; the number of periods left is also relevant

information. How much of the available resources is optimally consumed currently and how much is saved for the future also depends crucially on how long that future is. We have indicated this dependence by indexing value and policy functions by the subscript j, the number of periods left. As a result, the exact value of t, the time period at which the 'current' consumption/savings decision is being considered, matters. When the time period t matters for the optimal decisions, we say that the problem is *non-stationary*. The finite horizon problem is non-stationary, but the infinite horizon model will be stationary. The definition of stationarity follows.

4 Infinite Horizon and Stationarity

Consider a problem whose state variable is s_t. In the example considered here $s_t = k_t$ but, more generally, s_t can be a vector (see examples below). The problem is stationary if

$$s_t = s_j \Rightarrow s_{t+1} = s_{j+1} \text{ for all } t, j$$

What this says is that time doesn't matter. If at any two points in time t, j the state variable (or vector) takes the same value, then the decision will be exactly the same and, hence, the state variables will also coincide in periods $t+1$ and $j+1$. Note that this implies also $s_{t+2} = s_{j+2}$, which implies $s_{t+3} = s_{j+3}$ and so on. The problem facing the planner in period t is identical to the problem facing him in period j, provided the state variable takes the same value at t and j.

This property of stationarity of a problem does not hold for a finite horizon problem, but it will hold for the infinite horizon version of the Cass-Koopmans model. One way to understand why this is the case, is by noticing that in an infinite horizon problem, the number of periods left at any point in time is still infinite. The great thing about stationarity is that, whereas in the finite-horizon version of the model we had to compute several value functions and policy functions (V_1, V_2, V_3 etc.), in a stationary problem all of those will coincide since in all periods there are an infinite number of periods left and hence we will need to find V_∞, g_∞ and g^c_∞, i.e. only one value function and two policy functions that are the same in all periods. You can already plug in $T = \infty$ in the problem defined in (1) to notice that you will have the same function V_∞ on both the left and right hand side (but evaluated at different points k_0 and k_1 respectively).

Here's how to see this mathematically. Let the value function be denoted by $V(.)$. In order to economize on notation, I will drop the subscript ∞ and will also substitute out consumption. At time $t = 0$,

the inherited capital is k_0 and the value is defined as

$$\begin{aligned} V(k_0) = & \max_{\{k_{t+1}\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t u(f(k_t) + (1-\delta)\, k_t - k_{t+1}) \\ & s.t. \\ & 0 \leq k_{t+1} \leq f(k_t) + (1-\delta)\, k_t \quad \text{for } t = 0, 1, 2, ... \end{aligned}$$

where the first inequality is capital non-negativity and the second one is consumption non-negativity. The sum in the objective can be decomposed into the current ($t = 0$) utility and all the rest

$$\begin{aligned} & \sum_{t=0}^{\infty} \beta^t u(f(k_t) + (1-\delta)\, k_t - k_{t+1}) \\ = & \, u(f(k_0) + (1-\delta)\, k_0 - k_1) + \sum_{t=1}^{\infty} \beta^t u(f(k_t) + (1-\delta)\, k_t - k_{t+1}) \\ = & \, u(f(k_0) + (1-\delta)\, k_0 - k_1) + \beta \sum_{t=1}^{\infty} \beta^{t-1} u(f(k_t) + (1-\delta)\, k_t - k_{t+1}) \end{aligned}$$

Therefore

$$\begin{aligned} V(k_0) = & \max_{\{k_{t+1}\}_{t=0}^{\infty}} \left[u(f(k_0) + (1-\delta)\, k_0 - k_1) + \beta \sum_{t=1}^{\infty} \beta^{t-1} u(f(k_t) + (1-\delta)\, k_t - k_{t+1}) \right] \\ & s.t. \\ & 0 \leq k_1 \leq f(k_0) + (1-\delta)\, k_0 \\ & 0 \leq k_{t+1} \leq f(k_t) + (1-\delta)\, k_t \text{ for } t = 1, 2, 3, ... \end{aligned}$$

Now use the idea that the maximum can be computed sequentially (the principle of optimality) to write as

$$V(k_0) = \max_{k_1} \left[u(f(k_0) + (1-\delta)\, k_0 - k_1) + \beta \max_{\{k_{t+1}\}_{t=1}^{\infty}} \sum_{t=1}^{\infty} \beta^{t-1} u(f(k_t) + (1-\delta)\, k_t - k_{t+1}) \right]$$

where the maximization inside the brackets is done given k_1 and subject to $0 \leq k_{t+1} \leq f(k_t) + (1-\delta)\, k_t$ for $t = 1, 2, 3, ...$ and the maximization over k_1 is done given k_0 and subject to $0 \leq k_1 \leq f(k_0) + (1-\delta)\, k_0$. The crucial part is to notice that the maximization inside the brackets is a problem that is identical to the one we started with, with the only difference being that k_1 is given as the initial capital instead of k_0.[7] Put

[7] A renaming of the variables might help you to see the equivalence. Let us call $x_t = k_{t+1}$. Then the maximization in the brackets can be written as

differently, all this maximum subject to constraints is simply equal to $V(k_1)$ and thus

$$V(k_0) = \max_{k_1} \left[u\left(f(k_0) + (1-\delta) k_0 - k_1 \right) + \beta V(k_1) \right]$$

We now have the *dynamic programming formulation*. Given that we could do the same for any period (not just $t = 0$) and time doesn't matter, it makes sense to altogether drop the time notation (the subscripts) and write

$$V(k) = \max_{k' s.t. 0 \leq k' \leq f(k)+(1-\delta)k} \left[u\left(f(k) + (1-\delta) k - k' \right) + \beta V(k') \right]$$

where k denotes the inherited (given) capital and k' denotes the capital chosen today. This is the *Bellman equation*. It involves a simple univariate maximization over the choice of a number k' (as opposed to an infinite sequence). But it has to be solved for any k which means we will need to solve for a function, it is a functional equation where the unknown is the function $V(.)$. Associated with the solution $V(.)$ to the Bellman equation is the policy function for capital

$$g(k) = \arg \max_{k' s.t. 0 \leq k' \leq f(k)+(1-\delta)k} \left[u\left(f(k) + (1-\delta) k - k' \right) + \beta V(k') \right]$$

Note that we started with a sequence problem where we were looking for an infinite sequence of numbers and we managed to reduce the maximization problem over infinite sequences to many univariate maximization problems. But these are still infinitely many problems (one for each k) or, equivalently, we are looking for a function which is an infinite dimensional object.

To recap, we have switched from thinking about making decisions regarding the optimal plan over the whole (infinite) lifetime to thinking about making decisions one period at a time, *recursively*. In each period, the planner knows how much capital stock they inherit from the past and hence how much can be produced and how many resources are available for this period. The planner focuses on deciding the current savings rate,

---

$$\max_{\{x_t\}_{t=1}^{\infty}} \sum_{t=1}^{\infty} \beta^{t-1} u(f(x_{t-1}) + (1-\delta) x_{t-1} - x_t)$$

and also changing notation for the indices $s = t - 1$

$$\max_{\{x_{s+1}\}_{s=0}^{\infty}} \sum_{s=0}^{\infty} \beta^{s} u(f(x_s) + (1-\delta) x_s - x_{s+1})$$

where x_0 $(= k_1)$ is given.

i.e. what fraction of the resources to consume now and what fraction to leave for the future. With an infinite horizon the problem is stationary: all periods have the *same structure* in the sense that if the inherited capital stock is the same then the problem facing the planner is the same and hence their decisions are the same. We therefore just need to solve for a stationary policy and value function which is independent of time.

5 The Sequence Problem versus the Functional Equation

We have loosely argued that solving the sequence problem (SP) and solving the corresponding functional equation (FE), i.e. the Bellman equation, are in some sense equivalent. But we have not provided a very rigorous statement of what exactly we mean by equivalence nor have we provided a rigorous proof of such equivalence. In particular, we have not discussed at all under what conditions this equivalence holds and whether the equivalence can be extended to other dynamic models. In addition, we have not seen why establishing this connection might be of interest. The reason is that the functional equation can be a more fruitful way to obtain general results about existence and uniqueness of a solution as well as general results about the properties of the solution. A second reason is that we can establish results about the functional equation that will allow us to solve it using computational methods that work well. All these mathematical results are provided in Stokey and Lucas's textbook "Recursive Methods in Economic Dynamics", specifically in Chapter 4 (with Chapter 3 covering mathematical preliminaries needed). Here, I provide a brief description of what these two chapters actually achieve.

Stokey and Lucas Chapter 4.1 starts with providing conditions under which the principle of optimality holds. Specifically, under relatively mild conditions they show that the maximized value in the SP will satisfy also the FE (Theorem 4.2). With an additional restriction, they also prove a converse proposition, i.e. that the value function that solves the FE is the same as the maximized value of the SP (Theorem 4.3). Theorems 4.4 and 4.5 also prove similar statements regarding the relation between the optimal sequence in the SP and the optimal policy function in the FE. These four theorems together make precise the sense in which it is 'equivalent' to solve the FE or the SP.

Having established equivalence, Stokey and Lucas proceed to establish results about the solution to the FE. Theorem 4.6 shows that the value function exists and is unique. Importantly for computations, it also shows that iterating on the Bellman equation starting at an initial

guess for V will yield new value functions that get progressively closer, and eventually converge to, the solution. This is important for the numerical computation of the solution, because it provides an algorithm for solving and ensures that the algorithm will definitely work.[8] For these results they require stricter conditions on the objective and the constraint set, one of which requires the period utility to be bounded. However, they provide alternative theorems (Theorems 4.13 and 4.14) where similar results can be shown for the case of unbounded returns.

Finally, they show that the properties of the value function are, in a sense, inherited from the properties of the objective and the constraint set satisfy. Loosely speaking, monotonicity of the objective and the constraint set will imply monotonicity of the value function (Theorem 4.7), concavity of objective and convexity of constraint set will imply concavity of the value function and continuity of the policy function (Theorem 4.8) and continuous differentiability of the objective will imply continuous differentiability of the value function (Theorem 4.11).

Let me be clear: this is intended as a brief, loose summary and not as a substitute for reading Chapters 3-4 of Stokey and Lucas which I strongly encourage you to do. We will use these results in our discussions and also see them in action in the practical applications discussed below and in the assignments.

6 Formulating the Bellman Equation in any given problem

One of the most important skills that you will need to develop as a (macro)economist is to be able to formulate the Bellman equation for each problem. That is, to be able to think recursively and come up with a parsimonious recursive representation of a sequence problem. In this section you will find some practical tricks about how to identify state and control variables and how to write the Bellman equation correctly. At the end of the section there are some examples; it will really help you to try out these examples before looking at the answers.

When asked to formulate a dynamic program essentially this is a request for three items: what are the state variables, what are the control variables and what is the Bellman equation. By far the trickiest aspect is to identify the states; once you know the state variables, the rest follows almost mechanically.

In any period, control (or choice) variables are the variables that

[8] The idea is similar to what we did when we worked backwards starting in the last period with a given V. Iterating on equation (1) with T now interpreted as counting iterations, one can start with $V_0 = 0$ and compute $V_1, V_2, ...$ obtaining closer approximations of the stationary value function V.

are currently being chosen. In the Cass-Koopmans model above, at any period t, the planner chooses c_t, i_t and k_{t+1}. Keep in mind that we want to write Bellman equations in the timeless notation so we can define c: current consumption, i: current investment and k': the capital stock chosen today and available for production in the next period. We also define k to be the capital stock available for production in the current period.

State variables are variables which are known at t and are required information in order to make decisions at t. In the Cass-Koopmans model this is k. Some straightforward ways to judge if a variable is a state variable or not are:

1. From the perspective of the sequence problem, all the variables in that model are $\{c_t, k_{t+1}, i_t\}_{t=0}^{\infty}$. Some can easily be excluded from being possible state variables. Future variables cannot be states and current choice variables cannot be states. So, for any $t = s$, $\{c_t, k_{t+1}, i_t\}_{t=s}^{\infty}$ cannot be states. States have to be given (fixed and known) at $t = s$. All variables known already at $t = s$ are $\{c_t, k_{t+1}, i_t\}_{t=0}^{s-1}$.

2. The vector $\{c_t, k_{t+1}, i_t\}_{t=0}^{s-1}$ is clearly enough information to make decisions at $t = s$, it is the whole history of the economy. But it is too much information, much less is needed. Importantly, this vector grows as we look at points in time s further into the future. If we were to keep track of this information, we would need to keep track of progressively more information as time passes. For a recursive formulation, it is essential that the state vector is of fixed dimension across periods.

3. Consumption and investment of the previous periods is really irrelevant to any decision at $t = s$. In fact, the only relevant information at $t = s$ is capital of the previous period k_s. Note that, how exactly we ended up with k_s is, in this problem, irrelevant. In particular, whether we started at $t = 0$ with little capital and saved up to now to get k_s or we started with a lot of capital and consumed so much that it was reduced to k_s does not make any difference now. At $t = s$, all that matters about the past is how many resources the planner wakes up with $(1 - \delta)k_s + f(k_s)$ and this only depends on last period's choice of capital k_s.

4. Looking at a given sequence problem, good ways to notice state variables is to look at:

- which variables appear in the objective and/or the constraints at two different points in time (in our example, we see k_{t+1} and k_t in the constraints, all other variables only appear at t).
- which variables are given as initial conditions. Initial conditions are given because it is information needed at $t = 0$ to get started and you can expect that the same variables (but s periods later) will be needed to decide at $t = s$.

Keep in mind that a problem could have many alternative recursive formulations, all of which are correct (see examples below). One criterion to choose between formulations is parsimony: the less state variables one needs, the better. We say that we focus on minimum dimension state vectors. The reasons will become clear when we talk about computation of the solution. Still, there are situations where two alternative formulations have the same dimensionality of the state vector. In such cases, the choice between the two is just a matter of taste.

Given state and control variables, writing the Bellman equation is easy as long as one follows the rules. Every Bellman equation has the following structure

$$\begin{aligned} V() = & \max_{\{\}} \{u() + \beta V()\} \\ & s.t. \\ & constraints \\ & some\ variables\ given \end{aligned}$$

Knowing which variables are states and which are controls one only has to fill in the gaps. The states have to be arguments of V on the left. In a stationary environment, the function V on the right is *the same* function but starting from tomorrow. One implication is that it must have the same number of arguments as the function on the left. A second one is that the order in which arguments appear must also be consistent with the order used on the left. The only difference is the point at which the function is evaluated. If the state today is capital inherited k, then the state tomorrow is capital inherited tomorrow k', which is just the state today but one period ahead. The control (choice) variables should be below the 'max' operator since they are choices in the maximization problem. In addition, all state variables have to be given. Finally, after all these aspects have been specified it is good practice to stop and look at the Bellman equation to ensure that all variables that appear in it have been accounted for. This means, either a variable is currently

chosen by the planner or it is currently given to the planner.[9] If you have written down a Bellman equation and there are variables in the objective or constraints that are not accounted for, then you are missing something!

Some examples will help in explaining how we write problems recursively. Examples 1- 4 are slight variations of the Cass-Koopmans model where the preferences are as usual, production is as usual, the standard resource constraint holds ($c_t + i_t = f(k_t)$) and the only variation comes in the form of different capital accumulation equations. Example 5 changes the utility and examples 6-7 change the underlying model in more fundamental ways while still remaining close to the main idea of consumption versus savings.

6.1 Example 1: No depreciation ($\delta = 0$)

Although we can simply set $\delta = 0$ to our original problem, it is helpful for what follows to go back to first principles and rethink the capital accumulation equation from scratch. Current investment i_t adds to the capital stock from next period onwards k_{t+1} and remains there without depreciating. In any period t, the capital stock can be written as

$$k_{t+1} = \sum_{s=0}^{t} i_s + k_0$$

In order to think recursively, we would need to first write this in a period-by-period form. This can be done by noticing that

$$k_t = \sum_{s=0}^{t-1} i_s + k_0$$

and therefore

$$\begin{aligned} k_{t+1} - k_t &= i_t \Longrightarrow \\ k_{t+1} &= k_t + i_t \end{aligned}$$

which is the standard capital accumulation equation with $\delta = 0$. There is really no difference between this and the standard Cass-Koopmans model, all we've done is set a parameter value $\delta = 0$. The Bellman

[9] Sometimes, especially when we look at cases with exogenous state variables, the planner will need to know a future exogenously given variable. The maximization problem will still need to specify as a constraint the rule that the planner can use to find out what the value of this future variable will be on the basis of current information (i.e. current given variables).

equation is

$$\begin{aligned} V(k) = & \max_{\{c,i,k'\}} \{u(c) + \beta V(k')\} \\ & s.t. \\ c + i = & f(k) \\ k' = & k + i \\ c \geq & 0, k' \geq 0 \\ & k \ given \end{aligned}$$

Note that this could be written in different ways. For example, one could substitute out investment (but make sure it is substituted everywhere if you do so). The same goes for c.

6.2 Example 2: Full depreciation ($\delta = 1$)

Current investment i_t adds to the capital stock for next period, k_{t+1}, but then it completely disappears. That is, in no other period is the capital stock affected by i_t. Capital accumulation is thus simply

$$k_{t+1} = i_t$$

Again, there is really no difference between this and the standard Cass-Koopmans model, all we've done is set a parameter value $\delta = 1$. The Bellman equation is

$$\begin{aligned} V(k) = & \max_{\{c,i,k'\}} \{u(c) + \beta V(k')\} \\ & s.t. \\ c + i = & f(k) \\ k' = & i \\ c \geq & 0, k' \geq 0 \\ & k \ given \end{aligned}$$

Note that we could also write this in an alternative recursive form, where the investment undertaken in the previous period, denoted by i^-, is used as the state variable (since this is the same as capital k). But we'll need to be careful to substitute out capital everywhere

$$\begin{aligned} V(i^-) = & \max_{\{c,i\}} \{u(c) + \beta V(i)\} \\ & s.t. \\ c + i = & f(i^-) \\ c \geq & 0, i \geq 0 \\ & i^- \ given \end{aligned}$$

6.3 Example 3: Full depreciation after two periods

Suppose now that capital depreciates fully in two periods but not at all before that. This means that current investment i_t (i.e. the newly built capital) will add to the capital stock available for production at $t+1$ and at $t+2$ but not to the available capital from then on. This implies

$$k_{t+2} = i_{t+1} + i_t$$

or equivalently

$$k_{t+1} = i_t + i_{t-1}$$

In this framework, it is no longer true that inherited capital k_t is enough information for all decisions. Intuitively, you can come up with situations where the planner has the same inherited capital but is actually facing a different situation. Consider two levels of investment $i^H > i^L$. In one scenario $i_{t-1} = i^H$ and $i_{t-2} = i^L$ so that $k_t = i_{t-1} + i_{t-2} = i^H + i^L$. In the second scenario, $i_{t-1} = i^L$ and $i_{t-2} = i^H$ so that $k_t = i_{t-1} + i_{t-2} = i^L + i^H$ is the same as in the first scenario. But the situation is very different. In both cases, next period's capital will equal what is invested today i_t plus what was invested yesterday i_{t-1}. But in the first scenario i_{t-1} is high and in the second $i_{t-1}$is low. This difference will matter for the current choice of investment i_t. If you try to write the Bellman equation with only k as the state variable, you will obtain a formulation that is incorrect:

$$\begin{aligned} V(k) = & \max_{\{c,i,k'\}} \{u(c) + \beta V(k')\} \\ & s.t. \\ c + i = & f(k) \\ k' = & i + i^- \\ c \geq & 0, k' \geq 0 \\ & k \ given \end{aligned}$$

This formulation is problematic because we have not accounted for i^-. Presumably i^- should be listed as given together with k. But if we add it to the 'given' variables we will also have to realize that the value function also depends on it, i.e. it is a state. Once we add it to the arguments of V on the left hand side, we also need to be careful to add it (one period ahead) to the arguments of the same function on the right

hand side. Overall, a correct recursive representation is

$$\begin{aligned} V(k, i^-) = & \max_{c,i,k'} \{u(c) + \beta V(k', i)\} \\ & s.t. \\ c = & f(k) - i \\ k' = & i + i^- \\ c \geq & 0, k' \geq 0 \\ & k, i^- \text{ given} \end{aligned}$$

An alternative would be to keep track of investment last period i^- and investment two periods before $i^=$. If we do this, we can substitute out capital altogether and write this as

$$\begin{aligned} V(i^-, i^=) = & \max_{c,i} \{u(c) + \beta V(i, i^-)\} \\ & s.t. \\ c = & f(i^- + i^=) - i \\ c \geq & 0, \ i + i^- \geq 0 \\ & i^-, i^= \text{ given} \end{aligned}$$

This is an equally valid recursive representation. Check that all variables appearing in this statement of the problem are accounted for. Also, note the order in which we include the two arguments of the value function. On the left hand side, the first argument is last period's investment and the second argument is investment two period's ago. The same structure has to be maintained on the right hand side. From the point of view of tomorrow, i is last period's investment and i^- is investment two period's ago.

6.4 Example 4: Capital becomes operative in two periods, no depreciation

We now switch back to no depreciation of capital, but change how soon capital can be used in production. In the standard Cass-Koopmans model, the assumption is that capital saved today (k_{t+1}) can be used in production tomorrow to produce $y_{t+1} = f(k_{t+1})$. Here we want change that and add some more delay. The assumption will be that newly built capital (new savings) at time t, which is i_t, can only add to the productive capital at $t + 2$. It is now important to define variables carefully. In particular, we are going to denote by k_t the capital that is available for production at t. As a result, i_t does not add to k_{t+1} anymore but rather it adds to k_{t+2}. The capital accumulation equation looks like

$$k_{t+2} = k_{t+1} + i_t$$

or equivalently

$$k_{t+1} = k_t + i_{t-1}$$

In this scenario, the planner needs to know k_t (in order to know how much production there is available today) but also needs to know last period's investment i_{t-1} in order to infer how much capital will be available tomorrow. That is, capital available today is not enough to summarize the state of the economy and an additional state is needed. Here is the formulation (using as always the timeless notation)

$$\begin{aligned} V(k, i^-) = & \max_{\{c,k',i\}} \{u(c) + \beta V(k', i)\} \\ & s.t. \\ c + i = & f(k) \\ k' = & k + i^- \\ c \geq & 0, k' \geq 0 \\ & k, i^- \text{ given} \end{aligned}$$

An alternative would be to try to substitute out investment using $i = f(k) - c$. But we would need to be careful to substitute all investments, i.e. also substitute $i^- = f(k^-) - c^-$. If we did this, we would then need additional states c^- and k^-. The following formulation is correct

$$\begin{aligned} V(k, k^-, c^-) = & \max_{\{c,k'\}} \{u(c) + \beta V(k', k, c)\} \\ & s.t. \\ k' = & k + f(k^-) - c^- \\ c \geq & 0, k' \geq 0 \\ & k, k^-, c^- \text{ given} \end{aligned}$$

Although this is correct, it uses more state variables (three instead of two). As a result it will less efficient to use in computations and the previous formulation with the minimum dimension (two) state vector will be preferred.

6.5 Example 5: Habit Persistence

Suppose we change the utility specification in the standard Cass-Koopmans model. Specifically, suppose the period utility depends on both current consumption c_t and past consumption c_{t-1}, the underlying idea being that it's not current consumption that matters for utility but also how high is current consumption relative to last years's consumption. So the period utility can be written as $u(c_t, c_{t-1})$. This could be due to the formation of habits.

Determine the state variables, carefully define all variables you will use in the timeless notation and write the Bellman equation in this case.

6.6 Example 6: Cake Eating

Consider a cake-eating problem. We have a cake of size W which can be eaten over $T+1$ periods (and does not depreciate). Denote how much of the cake is eaten each period by c_t and assume utility is given as usual by

$$\sum_{t=0}^{T} \beta^t u(c_t)$$

The constraint can be expressed as

$$\sum_{t=0}^{T} c_t \leq W$$

Can you write this problem recursively? Will it be a stationary problem? What about the case $T = \infty$?

6.7 Example 7: Banana Problem with variable endowment

See Assignments, this is an example with exogenous state variable(s).

7 Functional Euler Equation

In the sequence problem we showed that the optimal capital sequence will satisfy an Euler equation describing the trade-off between current consumption and future consumption in terms of marginal benefits and marginal costs

$$u'(f(k_t)-k_{t+1}+(1-\delta)k_t) = \beta u'(f(k_{t+1})-k_{t+2}+(1-\delta)k_{t+1})\left[f'(k_{t+1})+(1-\delta)\right]$$

A similar characterization can be obtained using the dynamic programming approach, but now applied to the optimal policy function $g(k)$ rather than the optimal capital sequence.[10] To obtain this functional Euler equation for the standard Cass-Koopmans model recall that the Bellman equation is

$$V(k) = \max_{k' s.t. 0 \leq k' \leq f(k)+(1-\delta)k} \left[u\left(f(k)+(1-\delta)\,k-k'\right)+\beta V(k')\right]$$

[10] This section assumes differentiability of the value function $V(k)$. For conditions under which this is the case see the Benveniste-Scheinkman theorem (Theorem 4.10 in Stokey and Lucas).

Assuming the inequalities do not bind and that the objective is concave, we could find the maximum by taking first order conditions with respect to k'

$$-u'(f(k)+(1-\delta)k-k')+\beta V'(k')=0$$

Now note that this condition has to hold *at the optimal* k' and recall that we have defined the policy function to be exactly that

$$g(k)=\arg\max_{k's.t.0\leq k'\leq f(k)+(1-\delta)k}\left[u\left(f(k)+(1-\delta)k-k'\right)+\beta V(k')\right]$$

Therefore the condition can be written as

$$u'(f(k)+(1-\delta)k-g(k))=\beta V'(g(k)) \tag{2}$$

which again describes the intertemporal trade-off at the margin between current consumption and savings, but now states this as a condition on the policy function $g(k)$. The difficulty with this characterization is that it also involves the derivative of the value function (another unknown function). It turns out that this can be simplified and written in terms of the g function only. The idea is to write the value function $V(k)$ as a function of the policy function $g(k)$, which can be done by going back to the Bellman equation and substituting the optimal choice $g(k)$ for k' (and, hence, dropping the 'max' operator)

$$V(k)=u\left(f(k)+(1-\delta)k-g(k)\right)+\beta V(g(k))$$

Using this representation of $V(k)$, we can now compute the derivative of V with respect to k

$$\begin{aligned}V'(k)&=\left[f'(k)+1-\delta-g'(k)\right]u'(f(k)+(1-\delta)k-g(k))+\beta g'(k)V'(g(k))\\&=\left[f'(k)+1-\delta\right]u'(f(k)+(1-\delta)k-g(k))+g'(k)\left[-u'(f(k)+(1-\delta)k-g(k))+\beta V'(g(k)\right.\end{aligned}$$

An increase in the available capital k, increases current resources by $[f'(k)+1-\delta]$ and these are valued at marginal utility $u'(f(k)+(1-\delta)k-g(k))$. But it might also affect the optimal choice $k'=g(k)$, which is what the second term captures. An increase in k, will change the optimal choice by $g'(k)$ and this will have two effects on overall utility. First, it will decrease current consumption one-to-one and this decrease is valued at current marginal utility. Second, it will potentially increase the continuation value, i.e. increase overall utility by $\beta V'(g(k))$. These are the two terms in the second bracket. Notice. however, that these marginal effects of a change in k' must cancel out at the optimum (i.e. at $k'=g(k)$). This is exactly what the first order condition (2) required to be true for the optimal policy $g(k)$.

The fact that the 'indirect' effect of a change in k (through the effect $g'(k)$ that it has on the optimal choice) is zero, is an application of the well-known envelope theorem. The conclusion is that the effect of a change in k on the value, i.e. the V' function we were looking for, is simply the direct effect of k on current utility through the change in available resources

$$V'(k) = [f'(k) + 1 - \delta]\, u'(f(k) + (1-\delta)\, k - g(k))$$

As a result we can write the first order condition (2) as

$$u'(f(k)+(1-\delta)\, k-g(k)) = \beta\, [f'(g(k)) + 1 - \delta]\, u'(f(g(k))+(1-\delta)\, g(k)-g(g(k)))$$

which is an equation in an unknown function $g\,(k)$. Comparing with the Euler equation in the sequence problem, the idea is exactly the same. We have simply replaced k_t with current capital k, k_{t+1} with the current choice of capital $g\,(k)$ and k_{t+2} with the choice of capital tomorrow $g(g(k))$.

8 Solving the Dynamic Programming Problem

There are two ways to solve a dynamic programming problem

1. Analytically using the guess-and-verify approach
2. Numerically by iterating on a computer

The analytical approach will only work in very special cases, one example follows. But in general, the approach will have to be computational. The main idea of the computational approach is discussed at the end of this section.

8.1 Guess-and-verify Approach

The idea here is to come up with a guess for the functional form of the value function and then plug it in the Bellman Equation to: 1. Verify that the functional form is correct and 2. Determine any parameters used in the general functional form. The 'verify' part is easy, it only involves some calculations, in particular solving a simple maximization problem and equating coefficients. Coming up with a guess is obviously harder, in fact in most cases it is impossible since there is no analytical, closed form solution for the value function. Even for the cases where such a solution exists, guessing a functional form can be quite difficult. Fortunately, we can use the finite horizon version of a problem to come up with a good guess in a structured way. The strategy is as follows: Solve

first a one period model, then a two-period model , then a three-period model etc. Find the value functions for these cases and check if the functional form coincides even though the exact parameters differ. At this point, one can proceed to determining the parameters. This involves assuming the value function is given by this general form with some (yet to be determined) coefficients, plugging this parametric function into the right hand side of the Bellman equation and solving the maximization problem to obtain the resulting value function on the left hand side. Finally, comparing the two value functions (the one plugged in on the right with the one resulting on the left) and equating coefficients, one can solve for the parameters needed to ensure they are the same. This yield the stationary value function. An example follows.

8.1.1 An Example: The Brock-Mirmann solution

Consider the standard Cass-Koopmans model with the following additional assumptions:

$$\delta = 1,\ f(k) = k^{\alpha},\ u(c) = \ln c$$

Now start solving backwards starting with the value of a 'zero-period problem' $V_0(k) = 0$. First, solve the one period problem

$$\begin{aligned} V_1(k) = & \max_{\{c,k'\}} \{\ln c + \beta 0\} \\ & s.t. \\ c + k' = & k^{\alpha} \\ c \geq & 0, k' \geq 0 \\ & k \text{ given} \end{aligned}$$

Using the Inada condition on the utility one can conclude that $c > 0$ and using complementary slackness one can show that $k' = 0 \,(\equiv g_1(k))$. So $c = k^{\alpha}$ which implies $V_1(k) = \alpha \ln k$.

Now use $V_1(.)$ to solve a two-period problem and find $V_2(.)$):

$$\begin{aligned} V_2(k) = & \max_{\{c,k'\}} \{\ln c + \beta V_1(k')\} = \max_{\{c,k'\}} \{\ln c + \beta\alpha \ln k'\} \\ & s.t. \\ c + k' = & k^{\alpha} \\ c \geq & 0, k' \geq 0 \\ & k \text{ given} \end{aligned}$$

Note that if it happens that you find V_2 and it is exactly the same as V_1, then you have found a *stationary* solution! This is because if you then use V_2 and look for V_3 you'll find the same again, and so on. Here, it

turns out that V_2 is not the same as V_1. First, we'll need to show that none of the two inequality constraints will bind (show it). Given this, the first order conditions will give

$$\frac{1}{c} = \lambda$$
$$\frac{\alpha\beta}{k'} = \lambda$$

Using the resource constraint

$$\frac{\alpha\beta}{k'} = \frac{1}{k^\alpha - k'} \Rightarrow$$
$$k' = \frac{\alpha\beta}{1+\alpha\beta} k^\alpha \equiv g_2(k)$$

Now substitute in the objective to find V_2

$$\begin{aligned} V_2(k) &= \ln\left(\left(1 - \frac{\alpha\beta}{1+\alpha\beta}\right)k^\alpha\right) + \alpha\beta\left(\ln\frac{\alpha\beta}{1+\alpha\beta}k^\alpha\right) \\ &= \ln\left(\frac{1}{1+\alpha\beta}\right) + \alpha\beta\left(\ln\frac{\alpha\beta}{1+\alpha\beta}\right) + (1+\alpha\beta)\,\alpha\ln k \end{aligned}$$

Next step is to find V_3 (try it). But notice that a functional form is already emerging: when we plugged in a logarithmic function for the value function $V_1(k)$, the resulting value function $V_2(k)$ ended up also being a logarithmic function. The parameter coefficient on $\ln k$ is different and the constant is also different, but the functional form is the same and of the form $A + B\ln k$ for some parameters A, B. We can use this observation to guess that all V_j have the same form and try to determine what values should A and B take to ensure that using this as a continuation value would yield the exact same numbers for the current value. So let us guess that the stationary value function V has the form$V(k) = A + B\ln k$.

At this point the objective is to verify this functional form guess is correct and at the same time solve for the specific A and B. The infinite horizon Bellman equation we are trying to solve is

$$V(k) = \max_{0\le k'\le k^\alpha} \{\ln(k^\alpha - k') + \beta V(k')\}$$
$$k \text{ given}$$

First compute the right hand side by solving the maximization problem one more time (at this point it should be clear that you can show mathematically that the inequality constraints will not bind, so I just assume

this to avoid repetition)

$$\max_{k'}\{\ln(k^\alpha - k') + \beta(A + B\ln k')\}$$

First order conditions

$$\begin{aligned}\frac{1}{(k^\alpha - k')} &= \beta B\frac{1}{k'} \Rightarrow \\ \beta B(k^\alpha - k') &= k' \Rightarrow \\ k' &= \frac{\beta B k^\alpha}{1+\beta B} \equiv g(k)\end{aligned}$$

and

$$c = \frac{1}{1+\beta B}k^\alpha \equiv g^c(k)$$

These are the policy functions for capital and consumption which can be inserted into the objective to find the right hand side of the Bellman equation

$$\begin{aligned}&\{\ln(\frac{1}{1+\beta B}k^\alpha) + \beta(A + B\ln\left[\frac{\beta B k^\alpha}{1+\beta B}\right])\} \\ &= -\ln(1+\beta B) + \alpha\ln k + \beta A + \beta B\ln\left(\frac{\beta B}{1+\beta B}\right) + \beta B\alpha\ln k \\ &= -\ln(1+\beta B) + \beta A + \beta B\ln\left(\frac{\beta B}{1+\beta B}\right) + (1+\beta B)\alpha\ln k\end{aligned}$$

For the Bellman Equation to be satisfied it must be that this is equal to $V(k) = A + B\ln k$. All we have to do is find A and B so that the two are the same. We do this by equating coefficients

$$\begin{aligned}A &= -\ln(1+\beta B) + \beta A + \beta B\ln\left(\frac{\beta B}{1+\beta B}\right) \\ B &= (1+\beta B)\alpha\end{aligned}$$

and solving the two equations for A, B. The second one implies

$$B = \frac{\alpha}{1-\alpha\beta}$$

and the first one

$$\begin{aligned}
A &= -\ln(1+\beta B) + \beta A + \beta B \ln\left(\frac{\beta B}{1+\beta B}\right) \Rightarrow \\
(1-\beta)A &= -\ln\left(1+\beta\frac{\alpha}{1-\alpha\beta}\right) + \beta\frac{\alpha}{1-\alpha\beta}\ln\left(\frac{\beta\frac{\alpha}{1-\alpha\beta}}{1+\beta\frac{\alpha}{1-\alpha\beta}}\right) \Rightarrow \\
&= -\ln\left(\frac{1}{1-\alpha\beta}\right) + \frac{\alpha\beta}{1-\alpha\beta}\ln(\alpha\beta) \Rightarrow \\
A &= \frac{1}{1-\beta}\left[\frac{\alpha\beta}{1-\alpha\beta}\ln(\alpha\beta) + \ln(1-\alpha\beta)\right]
\end{aligned}$$

We therefore have a solution to the dynamic optimization problem. The value function is

$$V(k) = \frac{1}{1-\beta}\left[+\frac{\alpha\beta}{1-\alpha\beta}\ln(\alpha\beta) + \ln(1-\alpha\beta)\right] + \frac{\alpha}{1-\alpha\beta}\ln k$$

We also have expressions for the policy functions

$$\begin{aligned}
g(k) &= \frac{\beta\frac{\alpha}{1-\alpha\beta}k^{\alpha}}{1+\beta\frac{\alpha}{1-\alpha\beta}} = \alpha\beta k^{\alpha} \\
g^{c}(k) &= (1-\alpha\beta)k^{\alpha}
\end{aligned}$$

You might notice that the policy function for capital coincides with what we found in the assignments for this exact same model when using the sequence problem and the change of variable trick to solve the Euler equation

$$k_{t+1} = \alpha\beta k_t^{\alpha}$$

This special case of the Cass-Koopmans model delivers a *constant* savings rate, $s_t = \frac{i_t}{y_t} = \frac{k_{t+1}}{k_t^{\alpha}} = \alpha\beta$. Recall that in the Solow model this was assumed to start with, whereas here we have derived this as an endogenous outcome. This might suggest that the Solow assumption was reasonable after all, but note that this has been shown to be true only for a very special case ($\delta = 1$, $f(k) = k^{\alpha}$, $u(c) = \ln c$). A slight modification (e.g. if $\delta < 1$) would undo this result and the fraction of output consumed and saved will not be fixed, but will depend on time t (from a sequential point of view) or, better, on the current state (from a recursive point of view). Also note that, even in the special case, the optimal rate of savings is slightly different than the Golden rule in Solow's model due to the presence of discounting $\beta < 1$.

8.1.2 What goes wrong with a wrong guess?

Suppose we had (wrongly) guessed the functional form of V to be linear instead of logarithmic, i.e. $V(k) = A+Bk$. Try solving the maximization problem with this guess

$$\begin{aligned} &\max_{k'}\{\ln(k^\alpha - k') + \beta(A + Bk')\} \\ k' \geq 0 \\ &k \text{ given} \end{aligned}$$

and showing that the functional form guessed is incorrect.

8.1.3 Numerical Solution by Iterating on the Bellman Equation

The typical Bellman equation will not admit an analytical solution. The previous approach will fail because as we are iterating on the Bellman equation to find a pattern, no pattern will emerge. In fact, in most cases it will not even be possible to solve the two-period maximization problem analytically in closed form. As a result, the approach will have to be numerical. Conceptually, the idea is essentially the same as before, only now it is implemented on a computer. We will start from a one-period model, then move to a two-period etc. and keep on iterating until the value function we plug in on the right hand side results in a value function on the left hand side that is at least approximately the same. Theorem 4.6 of Stokey and Lucas ensures that this iterative approach will work because the right hand side of the Bellman equation is a contraction mapping. A brief summary of the practical implementation follows.

We are interested in obtaining an approximation of a function $V(k)$. Given that there is no closed form for the function, one way to approximate it is to compute its value at a large but finite number of points and then "connect the dots". So the first step consists in *discretizing* the state space. The state variable here is capital and the state space (the range of values that the state variable can take) is $k \in (0, \infty)$. To make practical implementation feasible, we will need to bound the state space, which means to pick a minimum and a maximum value of capital $k^{\min}$, $k^{\max}$ and focus on an approximation of the function within the interval $\left[k^{\min}, k^{\max}\right]$. We would like to include as much as possible of the state space. This means we can choose a $k^{\min}$ that is very low and close to zero (say $k^{\min} = 0.001$) and $k^{\max}$ as high as possible. Unfortunately, there are no general rules available to decide on what is high enough. The state space focused on should at least include the steady state k^* and hopefully at least some range of values above the steady state too, so one option is to choose $k^{\max}$ to be a multiple of the steady state value,

say $k^{\max} = 3k^*$.[11]

At this point we have restricted attention to the closed and bounded interval $\left[k^{\min}, k^{\max}\right]$. We want to move from a continuous state space to a discrete one (so we can have a finite number of points). So we choose a large number of points N. Clearly, the more points used, the more accurate the solution will be, but the trade-off is that it will take longer to compute it.[12] For the sake of exposition, let us pick an (unreasonably) small number of grid points $N = 4$, suppose $k^* = 3$ and choose $k^{\max} = 3k^* = 9$ and $k^{\min} = 0$. With these numbers, the continuous interval is $[0, 9]$ and if we choose equidistant grid points for the discretization the discretized version consists of 4 points $\{0, 3, 6, 9\}$.[13]

Given the discretization, the objective is now to compute the value function at these four points, i.e. to find $V(0)$, $V(3)$, $V(6)$ and $V(9)$. We could also find the corresponding policy function values. We do this by iterating on the Bellman equation until we find a fixed point. The general idea is to start with an initial guess for our (discretized) value function V^0. For example, we could start with $V^0(0) = V^0(3) = V^0(6) = V^0(9) = 0$. Then use this V^0 to find the next guess V^1 using the Bellman equation

$$\begin{aligned} V^j(k) = &\max_{k'} \left[u(f(k) - k' + (1-\delta)\,k) + \beta V^{j-1}(k')\right] \\ &s.t. \\ k' &\leq f(k) + (1-\delta)\,k \\ k' &\in \{0, 3, 6, 9\} \\ &k \text{ given} \end{aligned}$$

[11] For this specific model, we could also argue that there is value of capital k that is so high that it is unsustainable, meaning that it would necessarily lead to a smaller choice for k'. To find that bound $k^{\max}$, note that consumption non-negativity requires

$$k' \leq f(k) + (1-\delta)\,k$$

so choosing $k^{\max}$ such that

$$k^{\max} = f(k^{\max}) + (1-\delta)\,k^{\max}$$

we know that if $k = k^{\max}$ then necessarily $k' < k^{\max}$. This is an alternative choice for $k^{\max}$.

[12] Again, how large is 'large'? There is no obvious answer to this so the idea is to choose some number, say $N = 100$, compute everything and then increase N to 200, redo everything and check whether there is significant change in the obtained solution. This can be repeated until increasing N further does not affect results in a signficant way.

[13] Remember $k^{\min}$ should be close to, but not equal to, zero in practice. We continue with $\{0, 3, 6, 9\}$ as the discrete state space for the purpose of exposition only. The Figure shows instead the case $k^{\min} = 0.001$.

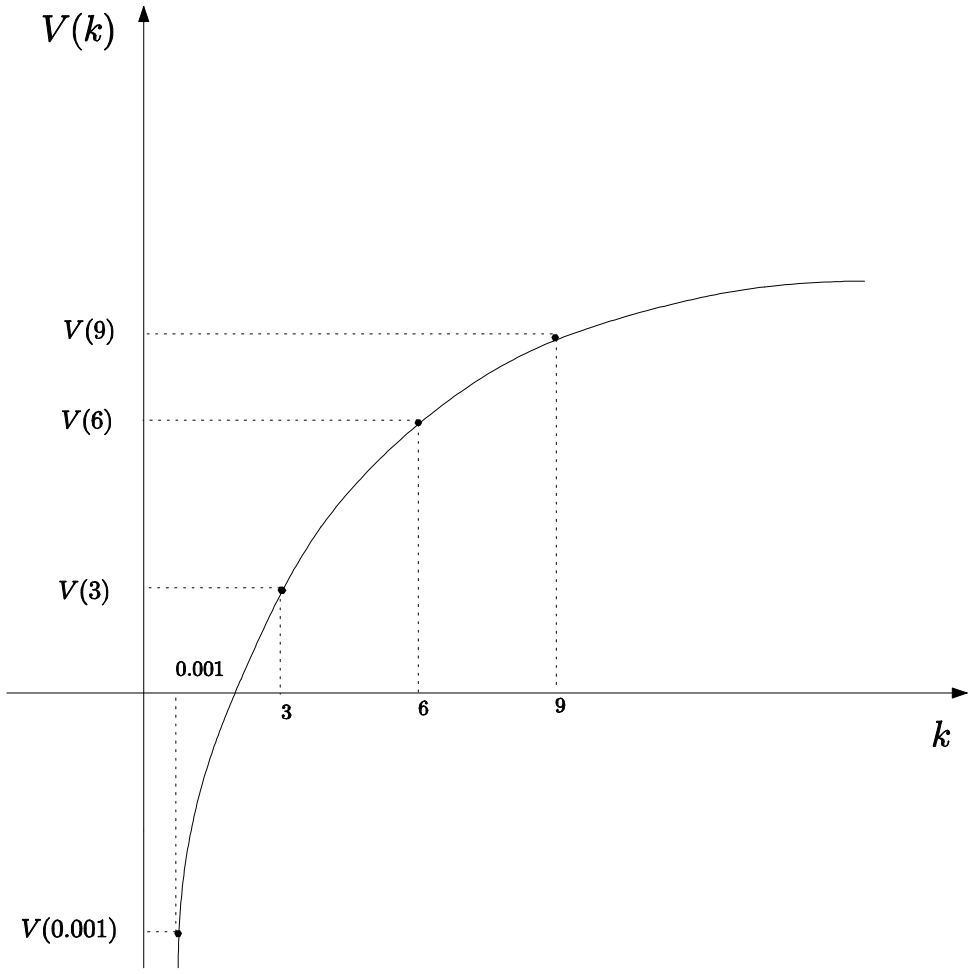

Figure 2: The value function and the four points at which it will be evaluated.

Note that finding $V^j(k)$ means finding four numbers that correspond to the value of the function V evaluated at the 4 capital points. For each of these numbers, one needs to solve a maximization problem on the right hand side. We have also restricted the possible choices k' in the maximization problem to lie in the discretized state space too. That is, there are only 4 possible values for k'. Taking first order conditions will not work in this setup (what does it mean to increase k' marginally if k' can only jump discretely?). But the maximization is actually easier: all that needs to be done to solve the maximization problem is to plug in the four possible choices, evaluate the objective and choose the choice that gives the highest objective (being careful not to allow for choices that violate the consumption non-negativity constraint). This maximization has to be done four times, for $k = 0, 3, 6, 9$ respectively. Having done this for all possible k in the grid, we have a new value function vector $\{V^1(0), V^1(3), V^1(6), V^1(9)\}$ and a new policy vector $\{g^1(0), g^1(3), g^1(6), g^1(9)\}$. The process can be restarted to find the next guess. We keep iterating on this process until the new guess and the old guess are close enough. To be precise on 'close enough', one needs to define a metric for the distance between two vectors. The safest bet is to take the maximum element of the absolute difference between the two. So the iterations should continue as long as

$$\max_k \left[\left\{V^j(k) - V^{j-1}(k)\right\}\right] > \varepsilon \tag{3}$$

where ε is a tolerance level that we will have to specify. Clearly, the closer ε is to zero, the more accurate is the final solution. To summarize, the algorithm is as follows

1. Discretize the state space: Choose the maximum and minimum values for capital as well as the number N of points in our discretization.

2. Initialize the (discretized) value function V^0 by using the zero-period problem value function, i.e. by choosing a vector of zeros of length equal to N.

3. At each iteration j, given the previous iteration's value function V^{j-1}, find the next (discretized) value function by solving N maximization problems, one for each value of the state k. The solution of each maximization boils down to evaluating the objective N times (once for each possible choice k') and choosing the one that yields the highest value. This defines the new value function V^j and can also be used to find the corresponding policy function g^j.

4. Compare the value V^j computed in step 3 with the previous value V^{j-1}. If condition (3) is true, then repeat step 3 for the next j.

5. Keep repeating steps 3-4 until the condition (3) is not true any more. At this point, we have found the value function.