We shall now study optimization problems when there are more than two periods involved. We shall essentially get introduced to dynamic optimization techniques while trying to minimize the description of mathematical detail underlying the methods.

1 Finite horizon problems

We shall first study a finite horizon optimization problem. In particular, we shall study a $T$-period extension of our two-period production economy model. Consider a closed economy inhabited by a representative agent who lives for $T \in [2, \infty)$ periods. The agent’s preferences are given by

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

where $u(.)$ is increasing and strictly concave in $c$ and $u'(0) = \infty$. The agent can produce goods using the technology

$$y_t = f(k_t)$$

where $f$ is strictly increasing and concave in $k$ while $k$ denotes capital per agent. The agent maximizes her welfare subject to

$$c_t + k_{t+1} = f(k_t) + (1 - \delta) k_t, \quad t \leq T$$

$$k_{t+1} \geq 0$$

The second constraint recognizes that capital cannot be negative in any period. Assume that $k_0$ is given.
There are multiple ways to proceed. The first is to set up the Lagrangean with a sequence of periodic budget constraints – one for each period. Thus,

\[ L = \sum_{t=0}^{T} \beta^t u(c_t) + \sum_{t=0}^{T} \lambda_t [f(k_t) + (1 - \delta) k_t - k_{t+1} - c_t] + \sum_{t=0}^{T} \mu_t k_{t+1} \]

where \( \lambda_t \) is the Lagrange multiplier for the constraint in period \( t \) and \( \mu_t \) is the multiplier on the non-negativity constraint on \( k_{t+1} \) for all \( t \).

The first order conditions that define the optimal choices of \( c_t \) and \( k_{t+1} \) for all \( t \leq T - 1 \) are

\[ \beta^t u'(c_t) = \lambda_t, \text{ for all } t \]
\[ \lambda_t = \lambda_{t+1} [f'(k_{t+1}) + 1 - \delta] + \mu_t, \text{ for } t \leq T - 1 \]
\[ \lambda_T = \mu_T \]
\[ \mu_t k_{t+1} = 0 \]

While the other conditions are standard, the last condition is the Kuhn-Tucker condition on the non-negativity constraint on \( k_{t+1} \). It says that if the constraint doesn’t bind, i.e., \( k_{t+1} > 0 \), then the corresponding Kuhn-Tucker multiplier on the constraint must be zero.

Next, note that for all \( t \leq T - 1 \) if \( k_{t+1} = 0 \) in any period then \( c_s = 0 \) for all \( s > t \). Since \( u'(0) = \infty \) this cannot be an equilibrium. Hence, \( k_t > 0 \) for all \( t < T \) and \( \mu_t = 0 \). In period \( T \) however there is no further value to \( k_{T+1} \). Hence, \( k_{T+1} = 0 \) and \( \mu_T = \lambda_T > 0 \). Combining the first two conditions and using the fact that \( \mu_t = 0 \) for all \( t < T \) gives

\[ u'(c_t) = \beta u'(c_{t+1}) [f'(k_{t+1}) + 1 - \delta], \text{ for } t \leq T - 1 \]

But this is the same Euler equation that we derived in the two period model. Moreover, since \( \mu_T = \lambda_T \), the first and fourth optimality conditions jointly give

\[ \beta^T u'(c_T) k_{T+1} = 0. \]

This condition is called the transversality condition and is a necessary condition for an optimum.
Since \( c_T = f ( k_T ) + ( 1 - \delta ) k_T \), we can now use the Euler equation for period \( T - 1 \) to solve for \( c_{T-1} \) as a function of \( k_T \). Since the period \( T - 1 \) constraint is

\[
c_{T-1} + k_T = f ( k_{T-1} ) + ( 1 - \delta ) k_{T-1},
\]

we can now solve for \( k_T \) as a function of \( k_{T-1} \). That, in turn allows us to solve for \( c_{T-1} \) as a function of \( k_{T-1} \) as well.

Now go to the period \( T - 2 \) Euler equation. It says that

\[
u' ( c_{T-2} ) = \beta u' ( c_{T-1} ) [ f' ( k_{T-1} ) + 1 - \delta ]
\]

Since the right hand side of this is a function of \( k_{T-1} \) (recall the last step in the previous paragraph), we can solve for \( c_{T-2} \) as a function of \( k_{T-1} \). Substituting this into the period \( T - 2 \) budget constraint yields a solution for \( k_{T-1} \) and, hence, \( c_{T-2} \) as function of \( k_{T-2} \). Proceeding backwards like this one would finally get a solution for \( c_0 \) and \( k_1 \) as functions of \( k_0 \). Since \( k_0 \) is known, we have solved for the entire path of \( c \) and \( k \).

This solution method can be extended to the infinite horizon case quite easily. The only modification to the optimality conditions would be that the transversality condition would now be written as

\[
\lim_{T \to \infty} \beta^T u' ( c_T ) k_{T+1} = 0.
\]

### 1.1 Dynamic Programming

Dynamic problems can alternatively be solved using dynamic programming techniques. Thus, we start by first defining the Bellman equation:

\[
V ( k_t ) = \max_{c_t, k_{t+1}} [ u ( c_t ) + \beta V ( k_{t+1} ) ].
\]

The Bellman equation defines \( V ( k ) \) as the maximized level of lifetime welfare that can be achieved by the agent when she starts with \( k \) units of capital. This is often also referred to as the recursive representation of the agent’s problem. The \( V \) function is often called the Value Function. In words, the value function is a mapping from the state today to the maximized value of the objective function.
The idea behind this equation is that at any point in time the agent is maximizing current utility and the entire present discounted value of future utility. In this problem \( k \) is the state variable, i.e., it is given in any period and can only be altered for future periods. The key idea behind the recursive representation of the problem is that one doesn’t need to keep track of all the past decisions of the agent in periods \( t - 1, t - 2, t - 3, \ldots \) etc.. All relevant decisions made in the past are captured in the value of the state variable at date \( t \), namely \( k_t \). In other words, the problem of choosing an infinite sequence of \( c \)’s can be broken down into a sequence of two period problems.

To make this method work one needs to have some regularity conditions satisfied by the objective function and the constraint set defined by the technology. Those conditions ensure that the value function \( V \) converges, that the limiting convergent function is increasing, continuous and differentiable in the state variable and a maximum exists. While we shall not dig any deeper here, those interested in the fiery underbelly of dynamic optimization methods should consult the books by Stokey and Lucas or Ljungquist and Sargent that are referenced in the reading list.

By using the constraint to substitute out for \( c \) in the value function one can rewrite the value function as

\[
V (k_t) = \max_{k_{t+1}} [u (f (k_t) + (1 - \delta) k_t - k_{t+1}) + \beta V (k_{t+1})]
\]

Differentiating the value function with respect to \( k_{t+1} \) gives the first order condition

\[
0 = -u' (c_t) + \beta V' (k_{t+1})
\]

To proceed we need to know what \( V' (k_{t+1}) \) is. We do so by first determining that

\[
V' (k_t) = u' (c_t) [f' (k_t) + 1 - \delta] + [-u' (c_t) + \beta V' (k_{t+1})] \frac{dk_{t+1}}{dk_t}
\]

But at an optimum we must have \(-u' (c_t) + \beta V' (k_{t+1}) = 0\). Hence, by the envelope theorem,

\[
V' (k_t) = u' (c_t) [f' (k_t) + 1 - \delta].
\]

This is called the Benveniste-Scheinkman condition. Note that the derivation of this expression crucially requires that the value function be not just continuous but also differentiable.
Notice that while we have determined $V'(k_t)$, the first order condition for optimality equates $u'(c_t)$ to $V'(k_{t+1})$. How do we go from one to the other? This is where we use the fact that the $V$ function is time-invariant (the functional form is time invariant, the arguments may be time varying). Using the convergence of the $V$ function, we can update the expression for $V'(k_t)$ by one period to get
\[ V'(k_{t+1}) = u'(c_{t+1}) \left[ f'(k_{t+1}) + 1 - \delta \right]. \]

Substituting this into the first order condition gives
\[ u'(c_t) = \beta u'(c_{t+1}) \left[ f'(k_{t+1}) + 1 - \delta \right], \quad t = 0, 1, ..., T - 1 \]
But this equation is exactly the much loved and admired Euler equation that we have been meeting frequently these past few weeks.

To proceed we now impose the additional condition
\[ \lim_{T \to \infty} \beta^T V(k_T) = 0 \]
which is a terminal condition on the value function. This condition basically says that the discounted value of capital goes to zero over time, i.e., it goes asymptotically to zero as time goes to infinity. This is then the infinite horizon analog of the last period in finite horizons.

With this additional condition the solutions to
\[ W = \max_{c_t} \sum_{t=0}^{\infty} \beta^t u(c_t) \]
and
\[ V(k_t) = \max_{c_t,k_{t+1}} \left[ u(c_t) + \beta V(k_{t+1}) \right] \]
coincide.

### 1.2 The method in action

To see how one can apply this consider a special case of the model that we outlined above. Let $u(c) = \ln c$, $f(k) = k^\alpha$, and $\delta = 1$, i.e., full depreciation. Under these assumptions the
Euler equation is
\[
\frac{c_{t+1}}{c_t} = \alpha \beta k_{t+1}^{\alpha-1}.
\]
Suppose the economy ends after date \( T \). Since \( k_{T+1} = 0 \), the period \( T \) budget constraint implies that \( c_T = k_T^\alpha \). Hence,
\[
V_T(k_T) = \alpha \ln k_T
\]
Moving back a period, at time \( T - 1 \) our problem then is to maximize
\[
\ln c_{T-1} + \alpha \beta \ln k_T
\]
subject to
\[
c_{T-1} + k_T = k_{T-1}^\alpha
\]
Note that at \( T - 1 \) we enter the period with \( k_{T-1} \). Given that, what is the best that we can do? This problem gives as the optimal solutions
\[
k_T = \left( \frac{\alpha \beta}{1 + \alpha \beta} \right) k_{T-1}^\alpha
\]
\[
c_{T-1} = \left( \frac{1}{1 + \alpha \beta} \right) k_{T-1}^\alpha
\]
Using these in the value function for period \( T - 1 \) yields
\[
V_{T-1}(k_{T-1}) = \text{const} + (\alpha + \alpha^2 \beta) \ln k_{T-1}
\]
Having solved for \( T - 1 \) choice variables we now can move one period further back to \( T - 2 \). We enter this period with \( k_{T-2} \) and try to maximize
\[
\ln c_{T-2} + \beta V_{T-1}(k_{T-1})
\]
subject to
\[
c_{T-2} + k_{T-1} = k_{T-2}^\alpha
\]
where \( V_{T-1}(k_{T-1}) \) was derived above. This problem yields the solutions
\[
k_{T-1} = \left( \frac{\alpha \beta (1 + \alpha \beta)}{1 + \alpha \beta (1 + \alpha \beta)} \right) k_{T-2}^\alpha
\]
The associated value function for period $T - 2$ is

$$V_{T-2}(k_{T-2}) = \text{const} + \alpha \left( 1 + \alpha \beta + \alpha^2 \beta^2 \right) \ln k_{T-2}$$

Proceeding backwards recursively in this way one would eventually end up with

$$k_{T-j+1} = \left( \frac{\alpha \beta + (\alpha \beta)^2 + \ldots + (\alpha \beta)^j}{1 + \alpha \beta + (\alpha \beta)^2 + \ldots + (\alpha \beta)^j} \right) k_{T-j}^\alpha$$

$$c_{T-j} = \left( \frac{1}{1 + \alpha \beta + (\alpha \beta)^2 + \ldots + (\alpha \beta)^j} \right) k_{T-j}^\alpha$$

It is easy to see that these successive terms for $c$ and $k$ have a progression to them. In particular, in the limit as $j \to \infty$ they converge to

$$k_t = \alpha \beta k_t^\alpha$$

$$c_t = (1 - \alpha \beta) k_t^\alpha.$$ 

You should check that the value function also converges in this example.

## 2 Determining the Value Function

How does one determine the functional form of the value function? Often, in applied policy work that is our primary object of interest. There are three basic methods of determining the value function: (a) Value function iteration; (b) Policy function iteration; and (c) Guess and verify method. We shall illustrate the first and the third methods by using a concrete example.

### 2.1 Value function iteration

Consider the general problem:

$$\max_{u_t} \sum_{t=0}^{T} \beta^t r(x_t, u_t)$$
subject to the constraint

\[ x_{t+1} = g(x_t, u_t) \]

where \( x \) is known at the beginning of every period, i.e., it is the state variable. In recursive form this problem can be written as

\[ V_1(x) = \max_u \{ r(x, u) + \beta V_0(g(x, u)) \} \]

A useful starting guess is \( V_0 = 0 \). Substituting this into the above gives

\[ V_1(x) = \max_u r(x, u) \]

This will yield an optimal solution \( u = h_0(x) \). This is typically called the policy function. Substituting this policy function back into \( V_1 \) and using this value function as the new guess gives the updated problem

\[ V_2(x) = \max_u \{ r(x, u) + \beta V_1(g(x, h_0(x))) \} \]

This problem yields the solution \( u = h_1(x) \) which can be used to derive a solution for \( V_2(x) \). Using this to update the initial guess one then gets the new problem

\[ V_3(x) = \max_u \{ r(x, u) + \beta V_2(g(x, h_1(x))) \} \]

Continuing this recursive process we will get

\[ V_{j+1}(x) = \max_u \{ r(x, u) + \beta V_j(g(x, u)) \} \]

where \( V_j \) denotes the jth guess for the value function. Under some conditions (see the Stokey and Lucas book for details)

\[ \lim_{j \to \infty} V_j(x) = V(x) \]

i.e., the value function converges to an invariant function of the state variable \( x \).

An alternative method is to start with an initial guess on the policy function \( h_0(x) \) and iterating on that until it converges to an invariant function. It works in a similar way to the value function iteration technique. Once a convergent policy function is determined the value function follows directly from the Bellman equation.
2.2 Guess and verify

Sometimes, based on past experiences or on having solved similar problems one may have a sense of what the value function may look like. In such cases it is often more direct and straightforward to start with a guess for the value function and then try and verify the guess. This verification essentially involves being able to solve for all the unknown guessed coefficients as functions of the exogenous parameters of the model. The typical way to proceed is to guess the value function as a function of the state variable (with the function being similar to the objective function) and a constant, solve for the implied policy function using that guess, substituting the solution back into the value function and then equate coefficients from the resulting value function with the initial guess to see if you can solve for all the unknown coefficients as functions of the known parameters of the model. Problem Set 3 will take you through one such exercise.

This method however doesn’t work all the time. The main problem is the lack of closed form solutions in a vast variety of problems including fairly simple ones. In such cases the more numerical approaches like Value function iteration or Policy function iteration are the way to go.