

Comments Welcome

# Lecture Notes On Solution Methods for Representative Agent Dynamic Stochastic Optimization Problems

Christopher D. Carroll  
ccarroll@jhu.edu

May 4, 2011

## **Abstract**

These lecture notes sketch a set of techniques that are useful in solving representative agent dynamic stochastic optimization problems. I make no attempt at a systematic overview of the many possible technical choices; instead, the notes present a very specific set of methods and techniques. Associated with these notes is a set of Mathematica programs that solve the problems described in the text. Both text and programs are available on my home page, <http://www.econ.jhu.edu/People/Carroll>. These notes were originally written for my Advanced Topics in Macroeconomic Theory class at Johns Hopkins University.

Department of Economics, Johns Hopkins University, Baltimore, MD 21218-2685,  
410-516-7602 (office), 410-516-7600 (fax).

# 1 Introduction

These notes describe several methods that can be used to solve the stochastic growth model widely used in the macroeconomics literature. The principal differences between this problem (and most other representative-agent macroeconomic dynamic programming problems) and the microeconomic problems that were the subject of the previous lecture notes is that here we assume that the agent has an infinite horizon rather than a finite lifetime; that the aggregate capital stock, and therefore interest rates (and potentially wage rates) are determined by the saving decisions of the representative agent; and that the fundamental structure of the problem does not change from period to period.

The assumption that the problem stays the same from period to period turns out to permit a variety of mathematical tricks to be used that were not available in the finite horizon case. These tricks can substantially reduce the computational burden of solving the problem.

## 2 The Problem

Consider the following problem for a representative agent.

$$\max \quad \mathbb{E}_t \sum_{n=0}^{\infty} \beta^n u(c_{t+n}) \quad (1)$$

$$\text{s.t.} \quad k_{t+1} = F(k_t, \theta_t) - c_t \quad (2)$$

$$\log \theta_{t+1} = \lambda \log \theta_t + \epsilon_{t+1} \quad (3)$$

where the variables are

- $\beta$  – time discount factor
- $k_t$  – aggregate capital stock
- $c_t$  – consumption in period  $s$
- $u(c)$  – utility derived from consumption
- $\theta$  – aggregate productivity
- $\delta$  – depreciation rate of capital
- $\epsilon_t$  – stochastic aggregate productivity shock
- $F(k_t, \theta_t)$  – Aggregate production function

and the productivity shock  $\epsilon_t \sim \mathcal{N}(0, \sigma_\epsilon^2)$ . Finally, assume that the utility function is of the CRRA form,  $u(c) = c^{1-\rho}/(1-\rho)$  and that the aggregate production function is  $F(k_t, \theta_t) = \theta_t k_t^\alpha + (1-\delta)k_t \Rightarrow F^k(k_t, \theta_t) = \alpha \theta k_t^{\alpha-1} + (1-\delta)$ .

### 3 The Usual Theory

#### 3.1 First Order Conditions

The usual theoretical analysis of this problem proceeds as follows.

Bellman's equation for this problem is

$$V_t(k_t, \theta_t) = \max_{\{k_{t+1}\}} u(F(k_t, \theta_t) - k_{t+1}) + \beta \mathbb{E}_t[V_{t+1}(k_{t+1}, \tilde{\theta}_{t+1})] \quad (4)$$

such that

$$\log \theta_{t+1} = \lambda \log \theta_t + \epsilon_{t+1}, \quad (5)$$

the first order condition for which is

$$u'(c_t) = \beta \mathbb{E}_t[V_{t+1}^k(k_{t+1}, \tilde{\theta}_{t+1})], \quad (6)$$

while  $V_t^k$  is given by

$$\begin{aligned} V_t^k(k_t, \theta_t) &= u'(c_t)(F^k(k_t, \theta_t) - (\partial k_{t+1}/\partial k_t)) + \beta \mathbb{E}_t[V_{t+1}^k(k_{t+1}, \tilde{\theta}_{t+1})(\partial k_{t+1}/\partial k_t)] \\ &= u'(c_t)F^k(k_t, \theta_t) + (\partial k_{t+1}/\partial k_t) \underbrace{\left( \beta \mathbb{E}_t[V_{t+1}^k(k_{t+1}, \tilde{\theta}_{t+1})] - u'(c_t) \right)}_{=0 \text{ from (6)}} \end{aligned} \quad (7)$$

$$= u'(c_t)F^k(k_t, \theta_t) \quad (8)$$

and equation (8) is the result of applying the Envelope theorem in this context.

Rolling this expression forward one period yields

$$V_{t+1}^k(k_{t+1}, \theta_{t+1}) = u'(c_{t+1})F^k(k_{t+1}, \theta_{t+1}), \quad (9)$$

and substituting into equation (6) gives us the Euler equation for consumption

$$u'(c_t) = \beta \mathbb{E}_t[F^k(k_{t+1}, \tilde{\theta}_{t+1})u'(c_{t+1})]. \quad (10)$$

A consumption rule  $c(k_t, \theta_t)$  will be a solution of the problem if for every  $\{k_t, \theta_t\}$  it satisfies this equation, i.e. if

$$u'(c(k_t, \theta_t)) = \beta \mathbb{E}_t \left[ F^k \left( \underbrace{F(k_t, \theta_t) - c(k_t, \theta_t)}_{=k_{t+1}}, \tilde{\theta}_{t+1} \right) u' \left( \underbrace{c(F(k_t, \theta_t) - c(k_t, \theta_t), \tilde{\theta}_{t+1})}_{=k_{t+1}} \right) \right]. \quad (11)$$

It will be convenient to define a function

$$\mathfrak{V}_t(k_{t+1}, \theta_t) = \mathbb{E}_t[V_{t+1}(k_{t+1}, \tilde{\theta}_{t+1})] \quad (12)$$

which returns the expected value associated with any given amount of savings. Note that this definition implies that

$$\mathfrak{V}_t^k(k_{t+1}, \theta_t) = \mathbb{E}_t[V_{t+1}^k(k_{t+1}, \tilde{\theta}_{t+1})], \quad (13)$$

or, substituting from equation (9),

$$\mathfrak{V}_t^k(k_{t+1}, \theta_t) = \mathbb{E}_t[F^k(k_{t+1}, \tilde{\theta}_{t+1})u'(c_{t+1}(k_{t+1}, \tilde{\theta}_{t+1}))]. \quad (14)$$

Finally, note that the first order condition (6) can be rewritten as

$$u'(c_t) = \beta \mathfrak{V}_t^k(F(k_t, \theta_t) - c_t, \theta_t). \quad (15)$$

### 3.2 Tricks

As in the lectures on solving microeconomic DSOPs, we will employ some tricks to make the problem easier to solve.

The first is once again to approximate the lognormal distribution for  $\epsilon$  with a discretized  $n$ -point distribution. Formally, define  $n$  points on the  $[0, 1]$  interval as  $\theta = [0, 1/(n-1), 2/(n-1), \dots, 1]$ . Call the inverse of the lognormal distribution  $F_\epsilon^{-1}(p)$ , and define the points  $\theta_i^{-1} = F^{-1}(\theta_i)$ . Then define

$$\epsilon_i = \int_{\theta_{i-1}^{-1}}^{\theta_i^{-1}} \epsilon dF(\epsilon). \quad (16)$$

The  $\epsilon_i$  represent the mean values of  $\epsilon$  in each of the regions bounded by the  $\theta_i^{-1}$  endpoints. Thus we approximate, for example,

$$\mathfrak{V}_t^k(k_{t+1}, \theta_t) = \mathbb{E}_t[F^k(k_{t+1}, \tilde{\theta}_{t+1})u'(c_{t+1}(k_{t+1}, \tilde{\theta}_{t+1}))] \quad (17)$$

$$\hat{\mathfrak{V}}_t^k(k_{t+1}, \theta_t) = \left(\frac{1}{n}\right) \sum_{i=1}^n [F^k(k_{t+1}, \exp(\lambda \log \theta_t + \epsilon_i))u'(c_{t+1}(k_{t+1}, \exp(\lambda \log \theta_t + \epsilon_i)))] . \quad (18)$$

The second important trick is to approximate the pseudoinverse of the marginal value functions. Thus we define

$$\hat{\Lambda}_t^k(k_{t+1}, \theta_t) = [\hat{\mathfrak{V}}_t^k(k_{t+1}, \theta_t)]^{-1/\rho}. \quad (19)$$

### 3.3 The Non-Stochastic Steady-State

Most methods for finding the optimal rules for problems like this one require a candidate “starting point” for a rule, and proceed by improving upon the starting point. A sensible starting point can be obtained from two facts. First, consider a deterministic version of the model in which  $\theta \equiv 1$  so that we can write the rule as a function only of  $k$ , e.g.  $c(k, \theta) = c(k)$ . We know that  $c(0) = 0$ , and, if the steady-state level of capital in the deterministic version of the model is  $\hat{k}$ , the accumulation equation implies

$$\hat{k} = \hat{k}^\alpha + (1 - \delta)\hat{k} - \hat{c} \quad (20)$$

$$\hat{c} = \hat{k}^\alpha - \delta\hat{k} \quad (21)$$

so we can construct an initial guess for  $c(k)$  as a linear interpolation between these two points. That is,  $c(k) = \gamma k$  where

$$\gamma = c(\hat{k})/\hat{k} \quad (22)$$

$$= \hat{k}^{\alpha-1} - \delta \quad (23)$$

and we can calculate the steady-state level of capital  $\hat{k}$  from the Euler equation, which implies that

$$\hat{c}^{-\rho} = \beta(\alpha\hat{k}^{\alpha-1} + (1-\delta))\hat{c}^{-\rho} \quad (24)$$

$$1/\beta = 1 - \delta + \alpha\hat{k}^{\alpha-1} \quad (25)$$

$$1/\beta - (1 - \delta) = \alpha\hat{k}^{\alpha-1} \quad (26)$$

$$\left(\frac{1 - \beta(1 - \delta)}{\alpha\beta}\right) = \hat{k}^{\alpha-1} \quad (27)$$

$$\hat{k} = \left(\frac{1 - \beta(1 - \delta)}{\alpha\beta}\right)^{1/(\alpha-1)} \quad (28)$$

implying that

$$\gamma = \left(\frac{1 - \beta(1 - \delta)}{\alpha\beta}\right) - \delta. \quad (29)$$

The analysis above was carried out under the assumption that  $\theta \equiv 1$ , its nonstochastic value. We need to define a starting function for  $c(k, \theta)$  for all values of  $\theta$ . Because  $\theta$  is highly serially correlated, a reasonable choice is

$$c_T(k, \theta) = \theta\gamma k. \quad (30)$$

## 4 Solution Methods

### 4.1 Solution Method 4.1: Time Iteration

The simplest method of solving this problem makes use of the fact that for any well-behaved ‘guess’ for  $c_{t+1}(k_{t+1}, \theta_{t+1})$  the  $c_t(k_t, \theta_t)$  which solves the Euler equation

$$u'(c_t(k_t, \theta_t)) = \beta \mathbb{E}_t \left[ F^k(F(k_t, \theta_t) - c_t(k_t, \theta_t), \tilde{\theta}_{t+1}) u'(c_{t+1}(F(k_t, \theta_t) - c_t(k_t, \theta_t), \tilde{\theta}_{t+1})) \right] \quad (31)$$

will be ‘closer’ to the true  $c(k, \theta)$  than was  $c_{t+1}(k, \theta)$ .<sup>1</sup> Thus we adopt an initial consumption rule and iteratively solve the maximization problem until the solution has ‘converged.’ This is essentially the same as our method for solving the finite-lifetime maximization problems in the earlier lectures, where we started with a decision rule for the last period of life and worked backwards. The principal difference is that in the finite-life problem it was clear when it was time to stop solving backwards: when the model had been solved back to the first period of life. In the infinite-horizon case, conceptually we want to keep iterating until the consumption rules in successive periods have ‘converged’; that is, until the successive rules are ‘close enough.’ There is no single correct answer to the question of how close is close enough, and in fact there are even several different possible measures of ‘closeness.’

---

<sup>1</sup>Formally, the requirement for this to be true is that the parameter values be such that the problem defines a contraction mapping.

As with the finite-horizon problem, we begin by defining a grid of possible values for  $k$  and  $\theta$ , `kGrid` = {`kMin`, ..., `kMax`} and `thetaGrid` = {`thetaMin`, ..., `thetaMax`} and a list of all combinations of these gridpoints variables indexed by  $i$ , `ArgArray` = {{`kGrid`[[1]],`thetaGrid`[[1]]}, {`kGrid`[[2]],`thetaGrid`[[1]]}, ...}.

We will choose the following somewhat arbitrary definition for convergence: the consumption rules are ‘close enough’ when the mean of the absolute value of the change in consumption measured at the gridpoints, satisfies two conditions,

$$c_t(k, \theta) \text{ has converged if } \begin{cases} |c_{t+1}(k_i, \theta_i) - c_t(k_i, \theta_i)| & \leq 0.0005 \forall i, \text{ and} \\ \max_i |c_{t+1}(k_i, \theta_i) - c_t(k_i, \theta_i)| & \leq .001 \end{cases} \quad (32)$$

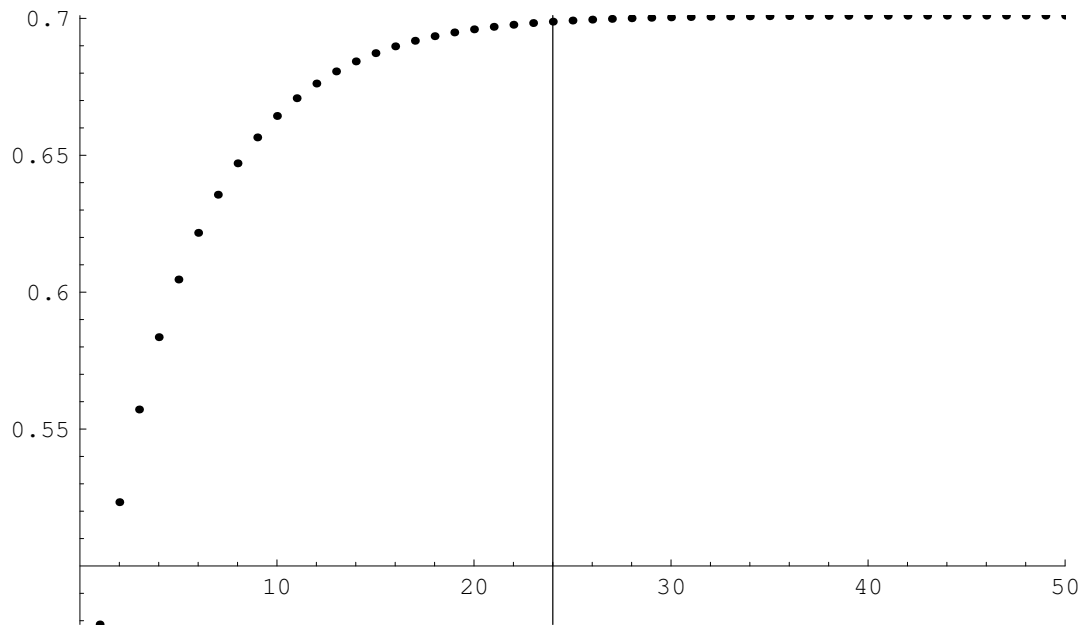
The program `timeiter.m` solves the problem using  $c_T(k, \theta) = \theta\gamma k$  as the starting value of the consumption function.

The iteration algorithm of the program is as follows:

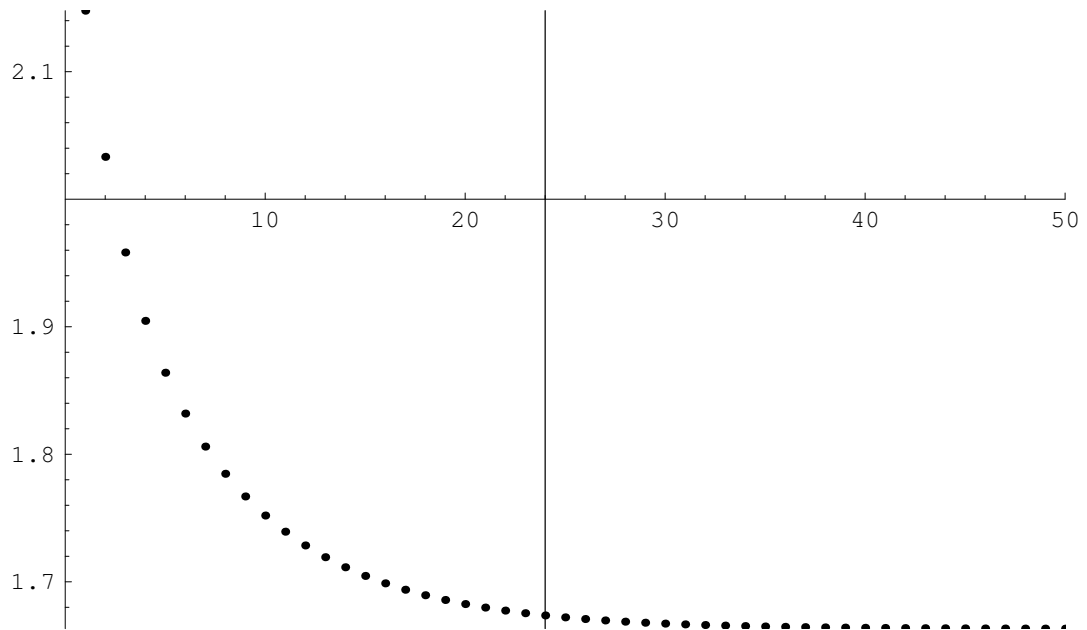
1. Construct `Omegaktp1InvtArgumentList`[[`LifePosn`]] which contains the list of combinations of  $\{k_i, \theta_i\}$  gridpoints.
2. At each of these gridpoints, calculate `Omegaktp1InvtRawResults` where the elements of this list correspond to  $\hat{\Lambda}_i^k$  where  $\hat{\Lambda}_i^k$  is the pseudoinverse of  $\hat{\mathfrak{Y}}_i^k$  as defined in (19) and in the previous lecture.
3. Construct the list `Omegaktp1InvtInterpData`[[`LifePosn`]] =  $\{k_i, \theta_i, \hat{\Lambda}_i\}$
4. Construct an interpolating function for  $\hat{\Lambda}_t(k_{t+1}, \theta_t)$  at the  $\Lambda_i^k$  points, which implicitly constructs a  $\hat{\mathfrak{Y}}_t^k(k_{t+1}, \theta_t)$  function as well
5. Calculate the associated values of  $c_i$  from (??)  
`Interpolation`[[`Omegaktp1InvtInterpData`[[`LifePosn`]]]] command.
6. For each  $i$  solve numerically for the  $c_i$  which satisfies the FOC  $u'(c_i) = \beta \hat{\mathfrak{Y}}_t^k(F(k_i, \theta_i) - c_i, \theta_i)$ , where  $\hat{\mathfrak{Y}}_t^k$  is obtained from the pseudoinverse of  $\hat{\Lambda}_t^k$ .
7. Construct the list `ctValsInterpData`[[`LifePosn`]] =  $\{k_i, \theta_i, c_i\}$
8. Construct the function  $\hat{c}_t(k, \theta) = \text{ctInterpFunc}[[`LifePosn`]] using the command `Interpolation`[[`ctInterpData`[[`LifePosn`]]]]$
9. Calculate whether the consumption function has converged. If so, halt. If not, decrement `LifePosn` and loop back to step 1.

Figure 1 shows the value of consumption evaluated at  $c_{T-s}(\text{kMin}, \text{thetaMin})$  for  $s = \{1, 2, \dots, 50\}$ ; Figure 2 shows the analogous result for the gridpoint  $\{\text{kMax}, \text{thetaMax}\}$ . The vertical line marks the point at which the consumption function has ‘converged’ according to the criteria defined above.

The figures show that the initial consumption function defined in equation (30) yielded less consumption than the amount yielded by the optimal steady-state rule at



**Figure 1**  $c_{T-s}(\underline{k}, \underline{\theta})$  for  $s = \{1, 2, \dots, 50\}$



**Figure 2**  $c_{T-s}(\bar{k}, \bar{\theta})$  for  $s = \{1, 2, \dots, 50\}$

{kMin,thetaMin}. This is no surprise, because the consumption function in equation (30) was defined so it would go through the deterministic optimum *under the assumption that  $\theta$  is constant*. Since the true value of  $\theta$  will revert to its mean value of one, assuming it is stuck at thetaMin is too ‘pessimistic’ and leads to consumption that is lower than optimal. The reciprocal logic applies to the {kMax,thetaMax} consumption rule, which starts too high because it is implicitly assuming that the high value of thetaMax will be maintained forever.

Because time iteration is a contraction mapping, the technique of time iteration is guaranteed eventually to converge on the solution if a solution exists, and each successive rule is guaranteed to be closer to the converged value than its predecessor. Time iteration’s downside is that it converges slowly. One interesting feature of Figures 1 and 2 is that the changes in  $c_t(k_t, \theta_t)$  from one period to the next seem to be fairly predictable, once the first few data points are visible. Your eye can ‘see where it’s going,’ at least approximately, long before it gets there. This observation motivates our first ‘improved’ method of finding the steady-state.

## 4.2 Solution Method 4.2: Time Iteration with Extrapolation

The time path of optimal consumption levels plotted in figures 1 and 2 strongly resembles exponential decay toward some fixed target. Furthermore, there are good mathematical reasons to expect the rules obtained by time iteration to exhibit exponential decay toward the converged rule.<sup>2</sup> In this section we explore whether we can speed up convergence by *assuming* exponential decay and seeing where that leads us.

### 4.2.1 What is Exponential Decay?

We say that a variable  $c_j$  is exhibiting exponential decay towards value  $c^*$  at rate  $\phi$  if it is the case that

$$(c_{j+1} - c^*) = \phi(c_j - c^*) \quad (33)$$

for  $0 < \phi < 1$ .

Note that if we have observations on  $c_j, c_{j+1}, c_{j+2}$  we can use these to estimate  $\phi$  and  $c^*$ :

$$(c_{j+2} - c^*) = \phi(c_{j+1} - c^*) \quad (34)$$

$$(c_{j+2} - \phi c_{j+1}) = c^*(1 - \phi) \quad (35)$$

$$(c_{j+1} - \phi c_j) = c^*(1 - \phi) \quad (36)$$

$$(c_{j+2} - \phi c_{j+1}) = (c_{j+1} - \phi c_j) \quad (37)$$

$$(c_{j+2} - c_{j+1}) = \phi(c_{j+1} - c_j) \quad (38)$$

$$\phi = \frac{(c_{j+2} - c_{j+1})}{(c_{j+1} - c_j)} \quad (39)$$

---

<sup>2</sup>These reasons have to do with the logic of contraction mappings.

and

$$c^* = \frac{(c_{j+2} - \phi c_{j+1})}{1 - \phi} \quad (40)$$

$$= \frac{(c_{j+1} - \phi c_j)}{1 - \phi} \quad (41)$$

Thus the three observations give us an estimate of  $\phi$  and two estimates of  $c^*$ .

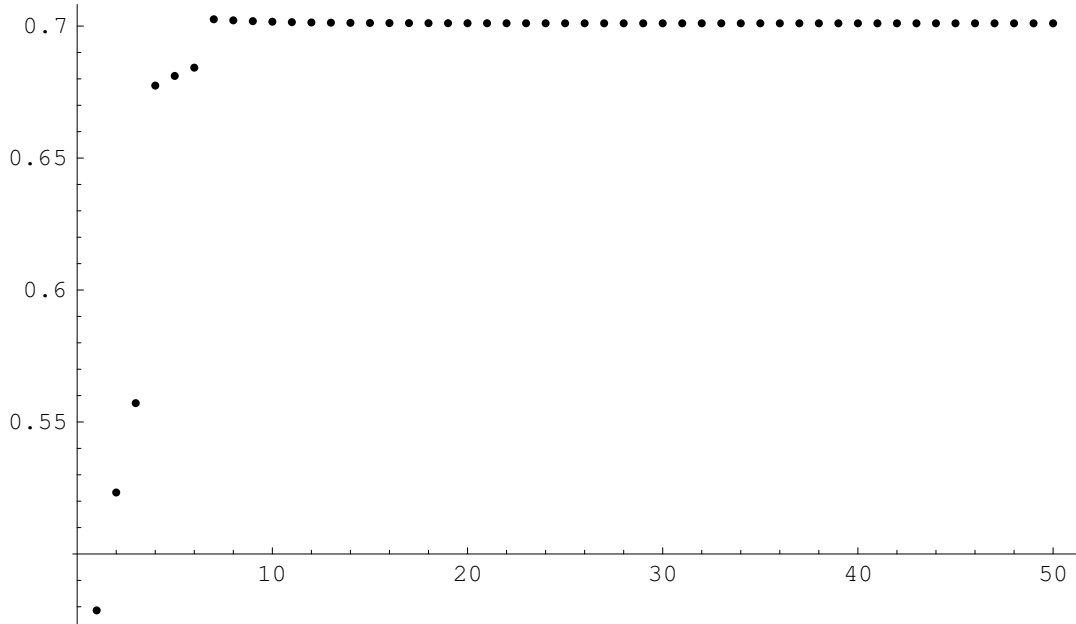
In the case of the stochastic growth model, once we have policy functions in hand for three periods,  $t$ ,  $t - 1$ , and  $t - 2$ , we can use these formulas to find estimates for  $\phi_i$  and  $c_i^*$  at each gridpoint (for our estimate of  $c_i^*$  we use the value that comes from using  $c_{t-1}(k, \theta)$  and  $c_{t-2}(k, \theta)$  because these are the most recently calculated values of the  $c$  function and are less likely to exhibit pathologies that reflect the starting value of the  $c$  function.)

The program `timeiter_extrap.m` implements the extrapolation idea. One of the tricky issues that comes up is figuring out what to do when the consumption rule is converging at some or most gridpoints, but diverging at others. A closely related problem is gridpoints for which the estimate of  $\phi$ , while less than one, is very close to 1. Because the estimate of  $c^*$  that the program makes comes from dividing the most recent change in  $c_t(k_i, \theta_j)$  by  $1/(1 - \phi)$ , if  $\phi$  is estimated to be, say, 0.99, the estimate of  $c^*$  is likely to be far off. Divergence or near-divergence are particularly likely to happen at gridpoints for which the rule is already nearly converged, because for those points the changes in the rule will be mostly noise.

The first response to the ‘nonconvergence’ problem, therefore, is to ignore datapoints for which the difference in consumption rules is already less than the tolerance level. Even when these points are excluded, however, it is likely that the rule will have diverged over the past couple of iterations at some gridpoints. There does not appear to be any clear-cut way to decide what to do at these points; the program simply assigns them all an exponential decay rate of 0.5. Finally, for ‘near-divergence’ points for which  $\phi$  is estimated to be greater than 0.9, we set  $\phi = 0.9$ .

The algorithm of the program is as follows.

1. Time-iterate as in `timeiter.m` for at least three periods to generate  $c_t, c_{t-1}$ , and  $c_{t-2}$ . If during the time iteration the convergence criteria are satisfied by any two successive consumption rules, then halt.
2. Form a list of the estimated values of  $\phi$  at each gridpoint, `ExpDecayRate`
3. At points for which the most recent difference between successive rules was less than `MeanctDiffTolerance`, set `ExpDecayRate[[i]]=0`.
4. At points for which `ExpDecayRate[[i]] > 1`, set `ExpDecayRate[[i]]=.5`
5. At points for which `0.9 < ExpDecayRate[[i]] <= 1`, set `ExpDecayRate[[i]]=.9`
6. Construct `cStarEst` using equation (41)



**Figure 3**  $c_{T-s}(\bar{k}, \bar{\theta})$  Using Time Iteration with Extrapolation

7. Construct an `InterpolatingFunction` object from the estimated values of  $c^*$  at each gridpoint and set  $c_{t-3}(k, \theta)$  equal to this interpolating function.
8. Return to step 1 and continue iteration from period  $t - 4$  backwards.

Figure 3 shows the results of the successive consumption rules evaluated at  $\{k_{\text{Min}}, \theta_{\text{Min}}\}$  as per Figure 1 for the straight time iteration. The ‘exponential jumps’ in the extrapolation method are obviously moving the estimate toward the truth much faster than it approaches the truth using straight time iteration. However, after the dramatic improvement in jumping close to the steady-state very quickly, the extrapolation method does not produce a substantial speedup of the convergence process once the rule is already quite close to the truth. In fact, by the strict convergence criteria defined above, the extrapolation method converges only after 20 periods - only three periods earlier than time iteration. This is because convergence in the near neighborhood of the converged rule is not really well captured by the exponential decay assumption. If we had a less strict convergence criterion, extrapolation would speed up solution considerably more.

### 4.3 Solution Method 4.3: Fixed Point Iteration

A consumption rule  $c(k_t, \theta_t)$  will be a solution to our problem if for every  $\{k_t, \theta_t\}$  it satisfies

$$u'(c(k_t, \theta_t)) = \beta \mathbb{E}_t[(F^k(F(k_t, \theta_t) - c_t(k_t, \theta_t), \tilde{\theta}_{t+1}))u'(c(F(k_t, \theta_t) - c(k_t, \theta_t), \tilde{\theta}_{t+1}))].$$

The method of time iteration (with or without extrapolation) required us to solve this nonlinear equation for a grid of possible values of  $\{k_{i,t}, \theta_{i,t}\}$  yielding a set of  $c_{i,t}$  that satisfied

$$u'(c_{i,t}) = \beta \mathbb{E}_t[(F^k(F(k_{i,t}, \theta_{i,t}) - c_{i,t}, \tilde{\theta}_{i,t+1}))u'(c_{t+1}(F(k_{i,t}, \theta_{i,t}) - c_{i,t}, \tilde{\theta}_{i,t+1})))] \quad (42)$$

In practice, we found that things could be speeded up by constructing  $\hat{\mathfrak{Y}}_t(k_{t+1}, \theta_t)$  and then solving

$$u'(c_{i,t}) = \beta \hat{\mathfrak{Y}}_t(F(k_{i,t}, \theta_{i,t}) - c_{i,t}, \theta_{i,t}). \quad (43)$$

By far the most time-consuming steps in the solution process are, first, constructing the approximating interpolation for  $\mathfrak{Y}_t^k$ , which requires calculating an expectation at every  $\{k_i, \theta_i\}$  gridpoint, and, second, solving (43) using a numerical rootfinding algorithm.

The method of fixed point iteration abandons the economic logic of the Euler equation and for each  $\{k_{i,t}, \theta_{i,t}\}$  finds the  $c_{i,t}$  which satisfies

$$c_{i,t} = \left\{ \beta \mathbb{E}_t[(F^k(F(k_{i,t}, \theta_{i,t}) - c_{t+1}(k_{i,t}, \theta_{i,t}), \tilde{\theta}_{i,t+1}))u'(c_{t+1}(F(k_{i,t}, \theta_{i,t}) - c_{t+1}(k_{i,t}, \theta_{i,t}), \tilde{\theta}_{i,t+1})))] \right\}^{-1/\rho} \quad (44)$$

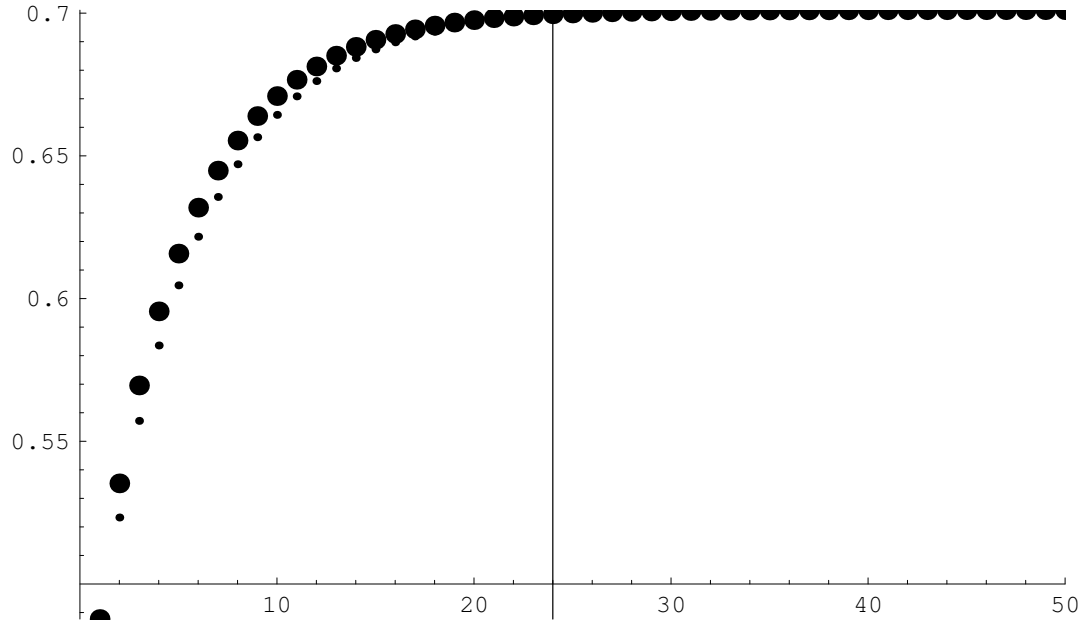
Note that the RHS of this equation no longer involves  $c_{i,t}$  and so we no longer need to solve a nonlinear numerical equation to obtain the next value of  $c_{i,t}$ . Recall now that the reason we needed to construct  $\hat{\mathfrak{Y}}_{i,t}$  was that when searching for the numerical solution to this equation, the rootfinding routine would be ‘trying out’ a great many potential values of  $k_{i,t+1}$ . The construction of  $\hat{\mathfrak{Y}}_t^k$  meant that the entire expectational summation did not have to be performed for every  $k_{i,t+1}$  the program wanted to try out.

Because we no longer need the rootfinding routine, we also no longer need to construct  $\hat{\mathfrak{Y}}_t^k$ , which saves some time. However, the amount of time saved is not enormous because we still have to calculate the expectation on the RHS of (44) at the same gridpoints at which we calculated it to construct  $\hat{\mathfrak{Y}}_t^k$ ; the only time saving comes from the fact that we do not then need to construct the interpolating function.

Thus the method of fixed point iteration eliminates one of the most time-consuming steps in the time-iteration approach, the rootfinding step. While we can no longer interpret the successive consumption rules as having any economic content, if this procedure converges to a rule that satisfies (42) then that rule will be the optimal infinite-horizon rule.

The big potential problem with fixed point iteration is the ‘if’ clause in the previous sentence: there is no guarantee that it will converge. However, it turns out that when the fixed point method diverges, it tends to do so by generating a next step that has gone in the ‘right direction’ (that is, in the direction toward the true converged rule) but has simply overshot. That is, call  $c_{i,t}^*$  the value of  $c$  which solves equation (44) at the gridpoint  $\{k_i, \theta_i\}$ . If  $c_{t+1}(k_i, \theta_i) - c(k_i, \theta_i) = .05$ , then  $c_{i,t}^* - c(k_i, \theta_i)$  might be  $-.07$ .

This problem can generally be fixed by picking some  $c_{i,t}$  which is between the value



**Figure 4**  $c_{T-s}(\underline{k}, \underline{\theta})$  Using Time Iteration versus Fixed Point Iteration

that solves equation (44) and  $c_{t+1}(k_i, \theta_i)$ . That is, the new guess for  $c_{i,t}$  is

$$c_{i,t} = \lambda c_{i,t}^* + (1 - \lambda)c_{t+1}(k_i, \theta_i) \quad (45)$$

where  $\lambda$  controls how far the new  $c_{i,t}$  point moves from its previous value in the direction of the solution to (44).

The program `fix_point.m` solves the problem using fixed point iteration. It turns out that in the particular problem at hand, overshooting is not a problem, but in the program we choose  $\lambda = .9$  so that the machinery is in place to implement a ‘dampening’ method if desired.

The results for `{kMin,thetaMin}` are shown in figure 4. The large dots correspond to the successive values for  $c_{i,t}$  that emerge from the fixed point iteration; for comparison, the smaller dots show the results that emerged from the time iteration procedure. Remarkably, not only is each fixed point iteration much faster to calculate than each time iteration because of the lack of a need to construct  $\hat{\mathfrak{Y}}$  and to solve (44), but each iteration actually finds a  $c_{i,t}$  that is closer to the steady-state than the one found by time iteration (though not by much)! For this problem, fixed point iteration is obviously a much better approach than time iteration to solving the problem. The routine actually converges one period earlier than the time iteration routine.

Note that it is also possible to combine fixed point iteration with an extrapolation method like that described in the previous section to achieve even faster convergence.

## 4.4 Transformation

In the notes on solving the microeconomic optimization problem, the most useful technique was a combination of transformation of the expected value function with the first order conditions that yielded an approximate function in period  $t$  without any rootfinding or maximization. An analogous technique can be used for the representative agent problem, as follows.

Start with the definition of the  $\mathfrak{Y}_t^k$  function,

$$\mathfrak{Y}_t^k(k_{t+1}, \theta_t) = \mathbb{E}_t \left[ F^k(k_{t+1}, \tilde{\theta}_{t+1}) u'(c_{t+1}(k_{t+1}, \tilde{\theta}_{t+1})) \right] \quad (46)$$

$$= \mathbb{E}_t \left[ \left( \alpha \tilde{\theta}_{t+1} k_{t+1}^{\alpha-1} + (1 - \delta) \right) \tilde{c}_{t+1}^{-\rho} \right]. \quad (47)$$

We want to construct a transformed version of this equation, with the transformation taking the form such that if there were no uncertainty the transformed function would be linear in  $c_{t+1}$ . In this case the appropriate transformation is

$$\Lambda^k(k_{t+1}, \theta_t) = \left[ \frac{\mathfrak{Y}_t^k(k_{t+1}, \theta_t)}{\alpha \bar{\theta} k_{t+1}^{\alpha-1} + (1 - \delta)} \right]^{-1/\rho} \quad (48)$$

where  $\bar{\theta} = \mathbb{E}_t[\tilde{\theta}_{t+1} | \theta_t]$ .

We can use our usual method of constructing an interpolating approximation to this function to generate  $\hat{\Lambda}_t^k(k_{t+1}, \theta_t)$ .

The next question is whether we can work backwards from a particular  $\{k_{t+1}, \theta_t\}$  pair to a unique  $c_t$  and  $k_t$  that produce it. It turns out that we can. Start with the first order condition with respect to  $c_t$ :

$$u'(c_t) = \beta \mathfrak{Y}_t^k(k_{t+1}, \theta_t) \quad (49)$$

$$c_t = \beta^{-1/\rho} [\mathfrak{Y}_t^k(k_{t+1}, \theta_t)]^{-1/\rho} \quad (50)$$

$$\beta^{1/\rho} c_t = [\mathfrak{Y}_t^k(k_{t+1}, \theta_t)]^{-1/\rho}. \quad (51)$$

Now for a given  $\theta_t$  the function on the RHS of (51) is an invertible function of  $k_{t+1}$  so define  $\Gamma_t(\mu_t, \theta_t)$  as that inverse; taking  $\Gamma_t$  of both sides of (51) we obtain

$$\Gamma_t(\beta^{1/\rho} c_t, \theta_t) = k_{t+1}. \quad (52)$$

Now we are finally in position to use the key fact that we used in the microeconomic notes: any two values of  $c_t$  and  $k_{t+1}$  that yield the same marginal utility (given a particular  $\theta_t$ ) must be the optimal choice associated with a total amount of resources that is equal to their sum. In other words, the budget constraint must hold:

$$k_{t+1} = \Gamma_t(\beta^{1/\rho} c_t, \theta_t) \quad (53)$$

$$\theta k_t^\alpha + (1 - \delta)k_t - c_t = \Gamma_t(\beta^{1/\rho} c_t, \theta_t) \quad (54)$$

$$\theta k_t^\alpha + (1 - \delta)k_t = \Gamma_t(\beta^{1/\rho} c_t, \theta_t) + c_t \quad (55)$$

Of course, this is still a nonlinear equation, but it is an analytical function with analytical derivatives and so it is numerically quick to solve. We can get an approximate solution immediately by taking a first order Taylor expansion of the production function

around the steady-state level of capital  $\kappa$ ,

$$\theta_t k_t^\alpha \approx \theta_t \kappa^\alpha + \alpha \theta_t \kappa^{\alpha-1} (k_t - \kappa) \quad (56)$$

so, defining  $\gamma \equiv \Gamma_t(\beta^{1/\rho} c_t, \theta_t) + c_t$  we have

$$\theta_t \kappa^\alpha + \alpha \theta_t \kappa^{\alpha-1} (k_t - \kappa) + (1 - \delta) k_t \approx \gamma \quad (57)$$

$$k_t \approx \left( \frac{\gamma + \theta_t \kappa^\alpha (\alpha - 1)}{1 - \delta + \alpha \theta_t \kappa^{\alpha-1}} \right) \quad (58)$$

Thus, the solution procedure is as follows.

1. For each of the  $m$  possible values of the  $\theta$  grid indexed by  $j$ , do the following

- For  $i$  from 1 to the number of elements  $n$  in `kGrid` calculate

$$\Lambda_i^k = \left( \frac{\mathbb{E}_t [F^k(k_i, e^{\log \theta_j + \tilde{\epsilon}}) (c_{t+1}(k_i, e^{\log \theta_j + \tilde{\epsilon}}))^{-\rho}]}{\alpha \theta k_i^{\alpha-1}} \right)^{-1/\rho} \quad (59)$$

and construct an interpolating function `OmegaInvktp1[ktp1, j]` from these

These `InterpolatingFunction` objects are stored in  $n$  separate elements of the `OmegaInvktp1InterpFunc` object, and the separate elements will be combined by the function `Omegaktp1` to generate an appropriate approximating  $\hat{\mathfrak{Y}}_t^k(k_{t+1}, \theta_t)$ .

- Find the unique  $c_i$  points associated with  $\{k_{t+1}, \theta_t\} = \{k_i, \theta_j\}$  using equation (50),

$$c_i = \beta^{-1/\rho} [\mathfrak{Y}_t^k(k_i, \theta_j)]^{-1/\rho} \quad (60)$$

- Find the associated values of beginning-of-period capital in time  $t$ ,  $k_t$ , either by solving the nonlinear equation (56) or approximately using equation (58) (or use a second-order Taylor expansion and find the approximate  $k$  using the quadratic equation).
- Construct the approximate consumption function associated with  $\theta_j$  by interpolation among the  $k_i, c_i$  pairs in hand.

The end result from this process will be an approximated consumption rule for period  $t$ ,  $\hat{c}_t(k_t, \theta_t)$ , and the iteration can proceed.

This process is obviously a bit more complicated than fixed point iteration, but it has one crucial advantage: because the function obtained in period  $t$  is obtained through the budget constraint and first order conditions of the problem, it is guaranteed to be an approximation to the truly correct consumption rule in period  $t$ ; therefore the usual convergence theorems imply that the consumption rules obtained this way are guaranteed to converge, which is not the case for fixed point iteration.

## 4.5 Solution Method 4.5: Using Polynomial Approximation Instead of Interpolation

The solution methods we have explored thus far all involved solving some problem at the set of gridpoints  $\{k_i, \theta_i\}$  and constructing a piecewise linear interpolating function between those solution points. There are many potential alternative schemes for constructing an approximation to a surface. The approximation scheme most familiar to economists is of course the linear regression. If one wanted to approximate the consumption rule defined by the set of points  $c_i, \{k_i, \theta_i\}$  one could do any of a variety of regressions of  $c_i$  on  $k_i, \theta_i$ , their powers, and their cross products. The resulting coefficients would define the three dimensional surface that minimizes the sum of squared differences between the actual  $c_i$  and the forecasted  $\hat{c}_i$ . For example, consider the regression:

$$c_i = a_0 + a_1k_i + a_2k_i^2 + a_3\theta_i + a_4\theta_i^2 + a_5k_i\theta_i. \quad (61)$$

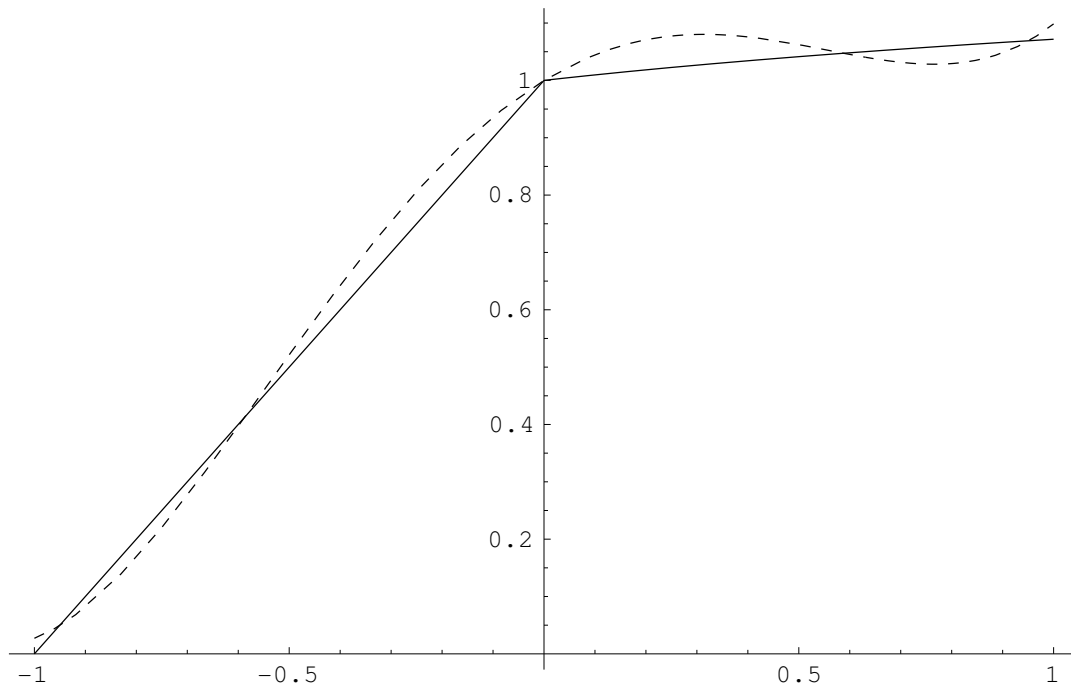
Once the  $a$  coefficients have been estimated, this equation provides a predicted value for  $c$  at any possible combination of  $k$  and  $\theta$  using only six coefficients. Recall that the piecewise linear interpolation characterizes the surface using  $m \times n$  pieces of information (where there are  $m$  gridpoints for  $\theta$  and  $n$  gridpoints for  $k$ ). Thus the coefficient estimates from a regression are likely to be a much more efficient way to characterize a surface. Furthermore, calculating the predicted value from a regression equation is likely to be faster than calculating the results from a piecewise linear interpolation.

The program `timeiter_poly.m` solves the problem using time iteration but substituting the results of a regression on polynomials like that in equation (61) for the piecewise linear interpolation scheme. The results are almost indistinguishable from those obtained using linear interpolation. In practice, as implemented, the polynomial regression approximation scheme is actually a bit slower than the piecewise linear interpolation scheme, but with a little bit of clever programming the polynomial interpolation scheme could probably be made somewhat faster than the piecewise interpolation scheme.

Three points are worth noting about polynomial approximation.

The first is that, at least for this problem, the interpolation step is not the primary computational bottleneck. Thus, even if polynomial approximation were infinitely faster than piecewise linear interpolation the solution to the problem would not speed up very much.

Second, polynomial interpolation can seriously misfire when the surface to be approximated has important nonlinearities. The stochastic growth model specified here is a very ‘smooth’ with no liquidity constraints, only very modest uncertainty, and nothing else to put kinks or bumps or other nonlinearities in the problem, so polynomial approximation works relatively well. But consider the example depicted in figure 5. The function drawn as a solid line resembles a consumption function with a liquidity constraint that begins binding at the point of the kink. Suppose we want to approximate this function with a polynomial. There is an extensive body of theory designed to help find the optimal polynomial approximation. One of the results of



**Figure 5** Polynomial Approximation to a Piecewise Linear Function

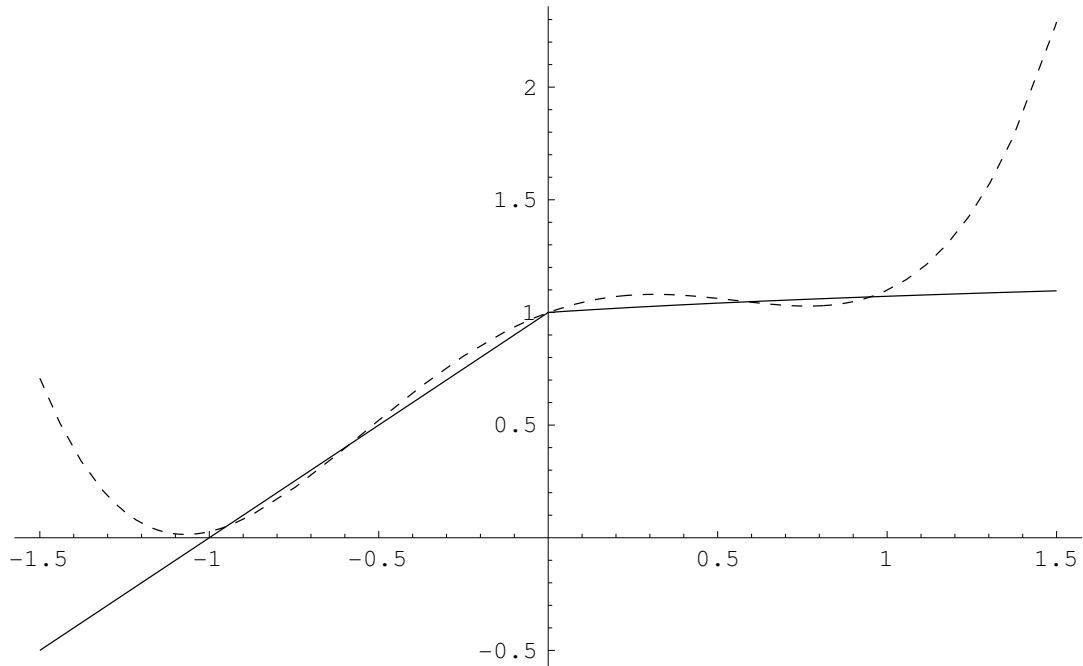
that theory is that a class of polynomials called Chebyshev polynomials do a good job at approximating various functions. There is another theorem, called the Weierstrass theorem, that says a sufficiently high-order polynomial can approximate any continuous function to an arbitrary degree of precision, suggesting that we should choose a fairly high-order polynomial if we want to match the function well. So we try to approximate this function with a fifth order Chebyshev polynomial.<sup>3</sup>

The results of the approximation are graphed as the dashed line. While the Chebyshev approximation does a fairly good job in matching the function over the  $[-1,1]$  range, there are some disturbing features of the approximation, most notably that it is not strictly concave. Well-behaved consumption functions must be strictly concave; consumption functions that are convex, even on a small interval, tend to imply all sorts of pathological behavior like negative precautionary saving (introducing an unavoidable risk makes agents save less).

The third point worth noting about polynomial interpolation is illustrated in the figure 6, which plots the polynomial approximation and the true function outside of the range on which the approximation was constructed. Here is where things really go haywire. Because the polynomial was constructed only to match data in the  $[-1,1]$  interval, it essentially ‘doesn’t care’ how bad its match is to the function outside that

---

<sup>3</sup>This is not a very accurate or detailed description of what is being done; to be somewhat more precise, we are estimating a fourth order polynomial using a collocation method that goes through the zeros of a fifth order Chebyshev polynomial on the  $[-1,1]$  interval. The calculations are all performed in the notebook `ChebyshevBad.nb`. For a rigorous treatment, see Judd 1998.



**Figure 6** Polynomial Approximation Blowup

interval. It turns out that the polynomial which fits the data best happens to shoot off to the moon just beyond the range over which it was fit.

Why does it matter what the function is doing outside of the range over which it has been fit? Well, in dynamic *stochastic* optimization problems, it will always be necessary to evaluate the consumption function, marginal value function, etc. at some points outside of whatever grid one has chosen. To see why, consider trying to calculate the marginal value function at the outermost value of  $k$ . To do so one needs to consider the consequences of all of the possible shocks that might occur. But if there is any serious uncertainty, there will probably be at least a small chance that capital tomorrow will be greater than capital today (if, for instance, the economy experiences the largest possible positive productivity shock). So whatever value one chooses for  $k_{\text{Max}}$ , it will be necessary to evaluate an extrapolated value of the function for some  $k$ 's greater than  $k_{\text{Max}}$ . If the extrapolation goes haywire just beyond  $k_{\text{Max}}$ , then the marginal value calculated at  $k_{\text{Max}}$  will also be haywire. Furthermore, in the prior period, any level of  $k$  such that in the succeeding period capital might equal (or exceed)  $k_{\text{Max}}$  will have a screwed up marginal value as well. Through this mechanism, the corruption of the function outside of the grid will work its way into the grid and can screw up the results even for points well inside the boundaries of the grid.

All of these caveats apply much more to micro problems than to representative agent type macro problems, where the representative agent is assumed to hold large amounts of capital and the production function is assumed to be very smooth. For such smooth problems, polynomial interpolation can work reasonably well. Indeed,

polynomial approximation allows some solution methods that are less feasible using interpolation methods - like the direct use of nonlinear equation solvers.

#### 4.6 Solution Method 4.6: Use a Nonlinear Equation Solver

As noted at the outset in discussing time iteration, any solution method for the infinite horizon problem must specify some criterion to use in determining when it has arrived at a solution that is ‘close enough’ to stop.

Any method of solving the problem involves approximating the consumption function with a finite set of parameters. In the piecewise linear interpolation scheme, the ‘parameters’ were the values  $c_i$  indicating the value of consumption at each of the grid points. In the polynomial approximation scheme, the parameters were the coefficients in the regression of the  $c_i$  on  $k_i, \theta_i$ , and their powers and cross products. For the purposes of this discussion it will be useful to abandon the  $t$  subscripting we have been using heretofore to index the consumption function at time  $t$  and instead to subscript by  $a$  to indicate the set of parameters. Thus what we previously wrote as  $c_t(k_i, \theta_i)$  we now write as  $c_a(k_i, \theta_i)$  where if we wanted a literal translation of, say, the time-iteration scheme we would have that  $a_t = \{c_{1,t}, c_{2,t}, \dots, c_{n,t}\}$  or in the specific polynomial approximation scheme described above  $a = \{a_0, a_1, \dots, a_5\}$ .

We can now define a *residual function* that tells us ‘how close’ a given set of parameters comes to describing the optimal consumption function. For example, define  $c_{i,a}$  from the ‘fixed point’ iteration equation (44)

$$c_{i,a} = \left\{ \beta \mathbb{E}_t[(F^k(F(k_t, \theta_t) - c_a(k_{i,t}, \theta_t), \tilde{\theta}_{t+1}))u'(c_a(F(k_t, \theta_t) - c_a(k_{i,t}, \theta_t), \tilde{\theta}_{t+1})))] \right\}^{-1/\rho} \quad (62)$$

and define the residual function as

$$R(a) = \sum_i (c_a(k_i, \theta_i) - c_{i,a})^2. \quad (63)$$

Thus  $R(a)$  is the sum of squared differences between the consumption implied by consumption rule  $c_a$  and the consumption that emerges from the RHS of the fixed point equation. Since we know that for a converged consumption rule  $c_{i,a} = c_a(k_i, \theta_i)$  if we found a set of parameter values such that this residual function was very small we would know that the consumption rule defined by those parameters was very close to optimal.

There are many alternative choices for residual function - one could use the absolute value of the differences, for instance, or one could use the difference between the RHS and the LHS of the time-iteration Euler equation. We choose to use the fixed point equation here because it performed well in our analysis in section 4.3.

Given a well-designed residual function and a starting point for  $a$ , any good nonlinear optimization routine should be able eventually to find a ‘good’ set of values for  $a$ .

It should be apparent now why I discussed this approach only after presenting the method of polynomial approximation. Polynomial approximation allows us to describe the consumption function surface with a very small number of parameters. In principle this general ‘minimize a residual function’ method could have been used with piecewise interpolation of the consumption surface as well, but in practice the number

of parameters required to describe the shape of the piecewise linearly interpolated consumption function is very large -  $m \times n$ . Since nonlinear equation solvers are in practice able to handle only a modest number of parameters, some more economical way of describing the surface had to be found.

The problem is solved in the program `minresids_poly.m`. In comparison with the other methods examined here, this method performs very poorly. After 924 iterations, the optimizer stops, returning a value  $a$  for which the mean residual value is approximately .003. In contrast, the program `fixpoint.m` converged on a solution for which the mean residual was .0004 after only 23 iterations. Furthermore, it had found a solution for which the mean residual was less than .003 in only 10 periods. By this metric the fixed point iteration program is roughly 100 times more efficient than the general purpose nonlinear optimizer. Furthermore, recall that the fixed point routine could be further sped up by using the extrapolation techniques applied in the time iteration case.

The difference between fixed point iteration and the use of a general-purpose nonlinear optimizer appears to spring from the fact that the nonlinear optimizer does not ‘know’ the direction of the optimum, and so has to try out a lot of guesses. In contrast, the fixed point routine (and for that matter, the time iteration method) ‘know’ the direction of the truth, even if they don’t know precisely how far away it is. A further advantage of these procedures is that they cannot converge on local minima as the nonlinear optimizer can.

## 5 Summary and Conclusion

This document has presented a selection of the many methods available for solving dynamic stochastic general equilibrium representative agent problems. To bring the topic into focus, note that there are essentially four decisions that must be made in order to solve these problems.

1. What method to use for approximating the policy function surface.

We have examined piecewise linear interpolation and polynomial approximation. Polynomial approximation is more efficient but less flexible, and in particular does not deal well with constraints and nonlinearities. However, representative agent problems typically do not involve constraints and nonlinearities, and so polynomial approximation methods may work well. See Judd 1998 for a detailed discussion of methods of choosing polynomials for macro models.

2. How to treat the stochastic shocks.

In the lectures on solving microeconomic problems, we showed that discretizing the continuous distribution function for the stochastic variables produces good results and increases speed enormously. Those points were implicitly carried over into this lecture without further elaboration. For alternative methods of treating the stochastic shocks (in particular, for a discussion of Gauss-Hermite quadrature, again see Judd 1998).

3. How to measure whether our method of searching for the optimal rule has converged.

We have generally relied on methods that compared the consumption rules at successive iterations and stopped when they were ‘close.’ There are many other potential methods of measuring ‘closeness.’ For example, one might want to define closeness on the basis of marginal utility differences yielded by the different rules, or the level of utility.

4. How to proceed from one trial solution to the next, better, trial solution.

This has been the main focus of the lecture. We found that time iteration was reliable and slow, but could be speeded up through extrapolation. We saw that fixed point iteration was much faster and could also be speeded up by extrapolation. Finally, we found that a general purpose method that relies upon a ‘dumb’ nonlinear optimizer was extremely slow and did not find a particularly good solution.

None of the results in these notes is a universal truth. And there are many alternative choices for each of these four items that might work better than any of the methods examined here for some problems. But the methods discussed here are a good starting point for solving such problems. For a much more thorough and comprehensive treatment of all of these issues and many more, see Judd 1998.



## References

JUDD, KENNETH L. (1998): *Numerical Methods in Economics*. The MIT Press, Cambridge, Massachusetts.