1 Discretization of a continuous stochastic process

In many of the problems we studied in the previous chapter, we postulated that agents face a continuous stochastic income process. A typical example would be assuming that income is given by

$$Y_t = \exp(y_t),$$

where y_t follows a first-order autoregressive process of the class

$$y_t = \rho y_{t-1} + \varepsilon_t, \tag{1}$$

where $|\rho| < 1$ and ε_t is a white noise process with variance σ_ε^2. Assume that ε_t is drawn from a distribution G. Let $\Pr\{\varepsilon_t \leq \bar{\varepsilon}\} = G(\bar{\varepsilon}) = F(\bar{\varepsilon}/\sigma_\varepsilon)$, where F is the "standardized" version of $G(\varepsilon_t)$ with unit variance. This $AR(1)$ process for y_t is covariance-stationary with mean zero and variance $\sigma_y^2 = \sigma_\varepsilon^2/(1-\rho^2)$.

Solving the household consumption-saving problem with a continuous shock can be done, but it is very costly, computationally. So it's useful to learn how to approximate discretely a continuous process through a finite-state Markov chain that will mimic closely the underlying process. To discretize the continuous process in (1) we need two ingredients: (i) the points on the finite state space and (ii) the transition probabilities.

1.1 Tauchen's method

Let $\tilde{y}$ be the discrete-valued process that approximates y and let $\{y_1, y_2, ..., y_N\}$ be the finite set of possible realizations of $\tilde{y}$.

Choice of points. Tauchen (1986) suggests to select a maximum value y_N as a multiple m (e.g., $m = 3$) of the unconditional standard deviation, i.e.

$$y_N = m\left(\frac{\sigma_\varepsilon^2}{1-\rho^2}\right)^{\frac{1}{2}},$$

and let $y_1 = -y_N$ (assuming G is symmetric), and $\{y_2, y_3, ..., y_{N-1}\}$ be located in a equi-spaced manner over the interval $[y_1, y_N]$. Denote with d the distance between successive points in the state space.

Transition probabilites. Let

$$\begin{aligned}\pi_{jk} &= \Pr\{\tilde{y}_t = y_k | \tilde{y}_{t-1} = y_j\} = \Pr\{y_k - d/2 < \rho y_j + \varepsilon_t \leq y_k + d/2\} \\ &= \Pr\{y_k - d/2 - \rho y_j < \varepsilon_t \leq y_k + d/2 - \rho y_j\}\end{aligned}$$

be the generic transition probability.

Then, if $1 < k < N-1$, for each j choose

$$\pi_{jk} = F\left(\frac{y_k + d/2 - \rho y_j}{\sigma_\varepsilon}\right) - F\left(\frac{y_k - d/2 - \rho y_j}{\sigma_\varepsilon}\right),$$

while for the boundaries of the interval $k = 1$ and $k = N$ choose:

$$\begin{aligned} \pi_{j1} &= F\left(\frac{y_1 + d/2 - \rho y_j}{\sigma_\varepsilon}\right), \\ \pi_{jN} &= 1 - F\left(\frac{y_N - d/2 - \rho y_j}{\sigma_\varepsilon}\right). \end{aligned}$$

Clearly, as $d \to 0$ (and therefore $N \to \infty$), the approximation becomes better and better until it converges to the true continuous process y_t. It is useful to notice that other integration rules (e.g., Gaussian quadrature or the collocation methods) could lead to a more efficient placement of the points on the the interval $[y_1, y_N]$.

Accuracy of approximation: To assess the adequacy of the approximation, note that the approximate process $\tilde{y}$ admits a representation

$$\tilde{y}_t = \tilde{\rho}\tilde{y}_{t-1} + \tilde{\varepsilon}_t,$$

where $\tilde{\rho} = cov\left(\tilde{y}_{t-1}, \tilde{y}_t\right) / var\left(\tilde{y}_t\right)$ and $\tilde{\sigma}_\varepsilon = \left(1 - \tilde{\rho}^2\right) var\left(\tilde{y}_t\right)$. The unconditional second moments of the $\tilde{y}_t$ distribution can just be computed from the stationary distribution of $\tilde{y}_t$. In turn, the stationary distribution can be computed as follows. Let Θ_N be the transition matrix of dimension $(N \times N)$ with generic element π_{jk}. Then, the stationary distribution π^* solves

$$\pi^* = \Theta_N \pi^* \to \left(I - \Theta_N\right) \pi^* = 0$$

which is a linear system of N equations. Using this method, one can directly compare ρ to $\tilde{\rho}$ and σ_ε to $\tilde{\sigma}_\varepsilon$ and gauge the quality of the approximation. Alternatively, one can simulate a long sample for $\{\tilde{y}_t\}$ (for example a sample of 10,000 draws with a burn-in period of 1,000 draws to avoid initial conditions to affect the outcome), compute some key statistics and compare them with those in the original process.

Multivariate processes: Tauchen describes how to approximate also a multivariate process. This strategy that can be used to approximate higher-order autoregressive processes as well. For example, consider the $AR(2)$ process

$$y_t = \rho_1 y_{t-1} + \rho_2 y_{t-2} + \varepsilon_t.$$

Define a column vector $Z_t = [y_t \quad y_{t-1}]'$. Then one can write the AR(2) process above in multivariate form as

$$Z_t = \begin{bmatrix} \rho_1 & \rho_2 \\ 0 & 1 \end{bmatrix} Z_{t-1} + [\varepsilon_t \quad 0]'.$$

1.2 Rouwenhorst method

This is the best method to discretize a continuous stochastic process, in particular those with very high persistence, near unit root, typical in macro. It is accurate and fast. Consider again the $AR(1)$ process above, but this time we do not need to make any distributional assumption. We want to approximate it through a discrete-space Markov chain $\tilde{y}$ over a symmetric and evenly-spaced state space $\{y_1, y_2, ..., y_N\}$ with $-y_1 = y_N = \psi$. Construct the Markov chain Θ_N recursively as a function of three parameters only (p, q, ψ) as follows:

- **Step 1** For $N = 2$, define Θ_2 as

$$\Theta_2 = \begin{bmatrix} p & 1-p \\ 1-q & q \end{bmatrix}$$

Step 2 For $N \geq 3$, define recursively Θ_N as the $N \times N$ matrix

$$\Theta_N = p \begin{bmatrix} \Theta_{N-1} & \mathbf{0} \\ \mathbf{0'} & 0 \end{bmatrix} + (1-p) \begin{bmatrix} \mathbf{0} & \Theta_{N-1} \\ 0 & \mathbf{0'} \end{bmatrix} + (1-q) \begin{bmatrix} \mathbf{0'} & 0 \\ \Theta_{N-1} & \mathbf{0} \end{bmatrix} + q \begin{bmatrix} 0 & \mathbf{0'} \\ \mathbf{0} & \Theta_{N-1} \end{bmatrix}$$

and note that, for example, for $N = 3$

$$\begin{aligned} \Theta_3 \;=\; & p \begin{bmatrix} p & 1-p & 0 \\ 1-q & q & 0 \\ 0 & 0 & 0 \end{bmatrix} + (1-p) \begin{bmatrix} 0 & p & 1-p \\ 0 & 1-q & q \\ 0 & 0 & 0 \end{bmatrix} \\ & + (1-q) \begin{bmatrix} 0 & 0 & 0 \\ p & 1-p & 0 \\ 1-q & q & 0 \end{bmatrix} + q \begin{bmatrix} 0 & 0 & 0 \\ 0 & p & 1-p \\ 0 & 1-q & q \end{bmatrix} \end{aligned}$$

which shows that the third line sums to 2, which justifies step 3

Step 3 Divide all but the top and bottom rows by 2 so that their elements sum to one.

One can show that this Markov chain converges to the invariant binomial distribution

$$\lambda_i^{(N)} = \binom{N-1}{i-1} s^{i-1} (1-s)^{N-1}$$

where

$$s = \frac{1-p}{2-(p+q)}.$$

Recall that *a binomial distribution gives the discrete probability distribution of obtaining exactly $i-1$ successes out of $N-1$ Bernoulli trials (where the result of each Bernoulli trial is true with probability s and false with probability $1-s$).*

We can compute analytically all the moments of this distribution, for example

$$\begin{aligned}
E(\tilde{y}_t) &= \frac{(q-p)\psi}{2-(p+q)} \\
var(\tilde{y}_t) &= \psi^2\left[1-4s(1-s)+\frac{4s(1-s)}{N-1}\right] \\
Corr(\tilde{y}_t, \tilde{y}_{t+1}) &= p+q-1 \\
E[\tilde{y}_{t+1}|y_k] &= (q-p)\psi+(p+q-1)y_k \\
&\quad etc...
\end{aligned}$$

For any continuous stochastic process $\{y_t\}$ we can compute analytically a number of unconditional and conditional moments and we can use a method of moment estimator to choose the three parameters of the Markov chain (p, q, ψ) that minimize the distance between true and approximated selected moments.

For example, consider the $AR(1)$ process $\{y_t\}$ above and note that

$$\begin{aligned}
E(y_t) &= 0 \rightarrow \frac{(q-p)\psi}{2-(p+q)} = 0 \rightarrow q = p \\
Corr(y_t, y_{t+1}) &= \rho \rightarrow p+q-1 = \rho \rightarrow p = q = \frac{1+\rho}{2} \\
var(y_t) &= \sigma_y^2 \rightarrow \psi^2\left[1-4s(1-s)+\frac{4s(1-s)}{N-1}\right] = \sigma_y^2 \rightarrow \psi = \sigma_y\sqrt{N-1}.
\end{aligned}$$

Therefore, in this example, one can replicate exactly unconditional mean, variance, and first-order auto correlation of the original continuous process. Finally, since we know that the binomial converges to a Normal distribution for $N \rightarrow \infty$, this method is particularly apt at approximating log-normal (in levels) income processes, but it works with other distributions as well.

2 Global Solution Methods for the Income Fluctuation Problem

Local approximation methods, like linear quadratic (LQ) approximations, construct functions that match well the properties of the original function around a particular point. In the stochastic growth model, for example, there is a natural point around which the approximation could be taken: the deterministic steady-state level of capital k^* and the mean of the shock (usually normalized to 1). Fluctuations due to aggregate productivity shocks move the system in a small neighborhood of k^*.

When solving the consumption-saving problem with uncertainty, additional issues arise. First, choosing a particular point on the asset grid where to approximate individual behavior is much less obvious. Second, quantitatively, individual income uncertainty is much larger than aggregate uncertainty, which means that the system moves over a large interval in the state space. Third, in LQ approximations the marginal utility is linear and the role of uncertainty as a motive for saving is artificially reduced (in fact, it completely disappears in absence of borrowing constraints).

For all these reasons, we need to resort to *global* solution methods for the consumption-saving problem. There are several ways to attack the problem. One can iterate directly on the value function (value function iteration), or on the Euler equation (policy function iteration). Moreover, one can choose among three alternative ways to approximate the policy function or the value function: 1) *discretization* means that the state is assumed to take only values on a pre-specified grid; 2) *interpolation* is a procedure that finds a function that goes exactly through the grid points and defines the function outside the grid by interpolating locally between two adjacent grid points; 3) *L^p-approximation* is any procedure that finds a function close to the original one in the sense of a L^p norm, so it's a more global interpolation method.[1]

Consider the problem in its recursive formulation, with states (a, y). In general, with two state variables, we must approximate/interpolate a function in two dimension, which

[1] A distance between two functions evaluated over the N points $\{x_k\}_{k=1}^N$ in the L^p norm is

$$||f(x) - \hat{f}(x)||_p = \left(\sum_{k=1}^{N} |f(x_k) - \hat{f}(x_k)|^p \right)^{1/p}.$$

is a hard problem.[2] This is where the discretization of the income process comes in handy. We maintain that asset holdings a are a continuous variable and discretize the income process. In other words, we need to approximate/interpolate N functions $a'(a, y_j)$, for $j = 1, ..., N$.

Suppose we have already discretized the income process following Tauchen's method. Let $y \in Y$, where y is income in level, and Y is the finite set of values the Markov chain y can take. And let $\pi(y'|y)$ be the discretized transition function. Then, we can write the income fluctuation problem in recursive form as:

$$\begin{aligned} V(a, y) &= \max_{\{c, a'\}} u(c) + \beta \sum_{y' \in Y} \pi(y'|y) V(a', y') \\ &\quad s.t. \\ c + a' &\leq Ra + y \\ a' &\geq -\phi \end{aligned}$$

The Euler equation, once we substitute the budget constraint, reads

$$u_c(Ra + y - a') - \beta R \sum_{y' \in Y} \pi(y'|y) u_c(Ra' + y' - a'') \geq 0.$$

Equality holds if $a' > -\phi$. This is a second-order stochastic difference equation. The objective is to find a decision rule for asset holding $a'(a, y)$, i.e., an invariant function of the states (a, y) that satisfies the Euler Equation. The strategy will be to guess a decision rule for a'', and solve for a'. At the solution, the two policy functions must be identical.

We present here in detail the three variants of the policy-function iteration method: discretization, interpolation, and approximation.

2.1 Discretization

1. Construct a grid on the asset space $\{a_1, a_2, ..., a_M\}$, with $a_1 = -\phi$. The best way to construct the grid varies problem by problem. In general, one must put more points where the policy function is expected to display more curvature (i.e., where it is far from linear). In our case, this happens for values of a near the debt constraint $-\phi$.

[2] If the income process is *iid*, remember that you can reduce the dimensionality of the state space to one variable, cash in hand $x = Ra + y$. In general, whenever you can reduce the dimensionality of the state-space you should do so. The numerical complexity of the problem increases geometrically with the dimension of the state space, e.g., the points on the grid to be evaluated grow from N to N^2 to N^3, and so on.

2. Guess an initial decision rule for a'' on the grid points (i.e., a $M \times N$ matrix):

$$\{\hat{a}_0(a_1, y_1), \hat{a}_0(a_1, y_2), ..., \hat{a}_0(a_1, y_N); \hat{a}_0(a_2, y_1), ..., \hat{a}_0(a_2, y_N); ..., \hat{a}_0(a_M, y_N)\}$$

by making sure that each decision rule maps into a point on the asset grid. The subscript denotes the iteration number, and "0" denotes the first iteration. A reasonable guess for $a''_0(a_i, y_j)$ would be

$$\hat{a}_0(a_i, y_j) = a_i,$$

for example, we know that this exactly true when the utility function is quadratic and shocks follow a random walk.

3. Determine if the liquidity constraint is binding. For each point (a_i, y_j) on the grid, check whether

$$u_c(Ra_i + y_j - a_1) - \beta R \sum_{y' \in Y} \pi(y'|y_j) u_c(Ra_1 + y' - \hat{a}_0(a_1, y')) > 0.$$

If this inequality holds, then it means that the borrowing constraint binds, set $a'_0(a_i, y_j) = a_1$ and repeat this check for the next grid point. If the equation instead holds with the $<$ inequality, it means that there is an interior solution (it is optimal to save, or to borrow less than ϕ, for the household) and we proceed to the next step.

4. Determine $a'_0(a_i, y_j)$. Find the pair of adjacent grid points (a_k, a_{k+1}) such that

$$\begin{aligned}
\delta(a_k) &\equiv u_c(Ra_i + y_j - a_k) - \beta R \sum_{y' \in Y} \pi(y'|y_j) u_c(Ra_k + y' - \hat{a}_0(a_k, y')) < 0 \\
\delta(a_{k+1}) &\equiv u_c(Ra_i + y_j - a_{k+1}) - \beta R \sum_{y' \in Y} \pi(y'|y_j) u_c(Ra_{k+1} + y' - \hat{a}_0(a_{k+1}, y')) > 0,
\end{aligned}$$

This means that $a'_0(a_i, y_j) \in (a_k, a_{k+1})$.[3] Since we only work with points on the grid, set

$$a'_0(a_i, y_j) = \arg \min_{i \in \{k, k+1\}} |\delta(a_i)|$$

5. Check convergence by comparing the guess $\hat{a}_0(a_i, y_j)$ to the solution $a'_0(a_i, y_j)$ and stop if, for each pair (a_i, y_j) on the grid, $a'_0(a_i, y_j) = \hat{a}_0(a_i, y_j)$.

[3] In other words, a_k is smaller than $a^*(a_i, y_j)$ because $u'(c)$ is too small and a_{k+1} is larger than $a^*(a_i, y_j)$ because $u'(c)$ is too large, relative to the optimal choice.

6. If convergence is achieved, stop. Otherwise, go back to point 3 with the new guess $\hat{a}_1(a_i, y_j) = a'_0(a_i, y_j)$.

This is a very simple and fast method because it avoids the calculation of the policy function outside the grid points, but at the same time it can be imprecise, unless M is a very large number, but in this case the method becomes computationally costly.

2.2 Policy Function Iteration with Linear Interpolation

1. Construct a grid on the asset space $\{a_1, a_2, ..., a_M\}$ with $a_1 = -\phi$.

2. Guess an initial vector of decision rules for a'' on the grid points, call it $\hat{a}_0(a_i, y_j)$.

3. For each point (a_i, y_j) on the grid, check whether the borrowing constraint binds, as before. I.e., for each point (a_i, y_j) on the grid, check whether

$$u_c(Ra_i + y_j - a_1) - \beta R \sum_{y' \in Y} \pi(y'|y_j)\, u_c(Ra_1 + y' - \hat{a}_0(a_1, y')) > 0.$$

   If this inequality holds, then it means that the borrowing constraint binds, set $a'_0(a_i, y_j) = a_1$ and repeat this check for the next grid point. If the equation instead holds with the $<$ inequality, it means that we have an interior solution (it is optimal to save, or to borrow less than ϕ, for the household) and we proceed to the next step.

4. For each point (a_i, y_j) on the grid, use a *nonlinear equation solver* to look for the solution a^* of the nonlinear equation

$$u_c(Ra_i + y_j - a^*) - \beta R \sum_{y' \in Y} \pi(y'|y_j)\, u_c(Ra^* + y' - \hat{a}_0(a^*, y')) = 0, \qquad (2)$$

   and notice that the equation solver will try to evaluate many times the function $\hat{a}_0$ outside the grid points for assets $\{a_1, a_2, ..., a_M\}$. Hence, the need for the linear interpolation.

   (a) Assume that, for every value y', the function $\hat{a}_0(a, y')$ is *piecewise linear* between the asset grid points. So, every time the equation solver above requires evaluating the function on a^* which lies between grid points, do as follows.

First, find the pair of adjacent grid points $\{a_k, a_{k+1}\}$ such that $a_k < a^* < a_{k+1}$, and then compute

$$\hat{a}_0(a^*, y') = \hat{a}_0(a_k, y') + (a^* - a_k)\left(\frac{\hat{a}_0(a_{k+1}, y') - \hat{a}_0(a_k, y')}{a_{k+1} - a_k}\right)$$

(b) If the solution of the nonlinear equation in (2) is a^*, then set $a_0'(a_i, y_j) = a^*$ and iterate on the next new grid point.

5. Check convergence by comparing $a_0'(a_i, y_j) - \hat{a}_0(a_i, y_j)$ through some pre-specified norm. For example, declare convergence at iteration n when

$$\max_{i,j}\{|a_n'(a_i, y_j) - \hat{a}_n(a_i, y_j)|\} < \varepsilon$$

for some small number ε which determines the degree of tolerance in the solution algorithm.

6. If convergence is achieved, stop. Otherwise, go back to point 3 with the new guess $\hat{a}_1(a_i, y_j) = a_0'(a_i, y_j)$.

Piecewise linear interpolation is fast, and obviously preserves positivity, monotonicity and concavity. However, the optimal policy obtained is not differentiable everywhere.

2.3 Policy function Iteration with Chebishev Approximation

There are several families of polynomials with good properties for function approximation, e.g., Legendre, Chebyshev, Laguerre, Hermite. In what follows, we use Chebyshev since in most of the common applications, they have the best properties (see below and Judd, chapter 6).

Chebishev Polynomials Suppose we want to approximate a function $f(x)$ over the interval $[-1, 1]$, through a polynomial function

$$\hat{f}(x) = \sum_{p=0}^{N} \kappa_p T_p(x) \tag{3}$$

where N is the order of the polynomial approximation, $T_p(x)$ is called the basis functions, i.e. it is a polynomial of order p, and κ_p are the coefficients that weight the various polynomials.

The Chebishev polynomials are a family of basis functions that is very useful for this type of approximations. They are given by the simple formula

$$T_p(x) = \cos(p \arccos(x)),$$

which can be simply constructed sequentially (please verify) as

$$\begin{aligned} T_0(x) &= 1, \\ T_1(x) &= x, \\ T_{p+1}(x) &= 2xT_p(x) - T_{p-1}(x), \quad \text{for } p > 1. \end{aligned}$$

The polynomial $T_p(x)$ with $p > 0$ has p zeros located at the points $x_k = -\cos\left(\frac{2k-1}{2p}\pi\right)$, with $k = 1, 2, .., p$ and it has$(p+1)$ extrema. All the maxima equal 1 and all the minima equal -1, so the range of the Chebishev polynomials is $[-1, 1]$. It is precisely this property that makes the Chebishev polynomials so useful in the approximation of functions. Note that the Chebishev polynomials satisfy the orthogonality condition among them

$$\sum_{k=1}^{M} T_q(x_k) T_p(x_k) = 0, \quad \text{for } p \neq q,$$

which makes them basis functions.

One can prove that, if the κ_p coefficients are defined as

$$\kappa_p = \frac{\sum_{k=1}^{M} f(x_k) T_p(x_k)}{\sum_{k=1}^{M} T_p(x_k)^2}, \quad p = 0, ..., N$$

then the approximation formula (3) is exact for those x_k equal to every zero of $T_N(x)$. Note that the expression for κ_p is that of an OLS estimator that solves

$$\min_{\{\kappa_p\}_{p=0}^{N}} \left\{ \sum_{k=1}^{M} \left[f(x_k) - \sum_{p=0}^{N} \kappa_p T_p(x_k) \right]^2 \right\}$$

i.e. the vector of κ_p minimizes the sum of squared residuals between the true function and the approximated function.

As we increase the order of the approximating Chebishev polynomial toward $N = \infty$, we get closer to the true function. However, even for relatively small N, this procedure yields very good approximations. In particular, notice that since the T_p functions are all bounded by one in absolute value, if the coefficients κ_p decline fast with p (and they do),

then the largest omitted term in the error has order $(N+1)$. Moreover, $T_{N+1}(x)$ is an oscillatory function with $(N+1)$ extrema distributed smoothly over the interval. This smooth spreading out of the error is a very important property of optimal approximations![4]

The Algorithm

1. Compute the M Chebishev interpolation nodes on the normalized interval $[-1,1]$ which are given by the simple formula

$$x_k = -\cos\left(\frac{2k-1}{2M}\pi\right),\ k=1,...,M,$$

i.e. they are the points where the Chebishev polynomials are exactly equal to zero.

   (a) Fix the bounds of the asset space $\{-\phi, a_{\max}\}$. Transform the Chebishev nodes over $[-1,1]$ into a grid over the $[-\phi, a_{\max}]$ interval, by setting

$$a_k = -\phi + (x_k+1)\left(\frac{a_{\max}+\phi}{2}\right),\ k=1,...,M$$

   In particular note that for $k=M$ and M large, $x_k \simeq 1$ and $a_M \simeq a_{\max}$; for $k=1$ and M large, $x_k \simeq -1$ and $a_k \simeq -\phi$.[5]

2. Guess an initial vector of decision rules for a'' on the grid point, call it $\hat{a}_0(a_i, y_j)$

3. For each point (a_i, y_j) on the grid, check whether the borrowing constraint binds. If not, continue.

4. For each point (a_i, y_j) on the grid, use a nonlinear equation solver to look for the solution a^* of the nonlinear equation

$$u_c(Ra_i + y_j - a^*) - \beta R \sum_{y' \in Y} \pi(y'|y_j)\, u_c(Ra^* + y' - \hat{a}_0(a^*, y')) = 0,$$

   and notice that the equation solver will try to evaluate many times $\hat{a}_0$ off the grid. Here, we use the Chebishev approximation.

[4] The best polynomial approximation is given by the *minimax* polynomial (see Judd, chapter 6) which has the property that the sign of the error term should alternate between points (the so-called Equioscillation Theorem).

[5] To understand this transformation, note that the reverse transformation is

$$x_k = 2\frac{a_k+\phi}{a_{\max}+\phi} - 1.$$

(a) Choose an order $N < M$ for the Chebishev polynomials. Compute the Chebishev coefficients

$$\kappa_p^0 = \frac{\sum_{k=1}^{M} \hat{a}_0 (a_k, y_j) T_p (x_k)}{\sum_{k=1}^{M} T_p (x_k)^2}, \; p = 0, ..., N \tag{4}$$

(b) Use the coefficients κ_p^0 for the evaluation outside the grid as follows:

$$\hat{a}_0 (a^*, y') = \sum_{p=0}^{N} \kappa_p^0 T_p \left(2 \frac{a^* + \phi}{a_{\max} + \phi} - 1 \right).$$

(c) If the solution of the nonlinear equation in (2) is a^*, then set $a_0' (a_i, y_j) = a^*$ and iterate on a new grid point from step 4.

5. To check convergence, in this case, it is better to compare iteration after iteration the Chebishev coefficients in order to get a sense of how globally distant are the policy functions obtained as a solution in successive iterations. For example, at iteration n, one could use criterion

$$\min_p |\kappa_p^n - \kappa_p^{n-1}| < \varepsilon.$$

As explained, the Chebishev approximation can be extremely good. It is smooth and differential everywhere, however keep in mind that it may not preserve concavity.

2.4 The Endogenous Grid Method

This is a newly developed method that, when it can be applied (not always), is much faster than the traditional method just described because it does not require the use of a nonlinear equation solver. The essential idea of the method is to construct a grid on a', next period's asset holdings, rather than on a, as is done in the standard algorithm. The method also requires the policy function for consumption to be at least weakly monotonic in current asset holdings.

Consider the above income fluctuation problem and write the Euler equation as follows:

$$u_c (c (a, y)) \geq \beta R \sum_{y' \in Y} \pi (y'|y) u_c (c (a', y')).$$

Equality holds if $a' > -\phi$. We now solve for the consumption policy instead that for the next-period assets policy, i.e. we look for a decision rule for consumption $c (a, y)$, which

is an invariant function of the states that satisfies the Euler Equation and that does not violate the borrowing constraint. We are going to start from a guess $\hat{c}_0(a, y)$ and will iterate on the Euler Equation until the decision rule for consumption that we solve for is essentially identical to the one in the previous iteration. The algorithm involves the following steps:

1. Construct a grid on (a, y) where $a \in G_A = \{a_1, ..., \bar{a}\}$ with $a_1 = -\phi$ and $y \in Y = \{y_1, ..., y_N\}$.

2. Guess a policy function $\hat{c}_0(a_i, y_j)$. A good initial guess is to set $\hat{c}_0(a_i, y_j) = ra_i + y_j$ which is the solution under quadratic utility if income follows a random walk.

3. Here's the key difference. Instead of iterating over pairs of $\{a_i, y_j\}$, we iterate over pairs $\{a'_i, y_j\}$. Fix y_j and iterate over all values of a'_i on the grid. For any pair $\{a'_i, y_j\}$ on the mesh $G_A \times Y$ construct the RHS of the Euler equation [call it $B(a'_i, y_j)$]

$$B(a'_i, y_j) \equiv \beta R \sum_{y' \in Y} \pi(y'|y_j) u_c(\hat{c}_0(a'_i, y'))$$

   where the RHS of this equation uses the guess $\hat{c}_0$.

4. Use the Euler equation to solve for the value $\tilde{c}(a'_i, y_j)$ that satisfies

$$u_c(\tilde{c}(a'_i, y_j)) = B(a'_i, y_j)$$

   and note that it can be done analytically, i.e. for $u_c(c) = c^{-\gamma}$ we have $\tilde{c}(a'_i, y_j) = [B(a'_i, y_j)]^{-\frac{1}{\gamma}}$. This is the step where the algorithm becomes much more efficient than the traditional one. First, we do not require a nonlinear equation solver, which takes a lot of computing time and could introduce numerical instabilities. Second, we only compute the expectation in step 3 once. In the traditional algorithm the expectation is computed within the nonlinear equation solver multiple times.

5. From the budget constraint, solve for $a^*(a'_i, y_j)$ such that

$$\tilde{c}(a'_i, y_j) + a'_i = Ra^*_i + y_j$$

   which implicitly gives the function $c(a^*_i, y_j) = \tilde{c}(a'_i, y_j)$. $a^*(a'_i, y_j)$ is the value of assets today that would lead the consumer to have a'_i assets tomorrow if his income shock was y_j today. Note that this function in general is not defined on the grid points of G_A. This is the endogenous grid and it changes on each iteration.

6. Let a_1^* be the value of asset holdings that induces the borrowing constraint to bind next period, i.e., the value for a^* that solves that equation at the point a_1', the lower bound of the grid.

7. Now we need to update our guess. To get the new guess $\hat{c}_1(a_i, y_j)$ on grid points $a_i > a_1^*$ we can use simple interpolation methods using values for $\left\{c(a_n^*, y_j), c\left(a_{n+1}^*, y_j\right)\right\}$ on the two most adjacent values $\left\{a_n^*, a_{n+1}^*\right\}$ that include the given grid point a_i. It is possible that some points a_i are beyond a_N^*, the upper bound of the endogenous grid. Then, we just use linear interpolation methods to obtain a new guess of the policy $\hat{c}_1$ on those grid points. To define the new guess of the consumption policy function on grid values $a_i < a_1^*$, then we use the budget constraint

$$\hat{c}_1(a_i, y_j) = Ra_i + y_j - a_1'$$

since we cannot use the Euler equation as the borrowing constraint is binding for sure next period. The reason is that we found it was binding at a_1^*, therefore a fortiori it will be binding for $a_i < a_1^*$.

8. Check convergence by comparing $\hat{c}_{n+1}(a_i, y_j)$ to $\hat{c}_n(a_i, y_j)$. For example declare convergence at iteration $n+1$ when

$$\max_{i,j}\left\{|\hat{c}_{n+1}(a_i, y_j) - \hat{c}_n(a_i, y_j)|\right\} < \varepsilon$$

for some small ε which determines the degree of tolerance in the solution algorithm.

3 Simulation

Once we solved for the policy function $a'(a, y)$ and obtained the consumption function $c(a, y)$ residually from the budget constraint (or viceversa if using the endogenous grid method), we can simulate paths for the variables of interest and compute statistics like the saving rate, the variance of consumption, the correlation between consumption change and income change, etc... This also allows us to do comparative statics exercises, i.e., asking questions such as what happens to average savings when uncertainty rises? Or, how do savings respond to changes in the interest rate? And so on.

One key step in the simulation is drawing paths of the income process. Suppose, for simplicity that we have a 2-state Markov chain with transition probabilities $\{\pi_{LL}, \pi_{LH}, \pi_{HL}, \pi_{HH}\}$ over two realizations (y_L, y_H). Then, we implement the following algorithm:

1. Draw a long sequence of T realizations from a uniform distribution on $[0,1]$, i.e. $\{\alpha_1, \alpha_2, ..., \alpha_T\}$. Note that $\pi_{sL} + \pi_{sH} = 1$, with $s \in \{L, H\}$.

2. Start from $y_0 = y_L$. If $\alpha_1 < \pi_{LL}$, then set $y_1 = y_L$. Otherwise, set $y_1 = y_H$. Suppose $\alpha_1 > \pi_{LL}$ so that $y_1 = y_H$. Next, if $\alpha_2 < \pi_{HL}$, then set $y_2 = y_L$. Otherwise, set $y_2 = y_H$. And continue until you have mapped the α-sequence into a y-sequence.

To simulate series for consumption and asset holdings, start from an initial condition (a_0, y_0) and, from the policy functions, compute

$$\begin{aligned} a_1 &= a'(a_0, y_0), a_2 = a'(a_1, y_1), ... \\ c_1 &= c(a_0, y_0), c_2 = c(a_1, y_1), ... \end{aligned}$$

One important trick to keep in mind when doing comparative statics (e.g., to study the effect of a rise in r on the saving rate) is that the statistics of interest (e.g., average savings) must be computed based on simulations run *with the same seed of the random number generator for the* α sequence, i.e., one has to use the same paths for the shocks in order to minimize the simulation variance.

It is useful to discard a certain number of initial observations (the so called burn-in period) to avoid dependence of the result from the initial condition (a_0, y_0) and allow draws to occur from the stationary distribution.

4 Checking Accuracy of the Numerical Solution

Suppose that we are using the piecewise linear interpolation. How can we determine when the solution is accurate enough that no more points in the grid are needed? Or, if we are using the Chebishev approximation method, how do we determine that increasing the order of the polynomial would not lead to any significant improvement in accuracy?

4.0.1 den Haan-Marcet Test

Den Haan and Marcet (1994) devise a simple test based on Hansen J test of overidentifying restrictions. We present here the test applied to the consumption-saving problem. The consumption Euler equation

$$u_c(c_t) = \beta R E_t [u_c(c_{t+1})]$$

implies that the residual

$$\varepsilon_{t+1} = u_c(c_t) - \beta R u_c(c_{t+1})$$

should not be correlated with any variable dated t and earlier, since the expectation at time t is conditional on everything observable up to then. Therefore, we should have, for every t

$$E_t[\varepsilon_{t+1} \otimes h(z_t)] = 0, \tag{5}$$

where the symbol $\otimes$ denotes element-by-element product. The term $h(z_t)$ is a $(r \times 1)$ vector of functions of z_t, which can include all the variables in the information set of the agent at time t like $\{y_j, c_j, a_j\}_{j=0}^{t}$. Clearly, this is true only for the *exact* solution. A badly approximated solution will not satisfy this property. This is the key idea of the test proposed by den Haan and Marcet.

One can obtain an estimate of the LHS of (5) through a simulation of length S of the model and the construction of

$$B_S = \frac{\sum_{t=1}^{S} \hat{\varepsilon}_{t+1} \otimes h(\hat{z}_t)}{S},$$

where $\hat{\varepsilon}_{t+1}$ and $\hat{z}_t$ are the simulated counterparts. It can be shown that, under mild conditions, $\sqrt{S}B_S \xrightarrow{d} N(0, V)$, and one can construct the appropriate quadratic form for the test statistics

$$S B_S' \hat{V}_S^{-1} B_S \xrightarrow{d} \chi_r^2$$

where $\hat{V}_S^{-1}$ is the inverse of some consistent estimate of V.

Some remarks are in order. First, the test does not require any knowledge of the true solution, which is a big advantage. Second, given a certain level of approximation in the solution, we can always find a number S large enough so that the approximation fails the accuracy test. This is not a problem when comparing solution methods, since one can fix the same S for both, or one can look for the smallest S such that the method fails the test and compare these thresholds. However, when we want to judge the accuracy of our unique solution, how large should S be? It's not clear: Den Haan and Marcet pick S to be 20 times larger than the typical sample period available.

4.0.2 Euler Equation Error Analysis

The numerically approximated Euler equation is

$$u_c(c_t) - \beta R E_t[u_c(c_{t+1})] \simeq 0.$$

The above equation will be very close to zero when evaluated on gridpoints, but it may be quite far from zero outside the grid.

One can define the relative approximation error ε_t as that value such that the equation holds exactly at t

$$u_c\left(c_t\left(1-\varepsilon_t\right)\right)=\beta RE_t\left[u_c\left(c_{t+1}\right)\right]$$

Let g denote the inverse function of the marginal utility, then we have

$$\varepsilon_t=1-\frac{g\left(\beta RE_t\left[u_c\left(c_{t+1}\right)\right]\right)}{c_t}.$$

For example an error of 0.01 means that the agent is making a mistake equivalent to \$1 for every \$100 consumed when choosing consumption and saving in period t. Santos (2000) proves that, the implied cost in terms of household welfare is the square of the Euler equation error, i.e., the error in the value function is of the order ε^2.

By running a long (but finite) simulation, one can compute a string of Euler equation errors and report either the maximum or the average of the errors in absolute value. This procedure may not explore the entire state space, but it will explore the region of the state space where the agent is more likely to find herself.

Alternatively, one can explore the entire state space by choosing a grid for assets different from the one used in the computation and calculate for each point a_i on this new grid, and for every y_j the Euler equation error $\varepsilon\left(a_i,y_j\right)$. Let $a^*\left(a,y\right)$ be the policy function obtained in the numerical solution. Then, compute:

$$u_c\left(\left(Ra_i+y_j-a^*\left(a_i,y_j\right)\right)\left(1-\varepsilon\left(a_i,y_j\right)\right)\right)-\beta RE_y\left[u_c\left(Ra^*\left(a_i,y_j\right)+y'-a^*\left(a^*\left(a_i,y_j\right),y'\right)\right)\right]=0.$$

Usually, absolute errors are reported in base 10 logarithm, so a value of -2 means an error of \$1 for every \$100, a value of -3 means an error of \$1 for every \$1,000, and so on. Acceptable average errors should be around values of 4 and above. See Aruoba, Fernandez-Villaverde and Rubio-Ramirez (2006) for a comparison of various global solution methods of the growth model based on Euler Equation error analysis.