University of Minnesota
8107 Macroeconomic Theory, Spring 2008, Mini2
Fabrizio Perri
Notes on numerical methods for income fluctuations problems

0.1 Tauchen's method to approximate a continuous income 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

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

with ε_t being an *iid* shock with distribution G, mean zero, and variance σ_ε.

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 a continuous process through a finite-state Markov chain that will mimic closely the underlying process. To approximate the process (1) we need two ingredients: the points on the state space and the transition probabilities. If the Markov Chain we want to use is a simple two-state chain then a simple method of moments can be used. Symmetry in the original process implies that the Markov chain for log-income can be written as the vector $[-z, +z]$ and that the transition probability matrix has the form

$$\begin{bmatrix} p & 1-p \\ 1-p & p \end{bmatrix}$$

so we need simply to find the value for the parameters z and p.These can be simply found by equating the variance and the serial correlation in the VAR and in the Markov Chain i.e.

$$\begin{aligned} \sigma^2_{AR} &\equiv \frac{\sigma^2_\varepsilon}{1-\rho^2} = z^2 \equiv \sigma^2_{MC} \\ E_{AR}(y_t y_{-1}) &\equiv \rho\frac{\sigma^2_\varepsilon}{1-\rho^2} = (2p-1)z^2 \equiv E_{MC}(y_t y_{-1}) \end{aligned}$$

which yields

$$z = \sqrt{\frac{\sigma^2_\varepsilon}{1-\rho^2}}, p = \frac{1+\rho}{2}$$

In general though we might want to approximate a VAR with a Markov Chain with many states (this is for example very useful in asset pricing problems) and in this case we commonly used a procedure developed by Tauchen (1986). Let $\Pr\{\varepsilon_t \leq \bar{\varepsilon}\} = G(\bar{\varepsilon}) = F\left(\bar{\varepsilon}/\sqrt{\sigma_\varepsilon}\right)$, where F is the "standardized" version of $G(\varepsilon_t)$ with unit variance. Let $\tilde{y}$ be the discrete-valued process that approximates y and let $\{\bar{y}_1, \bar{y}_2, ..., \bar{y}_N\}$ be the finite set of possible realizations of $\tilde{y}$.

Tauchen suggests to select values $\bar{y}_j$ so that $\bar{y}_N$ is a multiple m (e.g., $m = 3$) of the unconditional standard deviation, i.e.

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

and let $\bar{y}_1 = -\bar{y}_N$ (assuming G is symmetric), and $\{\bar{y}_2, \bar{y}_3, ..., \bar{y}_{N-1}\}$ be located in a equispaced manner over the interval $[\bar{y}_1, \bar{y}_N]$. Denote with d the distance between successive points in the state space. Let

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

be the generic transition probability.

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

$$\pi_{jk} = F\left(\frac{\bar{y}_k + d/2 - \rho\bar{y}_j}{\sqrt{\sigma_\varepsilon}}\right) - F\left(\frac{\bar{y}_k - d/2 - \rho\bar{y}_j}{\sqrt{\sigma_\varepsilon}}\right),$$

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

$$\begin{aligned}\pi_{j1} &= F\left(\frac{\bar{y}_1 + d/2 - \rho\bar{y}_j}{\sqrt{\sigma_\varepsilon}}\right), \\ \pi_{jN} &= 1 - F\left(\frac{\bar{y}_N - d/2 - \rho\bar{y}_j}{\sqrt{\sigma_\varepsilon}}\right).\end{aligned}$$

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 numerical integration rules (e.g., Gaussian quadrature) could lead to a more efficient placement of the points on the the interval $[\bar{y}_1, \bar{y}_N]$.

Multivariate processes: Tauchen describes how to approximate also a multivariate process of the form

$$y_t = Ay_{t-1} + \varepsilon_t$$

where y_t is now a $nx1$ vector, A is a nxn matrix and ε_t is a $nx1$ vector of i.i.d random variables with mean 0 and variance-covariance matrix Σ. The key insight in representing such a multivariate process is to recognize that the variance covariance matrix (which is symmetric) can be represented as

$$\Sigma = Q\Lambda Q'$$

where Λ is a diagonal matrix and Q is the matrix of eigenvectors of Σ as columns (remember that since Σ is symmentric $Q' = Q^{-1}$) so we can write the process as

$$\begin{aligned}y_t &= Ay_{t-1} + \varepsilon_t \\ Q'y_t &= Q'AQQ'y_{t-1} + Q'\varepsilon_t \\ \tilde{y}_t &= \tilde{A}y_{t-1} + \tilde{\varepsilon}_t\end{aligned}$$

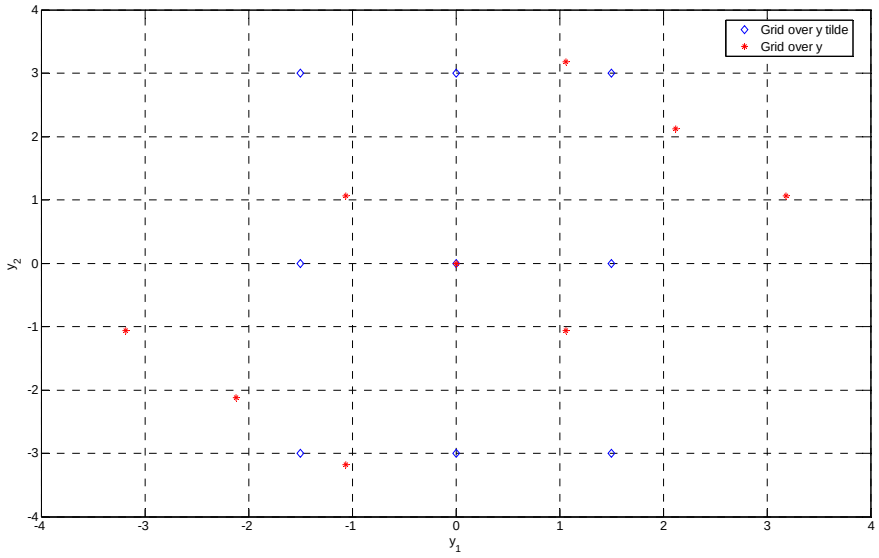

Figure 1:

where $\tilde{y}_t = Qy_t$, $\tilde{A}$ =$Q'AQ$ and $\tilde{\varepsilon}_t$ has a diagonal variance covariance matrix $Q'\Sigma Q = \Lambda$.We can then approximate $\tilde{y}_t$ with a Markov chain (since the ε are uncorrelated doing so is a straightforward extension of the procedure described for the univariate case) and let $\tilde{Y}$ be the nxm matrix containing the states of the Markov chain which approximate $\tilde{y}_t$ (here we assume that the Markov chain has the same number of states (m) for each of the n variables). Then the states of the Markov chain which approximate y_t are simply given by the matrix $Y = Q\tilde{Y}$. As an example consider a variance a biviariate i.i.d proces $y_t = \varepsilon_t$ where ε_t has variance covariance matrix given by

$$\Sigma = \begin{bmatrix} 1 & .8 \\ .8 & 1 \end{bmatrix} = \begin{bmatrix} -0.7071 & .7071 \\ .7071 & 0.7071 \end{bmatrix} \begin{bmatrix} .2 & 0 \\ 0 & .8 \end{bmatrix} \begin{bmatrix} -0.7071 & .7071 \\ .7071 & 0.7071 \end{bmatrix} = Q\Lambda Q'$$

using the transformation above the figure plots the grids obtained for y_t and for $\tilde{y}_t$

Notice that this strategy that can be used to approximate higher-order autoregressive processes as well. For example, consider the $AR(2)$ process

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

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

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

0.2 Global Solution Methods for the Income-Fluctuation Problem

Local approximation methods, like linear quadratic (LQ) approximation or log-linearization, construct functions that match well the properties of the desired object (say an optimal decision rule or an equilibrium price) 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, these considerations are no longer valid. First, choosing a particular point on the asset grid where to approximate individual behavior is much less obvious. In fact, depending on whether $\beta(1+r)$ is less, equal or greater than 1 we can have either one steady state (in which wealth is equal to the borrowing constraint) a continuum of those or none. 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. Let's say that our object of interest is the optimal saving rule $a'(a,y)$ (where a is current wealth and y is current income). There are several ways to attack the problem. First one has to decide the class of functions in which in want $a'(a,y)$ to lie. One possibility is to assume that a' (and hence a) can take only a finite, prespecified values on a grid: this method is called discretization. Another possibility is to assume that the optimal function lies in the class of piecewise linear functions on a grid for a (see below) or another possibility is to assume that the function can be described as the linear combination of special polynomials. Notice that with all this representation we reduce the problem of approximating a generic function (an infinite dimensional object) with something that can be described by a finite set of numbers (e.g. the number of values the function can take on a grid). The next step is to find the numbers that chracterize the function. These can be found either by using *iterative* methods (either on the value function or on the policy function) or using non-iterative methods that involve looking directly for the parameters characterizing the solution solving a minimization problem. In the remainder of the note we will present some examples of these methods.

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 is a computationally burdensome problem.[1] This is where the Tauchen's method comes in handy. We maintain that asset holdings a are a continuous variable and discretize the income process. In other words, we need to find N functions $a'(a,y_j)$, for $j=1,...,N$.

[1] 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.

Suppose we have already discretized the income process following Tauchen's method. Then, we can write the income fluctuation problem in recursive form as:

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

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

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

Equality holds if $a' > -\bar{a}$. This is a second-order stochastic difference equation. The objective is to find a decision rule for next period asset holding $a'(a,y)$, i.e., an invariant function of the states (a,y) that satisfies the Euler Equation.We first discuss dicretization (which works well in simple problems but is numerically very costly in more complicated problems) and then discuss more efficient way of representing the solution.

0.2.1 Discretization

1. Construct a grid on the asset space $\{a_1, a_2, ..., a_M\}$, with $a_1 = -\bar{a}$. 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 away from linear). In our case, this happens for values of a near the debt constraint $-\bar{a}$.

2. Guess an initial vector of decision rules for a'' on the grid points:

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

   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

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

   If this inequality holds, then it means that the borrowing constraint binds and $a'_0(a_i,y_j) = a_1$. If it does you found the optimal asset for this grid point, if it does not continue to the next grid point.

4. Determine $a_0'(a_i, y_j)$. Find the pair of adjacent grid points (a_k, a_{k+1}) such that

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

This means that $a_0'(a_i, y_j) \in (a_k, a_{k+1})$.[2] 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, $a_0'(a_i, y_j) = \hat{a}_0(a_i, y_j)$.

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 (especially if you have multi-dimensional state space)

0.2.2 Piecewise linear approximation with policy function iteration and exogenous grid

Let's now assume that instead of discretizing we search for the solution for our decision rule within the class of piecewise linear functions. As a simple background on piecewise linear function consider $f(x)$ a real univariate function which maps the real line into the real line. A piecewise linear approximation of the function is completely described by a set of increasing values for $x = \{x_1, x_2, ...x_N\}$ and by the corresponding values of the function $\{f(x_1), f(x_2), ...f(x_N)\}$. In particular we can define the piecewise linear approximation of $f, \tilde{f}$ as

$$\tilde{f}(x) = \sum_{j=1}^{N} \phi_j(x) f(x_j)$$

[2] 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.

where

$$\phi_1(x) = \begin{cases} \frac{x_2-x}{x_2-x_1} \text{ if } x \leq x_2 \\ 0 \text{ otherwise} \end{cases}$$

$$\phi_2(x) = \begin{cases} \frac{x-x_1}{x_2-x_1} \text{if } x \leq x_2 \\ \frac{x_3-x}{x_3-x_2} \text{ if } x_2 \leq x \leq x_3 \\ 0 \text{ otherwise} \end{cases}$$

$$\phi_j(x) = \begin{cases} \frac{x-x_{j-1}}{x_j-x_{j-1}} \text{ if } x_{j-1} \leq x \leq x_j \\ \frac{x_{j+1}-x}{x_{j+1}-x_j} \text{ if } x_j \leq x \leq x_{j+1} \\ 0 \text{ otherwise} \end{cases}$$

$$\phi_{N-1}(x) = \begin{cases} \frac{x-x_{N-1}}{x_{N-2}-x_{N-1}} \text{ if } x_{N-2} \leq x \leq x_{N-1} \\ \frac{x_{N-1}-x}{x_N-x_{N-1}} \text{if } x \geq x_{N-1} \\ 0 \text{ otherwise} \end{cases}$$

$$\phi_N(x) = \begin{cases} \frac{x-x_{N-1}}{x_N-x_{N-1}} \text{if } x \geq x_{N-1} \\ 0 \text{ otherwise} \end{cases}$$

those ϕ functions are sometimes called basis functions. Notice that $\tilde{f}(x)$ is just a convenient (for computers) representation of standard linear approximation and that the approximated function, even though the grid is finite, is defined on the entire real line. Also notice that for every x there are at most two basis functions that are positive. This property is particularly useful if one wants to extend the approximation to many dimensions. For example suppose you are interested in approximating the function $f(x,y)$: a piecewise linear approximation on the grid $\{x_1, ...x_N\} \otimes \{y_1, ...y_M\}$ is a straightforward extension of the univariate case

$$\tilde{f}(x,y) = \sum_{j=1}^{N} \sum_{i=1}^{M} \phi_j(x)\phi_i(y) f(x_j, y_i)$$

where only 4 values of the double summation above are positive.

So a piecewise linear approximation of the function $a'(a,y)$ (here remember that we do not treat y as an argument of the function but just as a parameter) is given by a grid over assets $\{a_1, a_2, ..., a_M\}$ and by a set of values for the policty function on the grid points i.e. $\{a'(a_1,y), a'(a_2,y), ..., a'(a_M,y)\}$. At this point we can describe one algorithm for searching a solution in this class as follows

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

2. Guess an initial vectors of decision rules for a'' on all grid points and call the resulting piecewise linear approximation $\hat{a}_0''(a,y)$.You will need to specify as many vectors as there are values for the Markov chain.

3. 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

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

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 specifying an efficient approximation routine.

(a) If the solution of the nonlinear equation in (2) $a^* \geq -\bar{a}$, then set $a'_0(a_i, y_j) = a^*$,otherwise set $a^* = -\bar{a}$ and go to on the next grid point.

4. At the end of this you will obtain new vectors for the value of the decision rule on the grid points which define a new piecewise linear approximation $\hat{a}''_1(a, y)$ Check convergence by comparing $a'_1(a_i, y_j) - a'_0(a_i, y_j)$ through some pre-specified norm. For example, declare convergence at iteration n when

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

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

Piecewise linear interpolation is fast, and obviously preserves positivity, monotonicity and concavity. However, the optimal policy obtained is not differentiable everywhere.Piecewise linear approximation with policy function iteration and exogenous grid

0.2.3 Piecewise linear approximation with policy function iteration and endogenous grid

One problem with our previous method is that at each step it has to use a non linear equation solver which takes time and can easily crash. If the problem is simple enough this can be avoided using the so called endogenous grid method (see a paper by Barillas and Villaverde, 2006) . Here is how the method work:

1. Construct a grid on the asset space $\{a_1, a_2, ..., a_M\}$ with $a_1 = -\bar{a}$. This will not change throughout the iteration so we called it the fixed grid.

2. Guess an initial vectors of decision rules for a'' on all grid points and call the resulting piecewise linear approximation $\hat{a}''_0(a, y)$.You will need to specify as many vectors as there are values for the Markov chain.

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

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

Notice that in the previous algorith you take as given the value of assets today, the shock today and future decision rules and solve for assets tomorrow. In this method you take as given shocks today, assets tomorrow, future decision rules and solve for assets today. The advantage is that the equation above, because assets tomorrow are known, can be solved analytically as

$$a^* = \frac{u^{-1}\left(\beta R \sum_{y' \in Y} \pi(y'|y_j) u'(Ra_i + y' - \hat{a}''_0(a_i, y'))\right) + a_i - y_i}{R}$$

so there is no need of using a non linear equation solver.

4. At the end of this step you will obtain a vector of current assets a_0 as a function of current shocks and future assets a_0' . In order to update your decision rules, which were defined on the fixed grid $\{a_1, a_2, ..., a_M\}$ you need to perform an extra step. In particular use the vector of current assets as a new grid (this is the endogenous grid and it will vary at each step) and the vector of future assets (which is the fixed grid) as the value of the functions on the endogenous grid. Since in general the endogenous grid will be different from the fixed grid you will need to ise linear interpolation (as defined above) to solve for the updated decison rules on the fixed grid. (This step was not needed in the previous method). Once you do so you obtain new vectors for decision rule which define a new piecewise linear approximation $\hat{a}_1''(a, y)$ on the fixed grid. Then check convergence as above.

0.2.4 Chebyshev Approximation with policy function iteration

Chebyshev 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 $[-1, 1]$.

Many textbooks in numerical analysis show that the Chebyshev polynomials have a useful orthogonality property, meaning that they can approximate arbitrarily well any continous function. In practice here we want to find a way to determine the κ_p so that if we know the value of a function on a bunch of points we can then characterize function on an entire interval. 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 minimize the square of the distance between the true function and the approximating polynomial on the approximating nodes, that is

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

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 to continous functions. 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 (this is something you should always check when choosing an order of approximation), 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![3]

The Algorithm

1. Compute the M Chebyshev 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 zeros of the Chebyshev polynomial of order M

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

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

   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 $a_k \simeq -\bar{a}$.

2. Guess an initial vector of decision rules for a'' on the nodes of the grid, 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

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

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

[3] 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).

(a) Choose an order $N < M$ for the Chebyshev polynomials. Compute the Chebyshev 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 points as follows:

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

(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.

0.3 Non Iterative methods

Non iterative methods (sometimes called minimum weighted residuals method or projection approach) use the same types of approximation we discuss above (for example discrete, piecewise-linear or Chebyshev) but they differ in the way the optimal function is found. Let $a'(a, y, \theta)$ be the optimal function we are looking for, which is characterized by the vector of parameters θ.Instead of iterating on the policy function (or the value function) this method picks a particular loss function (an example of a loss function would be

$$\sum_{j=1}^{N} \sum_{i=1}^{M} \left(u' (Ra_i + y_j - a'(a_i, y_j, \theta)) - \beta R \sum_{y' \in Y} \pi (y'|y_j) u' (Ra'(a_i, y_j, \theta) + y' - a'(a'(a_i, y_i, \theta), y', \theta)) \right)^2$$

together with grids for a and for y of N and M points, and then minimizes it woth respect to the vector of parameters θ. These methods are releatively efficient when the candidate solution is a function of a relatively small number of parameters (i.e. when θ has low dimensionality) but it gets harder to handle when this is not the case. So for example this method can be used when one approximates the optimal function using Chebyshev (whcih requires a relatively small numbers of parameters) but it is not recommended if you use discretization. Also keep in mind that when you solve a non minear minimization problem you always have to check that you are not stuck in a local minimum. See Judd, section 11.3 for a general description of these methods.

0.4 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 Chebyshev approximation method, how do we determine that increasing the order of the polynomial would not lead to any significant improvement in accuracy?

0.4.1 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

$$u'(c_t) = \beta R E_t [u'(c_{t+1})]$$

implies that the residual

$$\varepsilon_{t+1} = u'(c_t) - \beta R u'(c_{t+1})$$

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.

0.4.2 Euler Equation Error Analysis

The numerically approximated Euler equation is

$$u'(c_t) \simeq \beta R E_t \left[u'(c_{t+1})\right].$$

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

$$u'(c_t(1-\varepsilon_t)) = \beta R E_t \left[u'(c_{t+1})\right]$$

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

$$\varepsilon_t = 1 - \frac{g\left(\beta R E_t \left[u'(c_{t+1})\right]\right)}{c_t}.$$

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. 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.

0.5 Notes

See Tauchen (1986) for a detailed description of the discretization of a continuous vector-autoregressive process. See the book by Judd on "Numerical Methods in Economics", especially chapters 6 on approximation and chapter 12 on dynamic programming. The book by Marimon and Scott "Computational Methods for the Study of Dynamic Economies" is another very useful reference. A recent book by Adda and Cooper on practical dynamic programming explains in detail various solution methods and proposes many nice examples. Aruoba, Fernandez-Villaverde and Rubio-Ramirez (2006) describe in detail the properties and the accuracy of various solution methods applied to the neoclassical growth model.