Dynamic Economics
Quantitative Methods and Applications to Macro and Micro

Jérôme Adda and Nicola Pavoni

MACT1 2003-2004. I

Overview

- Dynamic Programming Theory
  - Contraction mapping theorem.
  - Euler equation
- Numerical Methods
- Econometric Methods
- Applications

Numerical Methods

Examples: Cake Eating

- Deterministic Cake eating:

$$V(K) = \max_c u(c) + \beta V(K - c)$$

with
- K: size of cake. $K \geq 0$
- c: amount of cake consumed. $c \geq 0$

- Stochastic Cake eating:

$$V(K, y) = \max_c u(c) + \beta E_{y'} V(K', y')$$

$$K' = K - c + y$$

- Discrete Cake eating:

$$V(K, \varepsilon) = \max[u(K, \varepsilon), \beta E_{\varepsilon'} V(\rho K, \varepsilon')] \qquad \rho \in [0, 1]$$

How do we Solve These Models?

- Not necessarily a closed form solution for $V(.)$.
- Numerical approximations.

Solution Methods

- Value function iterations. (Contraction Mapping Th.)
- Policy function iterations. (Contraction Mapping Th.)
- Projection methods. (Euler equation)

Value Function Iterations

Value Function Iterations

$$V_n(S) = \max_{\text{action}} u(\text{action}, S) + \beta E V_{n-1}(S')$$

$$V_n(.) = T V_{n-1}(.)$$

- Take advantage of the Contraction Mapping Theorem. If T is the contraction operator, we use the fact that

$$d(V_n, V_{n-1}) \leq \beta d(V_{n-1}, V_{n-2})$$

$$V_n(.) = T^n V_0(.)$$

- This guarantee that:
  1. successive iterations will converge to the (unique) fixed point.
  2. starting guess for V_0 can be arbitrary.
- Successive iterations:
  - Start with a given $V_0(.)$. Usually $V_0(.) = 0$.
  - Compute $V_1(.) = T V_0(.)$
  - Iterate $V_n(.) = T V_{n-1}(.)$
  - Stop when $d(V_n, V_{n-1}) < \varepsilon$.

Value Function Iterations: Deterministic Cake Eating

- Model:

$$V(K) = \max_c u(c) + \beta V(K - c)$$

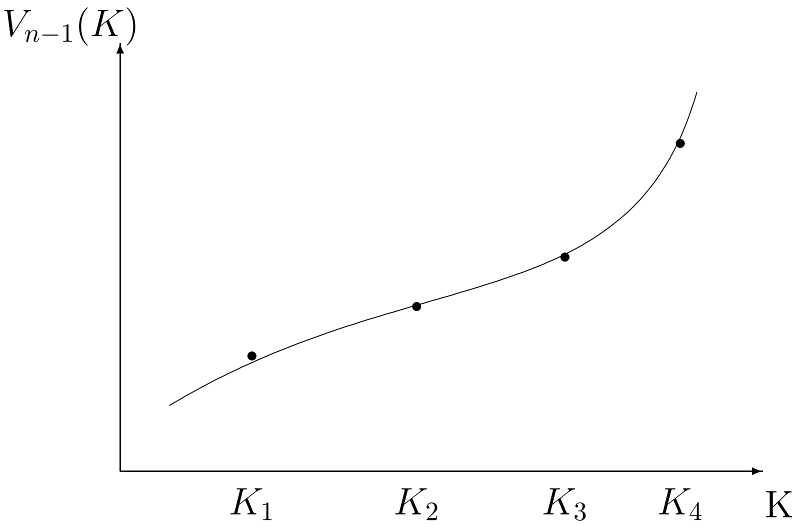
- Can be rewritten as:

$$V(K) = \max_{K'} u(K - K') + \beta V(K')$$

- The iterations will be on

$$V_n(K) = \max_{K'} u(K - K') + \beta V_{n-1}(K')$$

- example: take $u(c) = \ln(c)$.
  - we need a grid for the cake size. $\{K_0, \ldots, K_N\}$
  - we need to store the successive values of V_n: N x 1 vector.
  - search on the $\{K\}$ grid, the value K' which gives the highest utility flow.



Computer Code for Deterministic Cake Eating Problem

```
clear                        % clear workspace memory
dimIter=30;                  % number of iterations
beta=0.9;                    % discount factor

K=0:0.005:1;                 % grid over cake size, from 0 to 1
dimK=length(K);              % numbers of rows (size of grid)

V=zeros(dimK,dimIter);       % initialize matrix for value function


for iter=1:dimIter           % start iteration loop

   aux=zeros(dimK,dimK)+NaN;
   for ik=1:dimK             % loop on all sizes for cake
       for ik2=1:(ik-1)      % loop on all future sizes of cake
           aux(ik,ik2)=log(K(ik)-K(ik2))+beta*V(ik2,iter);
       end
   end
   V(:,iter+1)=max(aux')';    % computes the maximum over all future sizes
end

plot(K,V);            % plots all the successive values against size of cake
```

Discrete Cake Eating Model

- Model:

$$V(K,\varepsilon) = \max[u(K,\varepsilon), \beta E_{\varepsilon'} V(\rho K, \varepsilon')] \qquad \rho \in [0,1]$$

- Grid for the size of the cake: $\{K_0, \rho K_0, \rho^2 K_0, \ldots, \rho^N K_0\}$
- Markov process for the taste shock: $\varepsilon \in \{\underline{\varepsilon}, \bar{\varepsilon}\}$

$$\pi = \begin{bmatrix} \pi_{LL} & \pi_{LH} \\ \pi_{HL} & \pi_{HH} \end{bmatrix}$$

- We construct V as a Nx2 matrix, containing the value function.
- Let i_k denote an index for the cake size, $i_k \in \{1,\ldots,N\}$, and i_ε an index for the taste shock.
  - for a given i_k and i_ε, we compute:
    * the value of eating the cake now: $u(K[i_k], \varepsilon[i_\varepsilon]))$
    * the value of waiting: $\sum_{i=1}^{2} \pi_{i_\varepsilon,i} V(\rho K[i_K], \varepsilon[i])$
  - we then compute the max of these two values.

Code for Discrete Cake Eating Problem

```
%%%%%%%%%%%%%%%  Initialisation of parameters   %%%%%%%%%%%%%%%%%%%%%%%%%%
 itermax=60;                % number of iterations
 dimK=100;                  % size of grid for cake size
 dimEps=2;                  % size of grid for taste shocks
 K0=2;                      % initial cake size
 ro=0.95;                   % shrink factor
 beta=0.95;                 % discount factor

 K=0:1:(dimK-1);
 K=K0*ro.^K';               % Grid for cake size 1 ro ro^2...

 eps=[.8,1.2];              % taste shocks
 pi=[.6 .4;.2 .8];          % transition matrix for taste shocks

 V=zeros(dimK,dimEps);      % Stores the value function.
                            % Rows are cake size and columns are shocks
 auxV=zeros(dimK,dimEps);
%%%%%%%%%%%%%%%  End Initialisation of parameters   %%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%  Start of Iterations                %%%%%%%%%%%%%%%%%%%%%%%%%%
 for iter=1:itermax;                   % loop for iterations
    for ik=1:dimK-1;                   % loop over size of cake
        for ieps=1:dimEps;             % loop over taste shocks
        Vnow=sqrt(K(ik))*eps(ieps);    % utility of eating the cake now
        Vwait=pi(ieps,1)*V(ik+1,1)+pi(ieps,2)*V(ik+1,2);
        auxV(ik,ieps)=max(Vnow,beta*Vwait);
    end                                % end loop over taste shock
    end                                % end loop over size of cake     V=auxV;
 end

 plot(K,V)                             % graph the value function
                                       % as a function of cake size
```

Continuous Cake Eating Problem

- Program of the agent:

$$V(W,y) = \max_{0 \le c \le W+y} u(c) + \beta E_{y'|y} V(W',y') \qquad \text{for all } (W,y)$$

$$\text{with} \quad W' = R(W - c + y) \text{ and } \quad y \text{ is iid} \tag{1}$$

- We can rewrite this Bellman equation by defining:

$$X = W + y$$

  the total amount of cake available at the beginning of the period.

$$V(X) = \max_{0 \le c \le X} u(c) + \beta E_{y'} V(X') \qquad \text{for all } X$$

$$\text{with} \quad X' = R(X - c) + y' \tag{2}$$

- The operator is defined as:

$$T(V(X)) = \max_{c \in [0,X]} u(c) + \beta E_{y'} V(X'). \tag{3}$$

Value Function Iterations

- First, we need to discretize the state variable X: $\{X^1, \ldots, X^{nS}\}$
- Second, we discretize the choice variable c: $\{c^1, \ldots, c^{n_c}\}$
- Suppose we know $V_{n-1}(X^i)$, $i \in \{1, \ldots, n_s\}$.
- For any values on the grid X^i, and c^j, we evaluate:

$$v_{ij} = u(c^j) + \beta \sum_{k=1}^{K} \pi_k V_{n-1}(R(X^i - c^j) + y^k)$$

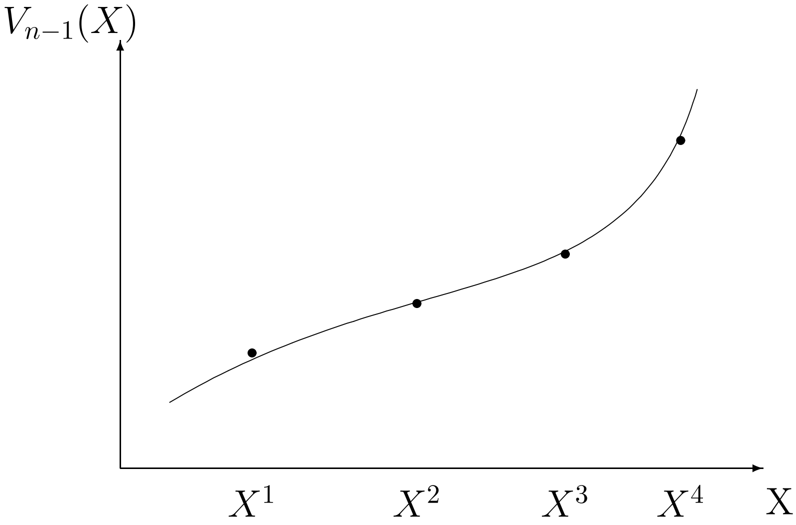
- then

$$V_n(X^i) = \max_j v_{ij}$$

- We stop when $|V_n(X^i) - V_{n-1}(X^i)| < \varepsilon, \forall X^i$

Approximating Value in Next Period

$$v_{ij} = u(c^j) + \beta \sum_{k=1}^{K} \pi_k V_{n-1}(R(X^i - c^j) + y^k)$$



to calculate $V_{n-1}(R(X^i - c^j) + y^k)$, they are several options:

- we find i' such that $X^{i'}$ is closest to $R(X^i - c^j) + y^k$

$$V_{n-1}(R(X^i - c^j) + y^k) \simeq V_{n-1}(X^{i'})$$

- find i' such that $X^{i'} < R(X^i - c^j) + y^k < X^{i'+1}$, then perform linear interpolation:

$$V_{n-1}(R(X^i - c^j) + y^k) \simeq \lambda V_{n-1}(X^{i'}) + (1-\lambda) V_{n-1}(X^{i'+1})$$

Policy Function Iterations

Policy Function Iteration

- Improvement over value function iterations.
- faster method for small problems.
- Implementation:
  - guess $c_0(X)$.
  - evaluate:

$$V_0(X) = u(c_0(X)) + \beta \sum_{i=L,H} \pi_i V_0(R(X - c_0(X)) + y_i)$$

    this requires solving a system of linear equations.
  - policy improvement step:

$$c_1(X) = \underset{c}{\text{argmax}}[u(c) + \beta \sum_{i=L,H} \pi_i V_0(R(X - c) + y_i)]$$

  - continue until convergence.

Projection Methods

Projection Methods

- Example: Continuous cake eating: Euler equation:

$$\begin{cases} u'(c_t) = \beta E_t u'(c_{t+1}) & \text{if} \quad c_t < X_t \\ c_t = X_t & \text{if corner solution} \end{cases}$$

- This can be rewritten as:

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

$$c_{t+1} = X_t - c(X_t) + y_{t+1}$$

- The solution to this equation is a function: $c(X_t)$

$$u'(c(X_t)) - \max\left[X_t, \beta E_{y'} u'\left(X_t - c(X_t) + y'\right)\right] = 0$$

$$F(c(X_t)) = 0$$

- Goal: Find a function $\hat{c}(X)$ which satisfies the above equation.
  Find the zero of the functional equation.

Approximating the Policy Function

- Define $\hat{c}(X,\Psi)$ be an approximation to the real $c(X)$.

$$\hat{c}(X,\Psi) = \sum_{i=1}^{n} \psi_i p_i(X)$$

where $\{p_i(X)\}$ is a base of the space of continuous functions.
Examples:

  - $\{1, X, X^2, \ldots\}$
  - Chebyshev polynomials:

$$\begin{cases} p_i(X) = \cos(i \ \arccos(X)) & X \in [0,1], \ i = 0,1,2,\ldots \\ p_i(X) = 2Xp_{i-1}(X) - p_{i-2}(X) & i \geq 2, \ \text{ with } p_0(0) = 1, \ \ p_1(X) = X \end{cases}$$

  - Legendre or Hermite polynomials.

- For instance, the policy function can be approximated by:

$$\hat{c}(X,\Psi) = \psi_0 + \psi_1 X + \psi_2 X^2$$

$$\hat{c}(X,\Psi) = \psi_0 + \psi_1 X + \psi_2 (2X^2 - 1) + ...$$

Defining a Metric

- We want to bring $F(\hat{c}(X, \psi)$ as "close as possible" to zero.
- How do we define "close to zero"?
- For any weighting function $g(x)$, the inner product of two integrable functions f_1 and f_2 on a space A is defined as:

$$\langle f_1, f_2 \rangle = \int_A f_1(x) f_2(x) g(x) dx \tag{4}$$

- Two functions f_1 and f_2 are said to be orthogonal, conditional on a weighting function $g(x)$, if

$$\langle f_1, f_2 \rangle = 0$$

  The weighting function indicates where the researcher wants the approximation to be good.

- In our problem, we want

$$\langle F(\hat{c}(X, \Psi)), f(X) \rangle = 0$$

  where $f(X)$ is a given function. The choice of the f function will give different projection methods.

Different Projection Methods

- Least square method:

$$\min_{\Psi} \langle F(\hat{c}(X,\Psi)), F(\hat{c}(X,\Psi)) \rangle$$

- Collocation method:

$$\min_{\Psi} \langle F(\hat{c}(X,\Psi)), \delta(X - X_i) \rangle \quad i = 1, \ldots, n$$

where $\delta(X - X_i)$ is the mass point function at point X_i:

$$\begin{array}{l} \delta(X) = 1 \;\text{ if }\; X = X_i \\ \delta(X) = 0 \quad \text{elsewhere} \end{array}$$

- Galerkin method:

$$\min_{\Psi} \langle F(\hat{c}(X,\Psi)), p_i(X) \rangle \quad i = 1, \ldots, n$$

where $p_i(X)$ is a base of the function space.

Collocation Methods

- We find Ψ by minimizing:

$$\langle F(\hat{c}(X,\Psi)), \delta(X - X_i)\rangle \qquad i = 1, \dots n$$

where $\delta()$ is the mass point function.

- The method requires that $F(\hat{c}(X,\Psi))$ is zero at some particular points X_i and not over the whole range $[\bar{X}_L, \bar{X}_H]$.

- The method is more efficient if these points are chosen to be the zeros of the basis elements $p_i(X)$, here $X_i = \cos(\pi/2i)$. (orthogonal collocation method).

- Ψ is the solution to a system of nonlinear equations:

$$F(\hat{c}(X_i,\Psi)) = 0 \qquad i = 1, \dots n$$

- Note:
  - This method is good at approximating policy functions which are relatively smooth.
  - Chebyshev polynomials tends to display oscillations at higher orders.

Computer Code for Projection Method

```
procedure c(x)                          * Here we define an approximation for
cc=psi_0+psi_1*x+psi_2*x*x                the consumption function based on
return(cc)                                a second order polynomial *
endprocedure



i_s=1
 do until i_s>n_s                         * Loop over all sizes of the total
                                                amount of cake *
  utoday=U'(c(X[i_s]))                  * marginal utility of consuming *
  ucorner=U'(X[i_s])                    * marginal utility if corner solution *
  i_y=1
  do until i_y>n_y                      * Loop over all possible realizations
                                          of the future endowment *
  nextX=R(X[i_s]-c(X[i_s]))+Y[i_y]      * next amount of cake *
  nextU=U'(C(nextX))                      * next marginal utility of consumption *
  EnextU=EnextU+nextU*Pi[i_y]           * here we compute the expected future
                                          marginal utility of consumption using
                                          the transition matrix Pi *
  i_y=i_y+1
  endo                                  * end of loop over endowment  *
F[i_s]=utoday-max(ucorner,beta*EnextU)
i_s=i_s+1
endo                                   * end of loop over size of cake *
```

Programming Languages

- C++, FORTRAN, PASCAL...
  - the real stuff. Very quick.
  - not very user friendly.
  - no graphic packages, no predefined commands.
- GAUSS, MATLAB
  - more user friendly.
  - matrix oriented.
  - graphic packages.
  - quick, except when doing loops.

Some Elements of Programming

- Structure of a program:
  - start with definition and initialisation of variables.
  - main code.
  - display results.
- A few tips:
  - create variables with meaningful names.
    (prefer 'beta' to 'x1').
  - break down complex calculations into smaller and understandable units.
  - create procedures (subroutines) which will do more complex calculations. For the main program, these procedures are just black boxes which transform some inputs into outputs:
    e.g.:
  - put comments into your program which state what the line is doing

Econometric Methods

Overview

- Dynamic Programming Theory
  - Contraction mapping theorem.
  - Euler equation
- Numerical Methods
- Econometric Methods
- Applications

Aim

- Estimate the "structural" parameters of a DP model.
  - parameters describing the utility function.
  - technology parameters.
  - discount factor.
- from observed data.
- Example:
  - Discrete cake eating problem:

$$V(K,\varepsilon) = \max[u(K,\varepsilon), \beta E_{\varepsilon'} V(\rho K, \varepsilon')] \qquad \rho \in [0,1]$$

  - Data on cake sizes and periods in which they are eaten:

Period	Cake Size	obs 1	obs 2	...	obs N
1	1	0	0	...	0
2	0.8	0	1	...	0
3	0.64	1	1	...	0
4	0.51	1	1	...	0
5	0.41	1	1	...	1

  - Infer, β, utility function and distribution of taste shocks.

Estimation Methods

- maximum likelihood.
- method of moments.
- simulated maximum likelihood.
- simulated method of moments.
- simulated non linear least squares.
- indirect inference.

A Simple Example: Coin Flipping

- Probability of head/tail $\{P_1, P_2\} \in [0,1]\text{x}[0,1]$.
- N draws: H, H, T, H, T
- Random variable X_t.
- Draws $\{x_1, x_2, \ldots, x_N\}$
- Denote N_i the number of observation that falls into category $i = 1, 2$.

Model: $P = P(\theta)$

Example 1:

$$\begin{cases} P_1 = \theta \\ \\ P_2 = 1 - \theta \end{cases} \quad \theta \in [0,1]$$

Example 2:

$$\begin{cases} P_1 = \Phi(\theta) \\ \\ P_2 = 1 - \Phi(\theta) \end{cases} \quad \theta \in]-\infty, +\infty[$$

Note: This is in fact a probit model:

$$X_t^* = u_t$$

$$\begin{array}{lll} X_t = 1 & \text{if} & X_t^* < \theta \\ X_t = 2 & \text{if} & X_t^* \geq \theta \end{array}$$

Coin Flipping: Maximum Likelihood

- Likelihood function: (simple as i.i.d. draws)

$$\begin{aligned} \pounds &= P(X_1 = x_1, X_2 = x_2, \ldots, X_N = x_N) \\ &= P_1(\theta)^{N_1}(1 - P_1(\theta))^{N_2} \end{aligned}$$

  - with sequence: H, H, T, H, T

$$\pounds = P_1 * P_1 * P_2 * P_1 * P_2 = P_1^3 * P_2^2 = P_1^3(1 - P_1)^2$$

- Maximum Likelihood Estimator: $P_i(\theta^*) = \frac{N_i}{N}$
  - example 1: $\theta^* = N_1/N$
  - example 2: $\theta_i^* = \Phi^{-1}(\frac{N_i}{N})$

Coin Flipping: Method of Moments

- Moment from data: μ. (mean, variance, covariance...)
- Moment from model: $\mu(\theta)$.
- Parameter estimate is the solution of:

$$\min_{\theta}(\mu(\theta) - \mu)^2$$

- With coin flipping:
  - $\mu = N_1/N$: observed fraction of heads.
  - $\mu(\theta) = P_1(\theta)$, predicted fraction of heads by model.
  - which trivially leads to $P_1(\theta^*) = N_1/N$

More generally:

$$\min_{\theta}(\mu(\theta) - \mu)'\Omega^{-1}(\mu(\theta) - \mu)$$

Ω is a weighting matrix.

Coin Flipping: Simulated Methods

Simulating the model (example 2):

- Guess θ.
- Draw S shocks $\{u_s\}$ from a standard normal density.
- Create $\{\tilde{x}_s\}$ such that

$$\begin{array}{ll} \tilde{x}_s = \text{head} & \text{if} \quad u_s < \theta \\ \tilde{x}_s = \text{tail} & \text{if} \quad u_s \geq \theta \end{array}$$

- Example: $\theta = 0$

Draw	u_s	Outcome
1	0.518	T
2	1.611	T
3	-0.89	H
4	1.223	T
⋮	⋮	⋮
S	0.393	T

- we get $S_1(\theta)$ heads and $S_2(\theta)$ tails. $(S_1(\theta) + S_2(\theta) = S)$

Coin Flipping: Simulated Maximum Likelihood

- Compute the frequency of each outcome using the simulated data.

$$P_i^S(\theta) = \frac{1}{S} \sum_{s=1}^{S} I(X_s = i) = \frac{S_1(\theta)}{S}$$

- θ_S^* solution to:

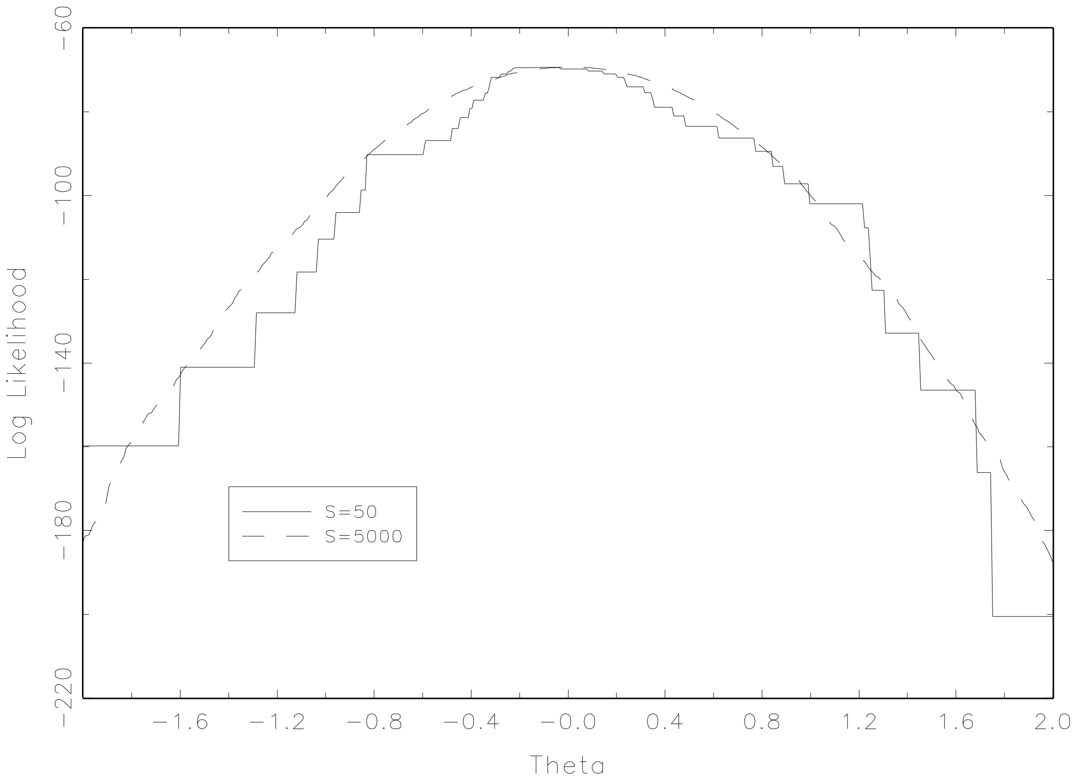
$$\max_\theta \prod_i P_i^S(\theta)^{N_i} = \max_\theta \left(\frac{S_1(\theta)}{S} \right)^{N_1} \cdot \left(\frac{S_2(\theta)}{S} \right)^{N_2}$$

- Optimal parameter:

$$\frac{S_1(\theta^*)}{S} = \frac{N_1}{N}$$

Likelihood Function

Figure 1: Log Likelihood, True $\theta_0 = 0$



Coin Flipping: Simulated Method of Moments

- Compute the vector of moments from the observed data: μ.
- Compute the vector of moments from the simulated data: $\mu^S(\theta)$.
- The optimal parameter θ_S^* is the solution of:

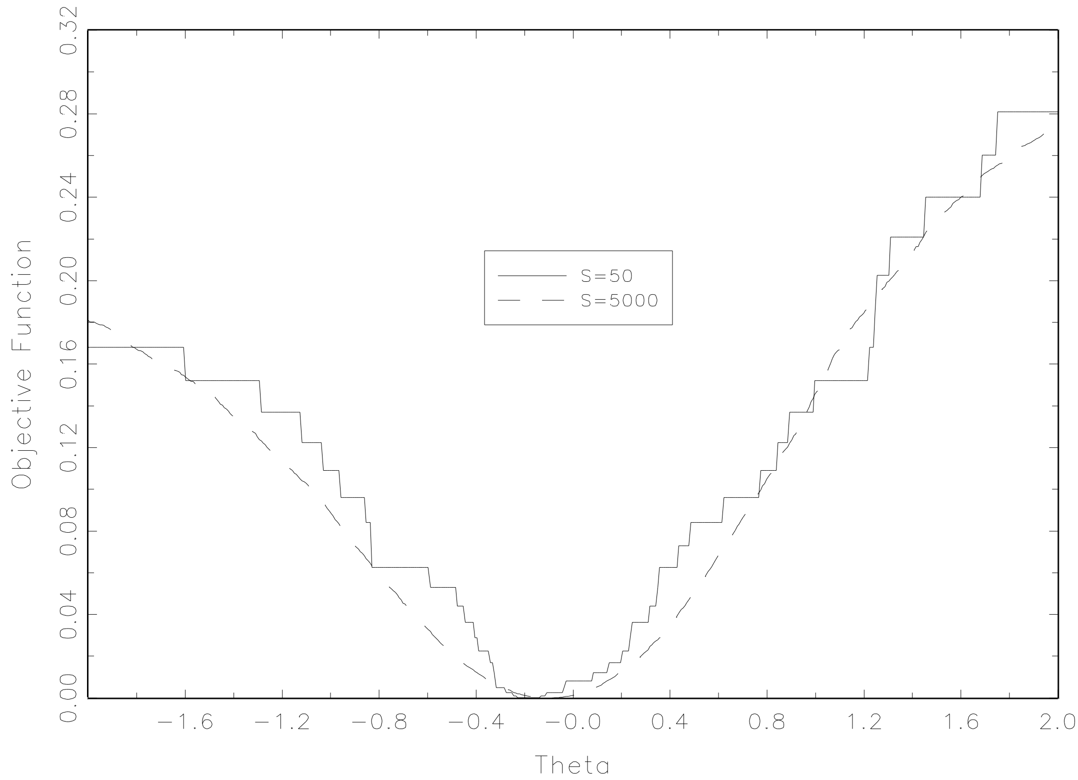
$$\min_{\theta}(\mu^S(\theta) - \mu)' W^{-1} (\mu^S(\theta) - \mu)$$

- In our coin flipping example:
  - observed moment: fraction of heads: N_1/N
  - simulated moment: $S_1(\theta)/S$
  - optimal parameter:

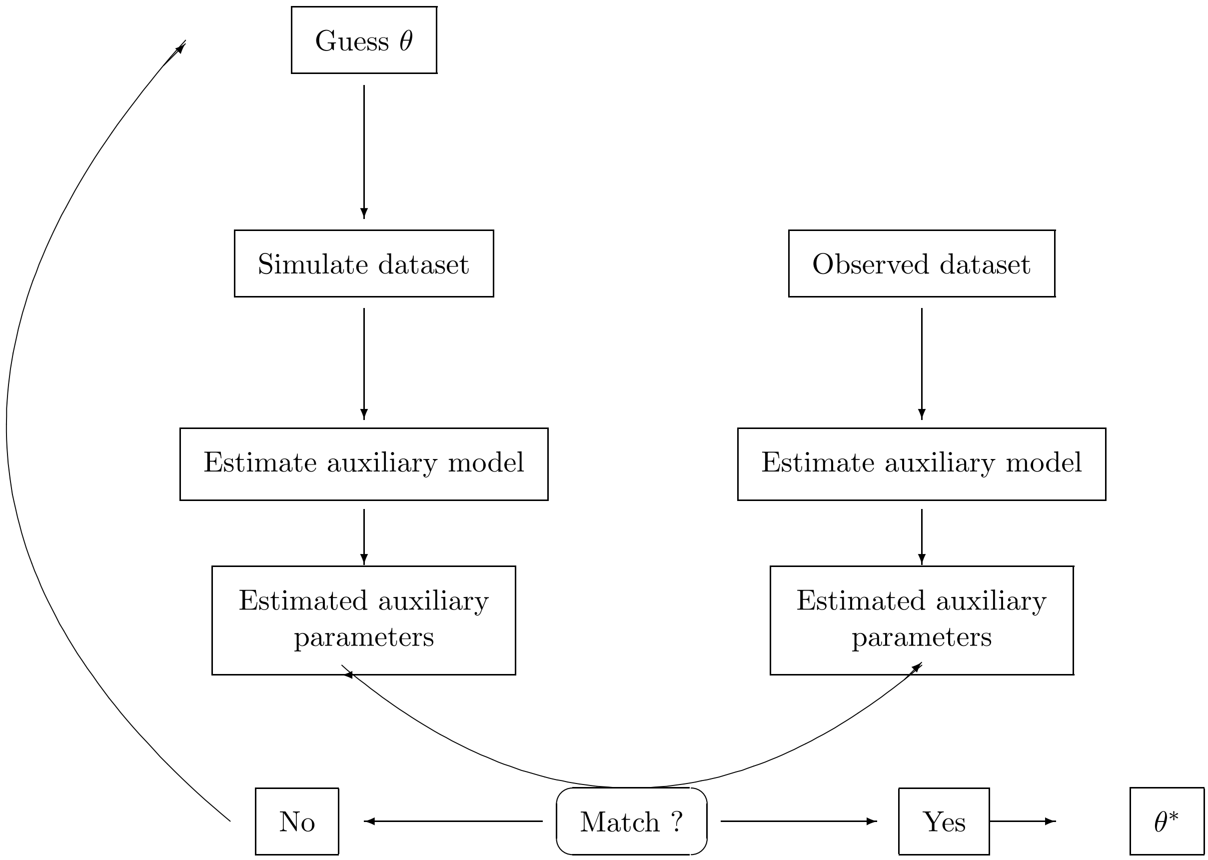
$$\frac{S_1(\theta^*)}{S} = \frac{N_1}{N}$$

Objective Function

Figure 2: Objective Function



Indirect Inference



Coin Flipping: Indirect Inference

- Auxiliary model. $\tilde{M}(\beta)$
- Auxiliary parameters β
- Estimate the auxiliary model on observed data: β_N
- Estimate the auxiliary model on simulated data: $\beta_S(\theta)$
- The optimal parameter estimates are:

$$\theta_S^* = \underset{\theta}{\operatorname{argmin}}(\beta_S(\theta) - \beta_T)^2$$

Example:

Auxiliary model (logit): $$P(X_t = 1) = \frac{\exp(\beta)}{1 + \exp(\beta)}$$

Log-Likelihood of auxiliary model for observed data:

$$l = N_1 \ln \frac{\exp(\beta)}{1 + \exp(\beta)} + N_2 \ln \frac{1}{1 + \exp(\beta)} = N_1\beta - N \ln(1 + \exp(\beta))$$

ML estimator for the auxiliary model:

$$\beta^* = \ln \frac{N_1}{N_2}$$

$$\theta_S^* = \underset{\theta}{\operatorname{argmin}}(\ln \frac{S_1(\theta)}{S_2} - \ln \frac{N_1}{N_2})^2$$

Cake Eating Problem: Maximum Likelihood

- Bellman Equation:

$$V(K,\varepsilon) = \max[u(K,\varepsilon), \beta E_{\varepsilon'/\varepsilon} V(\rho K, \varepsilon')]$$

- Define the threshold $\varepsilon^*(K,\theta)$ such as:

$$u(K,\varepsilon^*(K,\theta)) = \beta EV(\rho K, \varepsilon')$$

the agent is indifferent between eating and waiting.

- The probability of waiting is:

$$P(wait|K) = P(\varepsilon < \varepsilon^*(K,\theta)) = F(\varepsilon^*(K,\theta))$$

- Likelihood of observing a cake eaten after t_i periods for agent i:

$$l_i(t_i,\theta) = P(\varepsilon_{i1} < \varepsilon^*(K_{i1}), \ldots, \varepsilon_{i,t_i-1} < \varepsilon^*(K_{i,t_i-1}), \varepsilon_{it_i} > \varepsilon^*(K_{it_i}))$$

If the ε are **iid**, then:

$$\begin{aligned} l_i(t_i,\theta) &= \prod_{l=1}^{t_i-1} P(\varepsilon_{il} < \varepsilon^*(K_{il})) \ . \ P(\varepsilon_{it_i} > \varepsilon^*(K_{it_i})) \\ &= \prod_{l=1}^{t_i-1} F(\varepsilon^*(K_{il},\theta)) \ . \ (1 - F(\varepsilon^*(K_{it_i},\theta))) \end{aligned}$$

- Likelihood of entire sample:

$$L(\theta) = \prod_{i=1}^{N} l_i(t_i,\theta)$$

Properties of ML

Asymptotically normal and unbiased estimates:

$$\sqrt{N}(\hat{\theta}_N - \theta_0) \xrightarrow{L} N(0, I^{-1})$$

$$I = -\frac{1}{N}\sum_{i=1}^{N}\frac{\partial^2 \log l(t_i,\theta)}{\partial\theta\partial\theta'}$$

Cake Eating Problem: Serially Correlated Shocks

- If ε is not iid, then the likelihood is complicated

  $$l_i(t_i,\theta) = P(\varepsilon_{i1} < \varepsilon^*(K_{i1}), \ldots, \varepsilon_{i,t_i-1} < \varepsilon^*(K_{i,t_i-1}), \varepsilon_{it_i} > \varepsilon^*(K_{it_i}))$$

- Example: $t_i = 2$

  $$\begin{aligned} l_i(2) &= P(\varepsilon_1 < \varepsilon^*(K_1), \varepsilon_2 > \varepsilon^*(K_2)) \\ &= P\left(\varepsilon_2 > \varepsilon^*(K_2) | \varepsilon_1 < \varepsilon^*(K_1)\right) \; P\left(\varepsilon_1 < \varepsilon^*(K_1)\right) \\ &= \frac{1}{\sqrt{2\pi\sigma}} \int_{\varepsilon_2^*}^{+\infty} \int_{-\infty}^{\varepsilon_1^*} \exp(-\frac{1}{2\sigma^2}(u-\rho v)^2) du dv \;\; \Phi\left(\frac{\varepsilon_1^*(K_1)}{\sigma/\sqrt{1-\rho^2}}\right) \end{aligned}$$

  $\Longrightarrow$ for any agent i we have to solve t_i integrals: INTRACTABLE.

- use simulation based methods.

Simulation of Cake Eating Model

- Given vector θ: compute $\varepsilon^*(K, \theta)$.
- Fix S the size of the simulated dataset.
- For each agent s,
  - draw T serially correlated taste shocks.
  - Compute the date of consumption, t_s, as first taste shock exceeding the threshold.
- This gives a set of S stopping times.
- Simplified example:

```
S=1000;
T=100;
ro=0.5;
sig=0.1;
eps=zeros(T,S);
dateconso=zeros(S,1);
for s=1:S
t=1;
do while eps(t,s)<threshold;
eps(t+1,s)=ro*eps(t,s)+rand*sig;
t=t+1;
dateconso(s)=t;
end
end
```

Simulated Method of Moments

- From observed data: construct a moment $\mu(t_i)$:
  - $\mu(t_i) = t_i/N$, mean.
  - $\mu(t_i) = (t_i - \bar{t})^2/N$, variance.
- From simulated data, construct the same moment $\mu(t_i(\theta))$.
- The estimator for the SMM is defined as:

$$\hat{\theta}_{S,N}(\Omega) = \arg\min_{\theta} \left[\sum_{i=1}^{N} \left(\mu(t_i) - \frac{1}{S} \sum_{s=1}^{S} \mu(t_i(\theta)) \right) \right]' \Omega_N^{-1} \left[\sum_{i=1}^{N} \left(\mu(t_i) - \frac{1}{S} \sum_{s=1}^{S} \mu(t_i^s(\theta)) \right) \right]$$

Properties

- When the number of simulation S is fixed and $N \longrightarrow \infty$,
  - $\hat{\theta}_{SN}(\Omega)$ is consistent.
  - $\sqrt{N}(\hat{\theta}_{SN} - \theta_0) \longrightarrow N(0, Q_S(\Omega))$

  where

  $$Q_S(\Omega) = (1+\frac{1}{S})\left[E_0\frac{\partial\mu'}{\partial\theta}\Omega_N^{-1}\frac{\partial\mu}{\partial\theta'}\right]^{-1} E_0\frac{\partial\mu'}{\partial\theta}\Omega_N^{-1}\Sigma(\theta_0)\Omega_N^{-1}\frac{\partial\mu}{\partial\theta'}\left[E_0\frac{\partial\mu'}{\partial\theta}\Omega_N^{-1}\frac{\partial\mu}{\partial\theta'}\right]^{-1}$$

  where $\Sigma(\theta_0)$ is the covariance matrix of $1/\sqrt{N}(\frac{1}{N}\sum_{i=1}^{N}(\mu(t_i) - E_0\mu(t_i^s(\theta)))$.

- The optimal SMM is obtained when $\hat{\Omega}_N = \hat{\Sigma}_N$. In this case,

  $$Q_S(\Omega^*) = (1+\frac{1}{S})\left[E_0\frac{\partial\mu'}{\partial\theta}\Omega_N^{-1}\frac{\partial\mu}{\partial\theta'}\right]^{-1}$$

Indirect Inference

Use auxiliary model (misspecified) such that auxiliary parameters on observed and simulated data are similar.

- Auxiliary model: likelihood $\tilde{\phi}(t_i, \beta)$.
- Auxiliary parameters from *observed* data:

$$\hat{\beta}_N = \arg\max_{\beta} \prod_{i=1}^{N} \tilde{\phi}(t_i, \beta)$$

- Auxiliary parameters from *simulated* data:

$$\hat{\beta}_{sN}(\theta) = \arg\max_{\beta} \prod_{i=1}^{N} \tilde{\phi}(t_i^s(\theta), \beta)$$

- Average value of auxiliary parameters from *simulated data* :

$$\hat{\beta}_{SN} = \frac{1}{S} \sum_{s=1}^{S} \hat{\beta}_{sN}(\theta)$$

The indirect inference estimator $\hat{\theta}_{SN}$ is the solution to:

$$\hat{\theta}_{SN} = \arg\min_{\theta} [\hat{\beta}_N - \hat{\beta}_{SN}(\theta)]' \Omega_N [\hat{\beta}_N - \hat{\beta}_{SN}(\theta)]$$

where Ω_N is a positive definite weight matrix which converges to a deterministic positive definite matrix Ω.

Properties: For a fixed number of simulations S, when N goes to infinity the indirect inference estimator is consistent and normally distributed.

$$\sqrt{N}(\hat{\theta}_{SN} - \theta_0) \longrightarrow N(0, Q_S(\Omega))$$

Denote $\psi_N(\theta,\beta) = \sum_{i=1}^N \log \tilde{\phi}(t_i^s(\theta),\beta)$.

$$Q_S(\Omega^*) = (1+\frac{1}{S})\left(\frac{\partial^2\psi_\infty(\theta_0,b(\theta_0))}{\partial\theta\partial\beta'}(I_0-K_0)^{-1}\frac{\partial^2\psi_\infty(\theta_0,b(\theta_0))}{\partial\beta\partial\theta'}\right)^{-1}$$

$$(\widehat{I_0-K_0}) = \frac{N}{S}\sum_{s=1}^S (W_s-\bar{W})(W_s-\bar{W})'$$

with

$$W_s = \frac{\partial\psi_N(\hat{\theta},\hat{\beta})}{\partial\beta}$$

$$\bar{W} = \frac{1}{S}\sum_{s=1}^S W_s$$

Indirect Inference and Cakes

Auxiliary model: exponential duration model:

$$P(t_i = t) = \beta \exp(-\beta t)$$

Log-Likelihood of observed sample:

$$\ln L = \sum_{i=1}^{N} \ln(\beta \exp(-\beta t_i))$$

which has a maximum at:

$$\hat{\beta}_N = 1/N \sum_{i=1}^{N} t_i$$

From simulated data:

$$\hat{\beta}_{sN}(\theta) = 1/N \sum_{i=1}^{N} t_i^s(\theta)$$

so that

$$\hat{\beta}_{SN} = \frac{1}{NS} \sum_{s=1}^{S} \sum_{i=1}^{N} t_i^s(\theta)$$

$\hat{\theta}_{SN}$ is the solution of:

$$\min_{\theta} \left(\frac{1}{N} \sum_{i=1}^{N} t_i - \frac{1}{NS} \sum_{s=1}^{S} \sum_{i=1}^{N} t_i^s(\theta)\right)^2$$

Simulated Non Linear Least Squares

A "natural" way to proceed would be to look at a criterion such that:

$$\min \frac{1}{N}\sum_{i=1}^{N}(t_i - \bar{t}_i^S(\theta))^2$$

where $\bar{t}_i^S = 1/S \sum_{s=1}^{S} t_i^s(\theta)$

Problem: Not a consistent estimator of θ_0.

Laffont et al. (1995) proposes a criterion such that:

$$\min_{\theta} \frac{1}{N}\sum_{i=1}^{N}\left[(t_i - \bar{t}_i^S(\theta))^2 - \frac{1}{S(S-1)}\sum_{s=1}^{S}(t_i^s(\theta) - \bar{t}_i^S(\theta))^2\right]$$

Asymptotic Properties: For any fixed number of simulation S,

- $\hat{\theta}_{SN}$ is consistent.
- $\sqrt{N}(\hat{\theta}_{SN} - \theta_0) \xrightarrow{d} N(0, \Sigma_{S,N})$

A consistent estimate of the covariance matrix $\Sigma_{S,N}$ can be obtained by computing:

$$\hat{\Sigma}_{S,N} = \hat{A}_{S,N}^{-1}\hat{B}_{S,N}\hat{A}_{S,N}^{-1}$$

where $\hat{A}_{S,N}$ and $\hat{B}_{S,N}$ are defined below. To this end, denote $\nabla t_i^s = \partial t_i^s(\theta)/\partial\theta$, the gradient of the variable with respect to the vector of parameters, and $\overline{\nabla t}_i = \frac{1}{S}\sum_{s=1}^{S}\nabla t_i^s$, its average across all simulations.

$$\hat{A}_{S,N} = \frac{1}{N}\sum_{i=1}^{N}\left[\overline{\nabla t}_i\overline{\nabla t}_i' - \frac{1}{S(S-1)}\sum_{s=1}^{S}\left(\nabla t_i^s - \overline{\nabla t}_i\right)\left(\nabla t_i^s - \overline{\nabla t}_i\right)'\right]$$

$$\hat{B}_{S,N} = \frac{1}{N}\sum_{i=1}^{N} d_{S,i}(\theta)d_{S,i}(\theta)'$$

with $d_{S,i}$ a k dimensional vector:

$$d_{S,i}(\theta) = (t_i - \bar{t}_i(\theta))\overline{\nabla t}_i(\theta) + \frac{1}{S(S-1)}\sum_{s=1}^{S}[t_i^s(\theta) - \bar{t}(\theta)]\nabla t_i^s(\theta)$$