Lectures 6 and 7 . Dynamic discrete choice: full solution methods

Ciprian Domnisoru

Aalto University

Overview of Rust (1987)

This is a path-breaking paper that introduces a methodology to estimate a single-agent dynamic discrete choice (DDC) models.

Main contributions

1. An illustrative application in a simple model of engine replacement.
2. Development and implementation of Nested Fixed Point Algorithm
3. Formulation of assumptions that makes DDC models tractable
4. The first researcher to obtain ML estimates of DDC models
5. Bottom-up approach: Micro-aggregated demand for durable assets

Policy experiments:

- How does changes in replacement cost affect the demand for engines and the equilibrium distribution of mileage?

Dynamic discrete choice examples

- Occupational Choice (Keane and Wolpin, JPE 1997)
- Retirement (Rust and Phelan, ECMA 1997)
- Brand choice and advertising (Erdem and Keane, MaScience 1996)
- Choice of college major (Arcidiacono, JoE 2004)
- Individual migration decisions (Kennan and Walker, ECMA 2011)
- High school attendance and work decisions (Eckstein and Wolpin, ECMA 1999)
- Sales and dynamics of consumer inventory behavior (Hendel and Nevo, ECMA 2006)
- Advertising, learning, and consumer choice in experience good markets (Ackerberg, IER 2003)
- Route choice models (Fosgerau et al, Transp. Res. B)
- Fertility and labor supply decisions (Francesconi, JoLE 2002)
- Residential and Work-location choice (Buchinsky et al, ECMA 2015)

Methods used to estimate dynamic discrete choice methods

- Rust (1987): MLE using Nested-Fixed Point Algorithm (NFXP)
- Hotz and Miller (1993): CCP estimator - (two step estimator)
- Keane and Wolpin (1994): Simulation and interpolation
- Rust (1997): Randomization algorithm (breaks curse of dimensionality)
- Aguirregabiria and Mira (2002): Nested Pseudo Likelihood (NPL).
- Bajari, Benkard and Levin (2007): Two step-minimum distance (equilibrium inequalities).
- Arcidiacono Miller (2002): CCP with unobserved heterogeneity (EM Algorithm).
- Norets (2009): Bayesian Estimation (allows for serial correlation in ϵ)
- Su and Judd (2012): MLE using constrained optimization (MPEC)

Formulating, solving and estimating a dynamic model

Components of the dynamic model

- Decision variables: vector describing the choices, $d_t \in C(s_t)$
- State variables: vector of variables, s_t, that describe all relevant information about the modeled decision process
- Instantaneous payoff: utility function, $u(s_t, d_t)$, with time separable discounted utility
- Motion rules: agent's beliefs of how state variable evolve through time, conditional on states and choices. Here formalized by a Markov transition density $p(s_{t+1} \mid s_t, d_t)$

Solution is given by:

- **Value function**: maximum attainable utility $V(s_t)$
- **Policy function**: mapping from state space to action space that returns the optimal choice, $d^\star(s_t)$

Structural Estimation

- Parametrize model: utility function $u(s_t, d_t; \theta_u)$, motion rules for states $p(s_{t+1} \mid s_t, d_t; \theta_p)$, choice sets $C(s_t; \theta_c)$, etc.
- Search for (policy invariant) parameters θ so that model fits targeted aspects of data on (a subset of) decisions, states, payoff's, etc.

Zurcher's Bus Engine Replacement Problem

- Choice set: Binary choice set, $C(x_t) = \{0, 1\}$.
  - Engine replacement ($d_t = 1$) or ordinary maintenance ($d_t = 0$)
- State variables: Harold Zurcher observes $s_t = (x_t, \varepsilon_t)$:
  - x_t: mileage at time t since last engine overhaul/replacement
  - $\varepsilon_t = [\varepsilon_t(d_t = 0), \varepsilon_t(d_t = 1)]$: decision specific state variable
- Utility function: $U(x_t, \varepsilon_t, d_t; \theta_1) =$

$$u(x_t, d_t, \theta_1) + \varepsilon_t(d_t) = \begin{cases} -RC - c(0, \theta_1) + \varepsilon_t(1) & \text{if } d_t = 1 \\ -c(x_t, \theta_1) + \varepsilon_t(0) & \text{if } d_t = 0 \end{cases} \tag{1}$$

- State variables process
  - ε_t is iid with conditional density $q(\varepsilon_t | x_t, \theta_2)$
  - x_t (mileage since last replacement)

$$p(x_{t+1} | x_t, d_t, \theta_2) = \begin{cases} g(x_{t+1} - 0, \theta_3) & \text{if } d_t = 1 \\ g(x_{t+1} - x_t, \theta_3) & \text{if } d_t = 0 \end{cases} \tag{2}$$

  If engine is replaced, state of bus regenerates to $x_t = 0$.
- Parameters to be estimated $\theta = (RC, \theta_1, \theta_3)$
  (Fixed parameters: (β, θ_2))

General Behavioral Framework

The decision problem

- The decision maker chooses a sequence of actions to maximize expected discounted utility over a (in)finite horizon

$$V_\theta(s_t) = \sup_\Pi E\left[\sum_{j=0}^{T} \beta^j U(s_{t+j}, d_{t+j}; \theta_1) \,|\, s_t, d_t\right]$$

where

- $\Pi = (f_t, f_{t+1}, ..,), d_t = f_t(s_t, \theta) \in C(x_t) = \{1, 2, .., J\}$
- $\beta \in (0,1)$ is the discount factor
- $U(s_t, d_t; \theta_1)$ is a choice and state specific utility function
- We may consider an infinite horizon , i.e. $T = \infty$
- E summarizes expectations of future states given s_t and d_t

Recursive form of the maximization problem

- By Bellman Principle of Optimality, the value function $V(s)$ constitutes the solution of the following functional (Bellman) equation

$$V(x,\varepsilon) \equiv T(V)(x,\varepsilon) = \max_{d \in C(x)} \left\{ u(x,\varepsilon,d) + \beta E\left[V(x',\varepsilon') \middle| x,\varepsilon,d\right] \right\}$$

- Expectations are taken over the next period values of state $s' = (x',\varepsilon')$ given it's controlled motion rule, $p(s' \mid s,d)$

$$E\left[V(x',\varepsilon') \middle| x,\varepsilon,d\right] = \int_X \int_\Omega V(x',\varepsilon') p(x',\varepsilon' | x,\varepsilon,d) dx' d\varepsilon'$$

where $\varepsilon = (\varepsilon(1),\dots,\varepsilon(J)) \in \mathbb{R}^J$

Hard to compute fixed point V such that $T(V) = V$

- x is continuous and ε is continuous and J-dimensional
- $V(x,\varepsilon)$ is high dimensional
- Evaluating E may require high dimensional integration
- Evaluating $V(x',\varepsilon')$ may require high dimensional interpolation/approximation
- $V(x,\varepsilon)$ is non-differentiable

Rust's Assumptions

1. Additive separability in preferences (**AS**):

$$U(s_t, d) = u(x_t, d; \theta_1) + \varepsilon_t(d)$$

2. Conditional independence (**CI**):
   State variables, $s_t = (x_t, \varepsilon_t)$ obeys a (<u>conditional independent</u>) controlled Markov process with probability density

$$p(x_{t+1}, \varepsilon_{t+1} | x_t, \varepsilon_t, d, \theta_2, \theta_3) = q(\varepsilon_{t+1} | x_{t+1}, \theta_2) p(x_{t+1} | x_t, d, \theta_3)$$

3. Extreme value Type I (EV1) distribution of ε (**EV**)
   Each of the choice specific state variables, $\varepsilon_t(d)$ are assumed to be iid. <u>extreme value distributed</u> with CDF

$$F(\varepsilon_t(d); \mu, \lambda) = \exp(-\exp(-(\varepsilon_t(d) - \mu)/\lambda)) \text{ for } \varepsilon_t(d) \in \mathbb{R}$$

   with $\mu = 0$ and $\lambda = 1$

Rust's Assumptions simplifies DP problem

$$V(x,\varepsilon) = \max_{d\in C(x)} \left\{ u(x,d) + \varepsilon(d) + \beta \int_X \int_\Omega V(x',\varepsilon') p(x'|x,d) q(\varepsilon'|x') dx' d\varepsilon' \right\}$$

1. Separate out the deterministic part of choice specific value $v(x,d)$ (assumptions SA and CI)
2. Reformulate Bellman equation on reduced state space (assumption CI)
3. Compute the expectation of maximum using properties of EV1 (assumption EV)

1. DP problem under AS and CI

Separate out the deterministic part of choice specific value $v(x, d)$

$$V(x,\varepsilon) = \max_{d\in C(x)} \Big\{u(x,d) + \beta \int_X \Big(\int_\Omega V(x',\varepsilon') q(\varepsilon'|x') d\varepsilon' \Big) p(x'|x,d) dx' + \varepsilon(d)\Big\}$$

So that

$$V(x',\varepsilon') = \max_{d\in C} \big\{ v(x',d) + \varepsilon'(d) \big\}$$

where

$$v(x,d) = u(x,d) + \beta E\big[V(x',\varepsilon')\big|x,d\big]$$

2a. Bellman equation in expected value function space

Let $EV(x,d) = E\left[V(x',\varepsilon')\middle|x,d\right]$ denote the expected value function.

Because of CI we can now express the Bellman equation in expected value function space

$$EV(x,d) = \Gamma(EV)(x,d) \equiv \int_X \int_\Omega [V(x',\varepsilon')q(\varepsilon'|x')d\varepsilon']\, p(x'|x,d)dx'$$

where

$$V(x',\varepsilon') = \max_{d'\in C(x')} [u(x',d') + \beta EV(x',d') + \varepsilon'(d')]$$

- Γ is a contraction mapping with unique fixed point EV, i.e. $\|\Gamma(EV) - \Gamma(W)\| \leq \beta \|EV - W\|$
- Global convergence of VFI
- $EV(x,d)$ is lower dimensional: does not depend on ε

2b. Bellman equation in integrated value function space

Let $\bar{V}(x) = E\big[V(x,\varepsilon)\big|x\big]$ denote the integrated value function

Because of CI we can express Bellman equation in integrated value function space

$$\bar{V}(x) = \bar{\Gamma}(\bar{V})(x) \equiv \int_{\Omega} V(x,\varepsilon)q(\varepsilon|x)d\varepsilon$$

where

$$V(x,\varepsilon) = \max_{d\in C(x)} [u(x,d) + \varepsilon(d) + \beta \int_X \bar{V}(x')p(x'|x,d)dx']$$

- $\bar{\Gamma}$ is a contraction mapping with unique fixed point $\bar{V}$, i.e. $\left\|\bar{\Gamma}\left(\bar{V}\right) - \bar{\Gamma}\left(W\right)\right\| \leq \beta \left\|\bar{V} - W\right\|$
- Global convergence of VFI
- $\bar{V}(x)$ is lower dimensional: does not depend on ε and d

3. Compute the expectation of maximum under EV

We can express expectation of maximum using properties of EV1 distribution (assumption EV)

Expectation of maximum, $\bar{V}(x)$, can be expressed as "the log-sum"

$$\bar{V}(x) = E\left[\max_{d\in\{1,\dots,J\}}\{v(x,d)+\lambda\varepsilon(d)\} \mid x\right] = \lambda \log \sum_{j=1}^{J} \exp(v(x,d)/\lambda)$$

Conditional choice probability, $P(x,d)$ has closed form logit expression

$$\begin{aligned} P(d \mid x) &= E\left[\mathbb{1}\left\{d = \arg\max_{j\in\{1,\dots,J\}}\{v(x,j)+\lambda\varepsilon(j)\}\right\} \mid x\right] \\ &= \frac{\exp(v(x,d)/\lambda)}{\sum_{j=1}^{J}\exp(v(x,j)/\lambda)} \end{aligned}$$

HUGE benefits

- Avoids J dimensional numerical integration over ε
- $P(d \mid x)$, $\bar{V}(x)$ and $EV(x,d)$ are smooth functions.

The DP problem under AS, CI and EV

Putting all this together

- Conditional Choice Probabilities (CCPs) are given by

$$P(d|x,\theta) = \frac{\exp\{u(x,d,\theta_1) + \beta EV_\theta(x,d)\}}{\sum_{j\in C(x)} \exp\{u(x,j,\theta_1) + \beta EV_\theta(x,j)\}}$$

- The expected value function can be found as the unique fixed point to the contraction mapping Γ_θ, defined by

$$\begin{aligned} EV_\theta(x,d) &= \Gamma_\theta(EV_\theta)(x,d) \\ &= \int_y \ln\left[\sum_{d'\in C(y)} \exp\left[u(y,d';\theta_1) + \beta EV_\theta(y,d')\right]\right] \\ &\quad p(dy|x,d,\theta_2) \end{aligned}$$

- We have used the subscript θ to emphasize that the Bellman operator,Γ_θ depends on the parameters.
- In turn, the fixed point, EV_θ, and the resulting CCPs, $P(d|x,\theta)$ are implicit functions of the parameters we wish to estimate.

How to deal with continuous mileage state?

Rust discretize the mileage state space x into n grid points

$$X = \{x_1, ..., x_n\} \text{ with } x_1 = 0$$

Mileage transition probability: for $l = 0, \ldots, L$

$$p(x'|\hat{x}_k, d, \theta_2) = \begin{cases} Pr\{x' = x_{k+l}|\theta_2\} = \pi_l \text{ if } d = 0 \\ Pr\{x' = x_{1+l}|\theta_2\} = \pi_l \text{ if } d = 1 \end{cases}$$

- where $\theta_2 = [\pi_1, \ldots, \pi_L]$, $\pi_0 = 1 - \sum_{l=1}^{L} \pi_l$, and $\pi_l \geq 0$
- Mileage in the next period x' can move up at most L grid points.
- L is determined by the empirical distribution of mileage.

Transition matrix for mileage is sparse

Transition matrix conditional on keeping engine

$$
\Pi(d = \text{keep})_{nxn} = \begin{pmatrix}
\pi_0 & \pi_1 & \pi_2 & 0 & \cdot & \cdot & \cdot & 0 \\
0 & \pi_0 & \pi_1 & \pi_2 & 0 & \cdot & \cdot & 0 \\
0 & 0 & \pi_0 & \pi_1 & \pi_2 & 0 & \cdot & 0 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \\
0 & & & & \pi_0 & \pi_1 & \pi_2 & 0 \\
0 & & & & & \pi_0 & \pi_1 & \pi_2 \\
0 & & & & & & \pi_0 & 1 - \pi_0 \\
0 & 0 & & & & & & 1
\end{pmatrix}
$$

Transition matrix for mileage is sparse

Transition matrix conditional on replacing engine

$$\Pi(d = \text{replace})_{nxn} = \begin{pmatrix} \pi_0 & \pi_1 & \pi_2 & 0 & \cdot & \cdot & \cdot & 0 \\ \pi_0 & \pi_1 & \pi_2 & 0 & \cdot & \cdot & \cdot & 0 \\ \pi_0 & \pi_1 & \pi_2 & 0 & \cdot & \cdot & \cdot & 0 \\ \pi_0 & \pi_1 & \pi_2 & 0 & \cdot & \cdot & \cdot & 0 \\ \pi_0 & \pi_1 & \pi_2 & 0 & \cdot & \cdot & \cdot & 0 \\ \pi_0 & \pi_1 & \pi_2 & 0 & \cdot & \cdot & \cdot & 0 \\ \pi_0 & \pi_1 & \pi_2 & 0 & \cdot & \cdot & \cdot & 0 \end{pmatrix}$$

Bellman equation in matrix form

Bellman equation in expected value function space

$$EV(d) = \Gamma(EV) = \Pi(d) \ln \left[\sum_{d'} \exp[u(d') + \beta EV(d')] \right]$$

Bellman equation in integrated value function space

$$\bar{V} = \bar{\Gamma}(\bar{V}) = \ln \left[\sum_{d'} \exp[u(d') + \beta \Pi(d') \bar{V}] \right]$$

where

- $u(d) = [u(x_1, d), .., u(x_n, d)]$
- $EV(d) = [EV(x_1, d), .., EV(x_n, d)]$
- $\bar{V} = [\bar{V}(x_1), .., \bar{V}(x_n)]$
- $\Pi(d)$ is a $n \times n$ state transition matrix conditional on decision d

Structural Estimation

Data: $(d_{i,t}, x_{i,t})$, $t = 1, ..., T_i$ and $i = 1, ..., N$

Log likelihood function

$$L(\theta, EV_\theta)) = \sum_{i=1}^{N} \ell_i^f(\theta, EV_\theta)$$

$$\ell_i^f(\theta, EV_\theta) = \sum_{t=2}^{T_i} log(P(d_{i,t}|x_{i,t}, \theta)) + \sum_{t=2}^{T_i} log\left(p(x_{i,t}|x_{i,t-1}, d_{i,t-1}, \theta_3)\right)$$

where

$$P(d|x, \theta) = \frac{\exp\{u(x, d, \theta_1) + \beta EV_\theta(x, d)\}}{\sum_{d' \in \{0,1\}}\{u(x, d', \theta_1) + \beta EV_\theta(x, d')\}}$$

and

$$\begin{aligned} EV_\theta(x, d) &= \Gamma_\theta(EV_\theta)(x, d) \\ &= \int_y \ln\left[\sum_{d' \in \{0,1\}} \exp[u(y, d'; \theta_1) + \beta EV_\theta(y, d')]\right] p(dy|x, d, \theta_3) \end{aligned}$$

The Nested Fixed Point Algorithm

Since the contraction mapping Γ_θ always has a unique fixed point, the constraint $EV_\theta = \Gamma(EV_\theta)$ implies that the fixed point EV_θ is an <u>implicit function</u> of θ.

Hence, NFXP solves the <u>unconstrained</u> optimization problem

$$\max_\theta L(\theta, EV_\theta)$$

Outer loop (Hill-climbing algorithm):

- Likelihood function $L(\theta, EV_\theta)$ is maximized w.r.t. θ
- Quasi-Newton algorithm: Usually BHHH, BFGS or a combination.
- Each evaluation of $L(\theta, EV_\theta)$ requires solution of EV_θ

Inner loop (fixed point algorithm):
The implicit function EV_θ defined by $EV_\theta = \Gamma(EV_\theta)$ is solved by:

- Successive Approximations (SA)
- Newton-Kantorovich (NK) Iterations

NFXP vs. MPEC

- The MPEC method does not need a specialized inner loop algorithm to compute the fixed point.
- Instead, it recasts the problem of maximizing the likelihood function as a constrained optimization problem with respect to the K structural parameters plus N additional variables, which are the values of value function at a set of N grid points in the state space.
- These N additional variables must satisfy the Bellman equation, which can be recast as a N "side constraints".
- MPEC also implicitly solves the fixed point problem while searching for structural parameter values that maximize the likelihood, but using a general-purpose constrained optimization algorithm

Mathematical Programming with Equilibrium Constraints

MPEC solves the constrained optimization problem

$$\max_{\theta, EV} L(\theta, EV) \text{ subject to } EV = \Gamma_{\theta}(EV)$$

using general-purpose constrained optimization solvers such as KNITRO

Su and Judd (Ecta 2012) considers two such implementations:

MPEC/AMPL:

- AMPL formulates problems and pass it to KNITRO.
- Automatic differentiation (Jacobian and Hessian)
- Sparsity patterns for Jacobian and Hessian

MPEC/MATLAB:

- User need to supply Jacobians, Hessian, and Sparsity Patterns
- Su and Judd do not supply analytical derivatives.
- ktrlink provides link between MATLAB and KNITRO solvers.

MPEC is used in multiple contexts

Single-Agent Dynamic Discrete Choice Models

- Rust (1987): Bus-Engine Replacement Problem
- Nested-Fixed Point Problem (NFXP)
- Su and Judd (2012): Constrained Optimization Approach

Random-Coefficients Logit Demand Models

- BLP (1995): Random-Coefficients Demand Estimation
- Nested-Fixed Point Problem (NFXP)
- Dube, Fox and Su (2012): Constrained Optimization Approach

Estimating Discrete-Choice Games of Incomplete Information

- Aguirregabiria and Mira (2007): NPL (Recursive 2-Step)
- Bajari, Benkard and Levin (2007): 2-Step
- Pakes, Ostrovsky and Berry (2007): 2-Step
- Pesendorfer and Schmidt-Dengler (2008): 2-Step
- Pesendorfer and Schmidt-Dengler (2010): comments on AM (2007)
- Kasahara and Shimotsu (2012): Modified NPL
- Su (2013), Egesdal, Lai and Su (2014): Constrained Optimization

Monte Carlo: Rust's Table X - Group 1,2, 3

- Fixed point dimension: $n = 175$
- Maintenance cost function: $c(x, \theta_1) = 0 : 001 * \theta_1 * x$
- Mileage transition: stay or move up at most $L = 4$ grid points
- True parameter values:
  - $\theta_1 = 2 : 457$
  - $RC = 11.726$
  - $\theta_2 = (\pi_1, \pi_2, \pi_3, \pi_4) = (0.0937, 0.4475, 0.4459, 0.0127)$
- Solve for EV at the true parameter values
- Simulate 250 datasets of monthly data for 10 years and 50 buses

Is NFXP a dinosaur method?

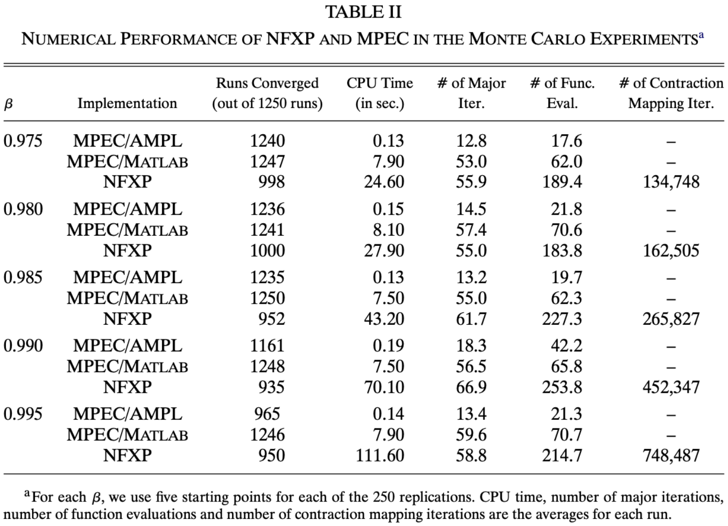


TABLE II

NUMERICAL PERFORMANCE OF NFXP AND MPEC IN THE MONTE CARLO EXPERIMENTS[a]

β	Implementation	Runs Converged (out of 1250 runs)	CPU Time (in sec.)	# of Major Iter.	# of Func. Eval.	# of Contraction Mapping Iter.
0.975	MPEC/AMPL	1240	0.13	12.8	17.6	–
	MPEC/MATLAB	1247	7.90	53.0	62.0	–
	NFXP	998	24.60	55.9	189.4	134,748
0.980	MPEC/AMPL	1236	0.15	14.5	21.8	–
	MPEC/MATLAB	1241	8.10	57.4	70.6	–
	NFXP	1000	27.90	55.0	183.8	162,505
0.985	MPEC/AMPL	1235	0.13	13.2	19.7	–
	MPEC/MATLAB	1250	7.50	55.0	62.3	–
	NFXP	952	43.20	61.7	227.3	265,827
0.990	MPEC/AMPL	1161	0.19	18.3	42.2	–
	MPEC/MATLAB	1248	7.50	56.5	65.8	–
	NFXP	935	70.10	66.9	253.8	452,347
0.995	MPEC/AMPL	965	0.14	13.4	21.3	–
	MPEC/MATLAB	1246	7.90	59.6	70.7	–
	NFXP	950	111.60	58.8	214.7	748,487

[a] For each β, we use five starting points for each of the 250 replications. CPU time, number of major iterations, number of function evaluations and number of contraction mapping iterations are the averages for each run.

NFXP survival kit

Step 1: Read NFXP manual and print out NFXP pocket guide
Step 2: Recenter logit and logsum formulas
Step 3: Use Fixed Point Poly-Algorithm (SA+NK)
Step 4: Provide analytical gradients of Bellman operator
Step 5: Provide analytical gradients of likelihood
Step 6: Use BHHH (outer product of gradients as hessian approx.)

STEP 1: NFXP documentation

References

- Rust (1987): "Optimal Replacement of GMC Bus Engines: An Empirical Model of Harold Zurcher" Econometrica 55-5, pp 999-1033.
- Rust (2000): “Nested Fixed Point Algorithm Documentation Manual: Version 6” `https://editorialexpress.com/jrust/nfxp.html`
- Iskhakov, F. , J. Rust, B. Schjerning, L. Jinhyuk, and K. Seo (2015): "Constrained Optimization Approaches to Estimation of Structural Models : Comment." Econometrica 84-1, pp. 365-370.

STEP 2: Recenter to ensure numerical stability

Logit formulas must be reentered.

$$P_i = \frac{\exp(v_i)}{\sum_j \exp(v_j)} = \frac{\exp(v_i - v_0)}{\sum_j \exp(v_j - v_0)}$$

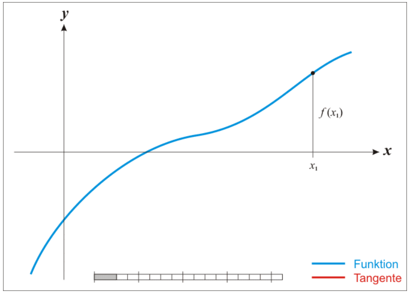
and "log-sum" must be recentered too

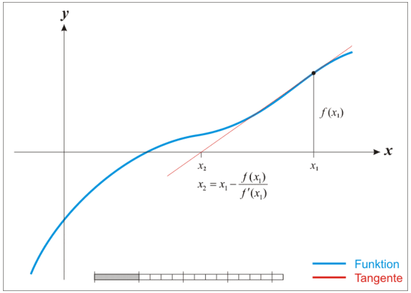
$$\ln \sum_j \exp(v_j) = v_0 + \ln \sum_j \exp(v_j - v_0)$$

If v_0 is chosen to be $v_0 = \max_j v_j$ we can avoid numerical instability due to overflow/underflow

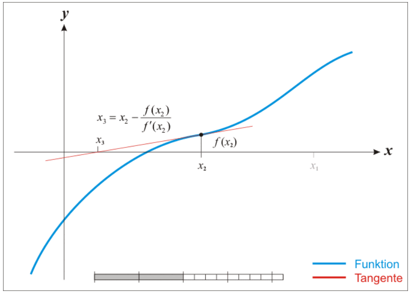
Newton's method

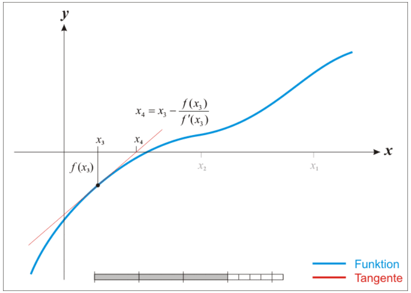
For a continous f(x), at point a, p(x) $\approx f(a) + f'(a)(x-a) + \frac{f''(a)}{2}(x-a)^2$ is the second order approximation. Minimize f by minimizing p(x). $x_m = a - \frac{f'(a)}{f''(a)}$. Finds critical points, not min or max.

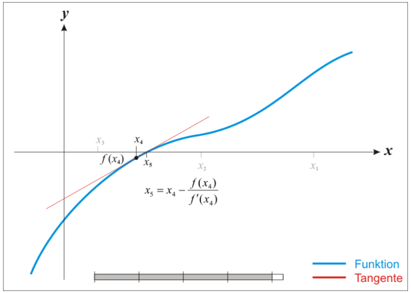




Newton's method







STEP 3: Use Fixed Point Poly-Algorithm (SA+NK)

Problem: Find fixed point of the contraction mapping, Γ_θ

$$EV_\theta = \Gamma(EV_\theta)$$

Fixed Point Poly-Algorithm:

1. Successive Approximations (SA) by contraction iteration:

$$EV_{k+1} = \Gamma_\theta(EV_k)$$

   - Error bound: $||EV_{k+1} - EV|| \leq \beta ||EV_k - EV||$
     $\rightarrow$ Linear convergence $\rightarrow$ slow when β close to 1

2. Newton-Kantorovich (NK) iteration:
   - Solve $F = [I - \Gamma](EV_\theta) = 0$ using Newtons method

$$EV_{k+1} = EV_k - (I - \Gamma')^{-1}(I - \Gamma)(EV_k)$$

     Γ'_θ is the Fréchet derivative of Γ_θ
     I is the identity operator on B
     0 is the zero element of B
   - Error bound: $||EV_{k+1} - EV|| \leq A||EV_k - EV||^2$
     $\rightarrow$ Quadratic convergence around fixed point, EV

Value function iteration

- Specify some V_0
- Apply the Bellman operator to V_0
- Iterate until convergence

The contraction mapping iteration is:

$$\bar{V}^{i+1}(x_t) = \gamma + log\{exp[-c(x_t, \theta_1) + \beta\theta_{31}\bar{V}^i(x_t) + \beta\theta_{32}\bar{V}^i(x_{t+1}) + \beta\theta_{33}\bar{V}^i(x_{t+2})] + exp[-c(0, \theta_1) - RC + \beta\theta_{31}\bar{V}^i(0) + \beta\theta_{32}\bar{V}^i(1) + \beta\theta_{33}\bar{V}^i(2)]\}$$

When to switch to Newton-Kantorovich?

When to switch to Newton-Kantorovich?

- Suppose that $EV_0 = EV + k$.
  (Initial EV_0 equals fixed point EV plus an arbitrary constant)
- Another successive approximation does not solve this:

$$
\begin{aligned}
tol_0 &= \|EV_0 - \Gamma(EV_0)\| = \|EV + k - \Gamma(EV + k)\| \\
&= \|EV + k - (EV + \beta k)\| = (1-\beta)k \\
tol_1 &= \|EV_1 - \Gamma(EV_1)\| = \|EV + \beta k - \Gamma(EV + \beta k)\| \\
&= \|EV + \beta k - (EV + \beta^2 k)\| = \beta(1-\beta)k \\
tol_1/tol_0 &= \beta
\end{aligned}
$$

- Newton will immediately "strip away" the irrelevant constant k
- Switch to Newton whenever tol_1/tol_0 is sufficiently close to β

The Fixed Point (poly) Algorithm

Fixed Point poly Algorithm

1. Successive contraction iterations

$$EV_{k+1} = \Gamma_\theta(EV_k)$$

   until EV_k is in the domain of attraction
   (i.e. when tol_{k+1}/tol_k is close to β)

2. Newton-Kantorovich (quadratic convergence)

$$EV_{k+1} = EV_k - (I - \Gamma')^{-1}(I - \Gamma)(EV_k)$$

   until convergence
   (i.e. when $\|EV_{k+1} - EV_k\|$ is close to machine precision)

STEP 4: Analytical derivative of Bellman operator

Derivative of Bellman operator, $\bar{\Gamma}'$

- Needed for the NK iteration
- In the discretized approximation, $\bar{\Gamma}'$ is a $n \times n$ matrix with partial derivatives of the $n \times 1$ vector function $\bar{\Gamma}(V_\theta)$ with respect to the $n \times 1$ vector $\bar{V}_\theta$
- $\bar{\Gamma}'_\theta$ is simply β times the choice probability weighted state transition probability matrix

$$\bar{\Gamma}'_\theta = \beta \sum_j \Pi(j). * P(j)$$

- One line of code in MATLAB
- A similar matrix can be derived for Γ'

STEP 1-4: MATLAB implementation of $\bar{\Gamma}_\theta$ and $\bar{\Gamma}'_\theta$

```
function [V1, pk, dBellman_dV]=bellman_iv(V0, mp, u, P)
  vK= u(:,1) + mp.beta*P{1}*V0;      % Value of keeping
  vR= u(:,2) + mp.beta*P{2}*V0;      % Value of replacing

  % Recenter logsum
  maxV=max(vK, vR);
  V1=(maxV + log(exp(vK-maxV)  +  exp(vR-maxV)));

  % If requested, compute keep probability
  if nargout>1
    pk=1./(1+exp((vR-vK)));
  end

  % If requested, compute derivative of Bellman operator
  if nargout>2
    dBellman_dV=mp.beta*(P{1}.*pk + P{2}.*(1-pk));
  end
end
```

STEP 1-4: MATLAB implementation of Γ_θ and Γ'_θ

```
function [ev, pk, dbellman_dev]=bellman_ev(ev0, mp, u, P)
  vK= u(:,1) + mp.beta*ev0;       % Value off keep
  vR= u(:,2) + mp.beta*ev0(1);    % Value of replacing

  % Need to recenter logsum by subtracting max(vK, vR)
  maxV=max(vK, vR);
  V=(maxV + log(exp(vK-maxV)  +  exp(vR-maxV)));
  ev=P{1}*V; % compute expected value of keeping
             % ev(1) is the expected value of replacing

  % If requested, also compute choice probability
  if nargout>1
    pk=1./(1+exp((vR-vK)));
  end

  % If requested, compute derivative of Bellman operator
  if nargout>2
    dbellman_dev=mp.beta*(P{1}.*pk');
    % Add additional term for derivative wrt ev(1),
    % since ev(1) enter logsum for all states
    dbellman_dev(:,1)=dbellman_dev(:,1)+mp.beta*P{1}*(1-pk);
  end
end
```

STEP 5: Provide analytical gradients of likelihood

Simple use of chain rule:

3. Gradients (wrt utility parameters) - similar to standard logit

$$\partial \ell_i^1(\theta)/\partial\theta_1 = \sum_t \sum_j [y_{j(it)} - P(j|x_{it},\theta)]\partial v(x_{it},j)/\partial\theta_1$$

2. Derivative of the choice specific value function

$$\partial v(j)/\partial\theta_1 = \partial u(j)/\partial\theta_1 + \beta\Pi(j)\partial\bar{V}/\partial\theta_1$$

- $\partial u(j)/\partial\theta_1$, is trivial to compute
- $\partial\bar{V}_\theta/\partial\theta$ can be obtained by the implicit function theorem

$$\partial\bar{V}_\theta/\partial\theta = [I - \bar{\Gamma}'_\theta]^{-1}\partial\bar{\Gamma}/\partial\theta$$

where $[I - \bar{\Gamma}'_\theta]^{-1}$ is a by-product of the N-K algorithm!!!.

1. Derivative of Bellman operator wrt. θ_1

$$\partial\bar{\Gamma}/\partial\theta_1 = \beta\sum_j P(j)\cdot\partial u(j)/\partial\theta_1$$

where $\cdot$ is the element by element product

STEP 5: MATLAB implementation of scores

```
function score = score(data, mp, P, pk, px_j, V0, du, dBellman_dV);
  y_j=[(1-data.d) data.d];  % choice dummies [keep replace]

  % Compute scores (use chain rule - three steps)

  % STEP 1: derivative of bellman operator wrt. utility parameters
  dbellman=pk.*du(:,:,1) + (1-pk).*du(:,:,2);
  if strcmp(mp.bellman_type, 'ev');
    dbellman=P{1}*dbellman;
  end

  % STEP 2: derivative of fixed point, V, wrt. utility parameters
  dV=(speye(size(dBellman_dV)) - dBellman_dV)\dbellman;

  % STEP 3: derivative of log-likelihood wrt. utility parameters
  score=0;
  for j=1:size(y_j, 2);
    dv= du(:,:,j) + mp.beta*P{j}*dV;
    score = score+ (y_j(:,j)-px_j(:,j)).*dv(data.x,:);
  end
end
```

STEP 6: BHHH

- Recall Newton-Raphson

$$\theta^{g+1} = \theta^{g} - \lambda \left(\Sigma_i H_i\left(\theta^{g}\right)\right)^{-1} \Sigma_i s_i\left(\theta^{g}\right)$$

- Berndt, Hall, Hall, and Hausman, (1974):
  Use <u>outer product of scores</u> as approx. to Hessian

$$\theta^{g+1} = \theta^{g} + \lambda \left(\Sigma_i s_i s_i'\right)^{-1} \Sigma_i s_i$$

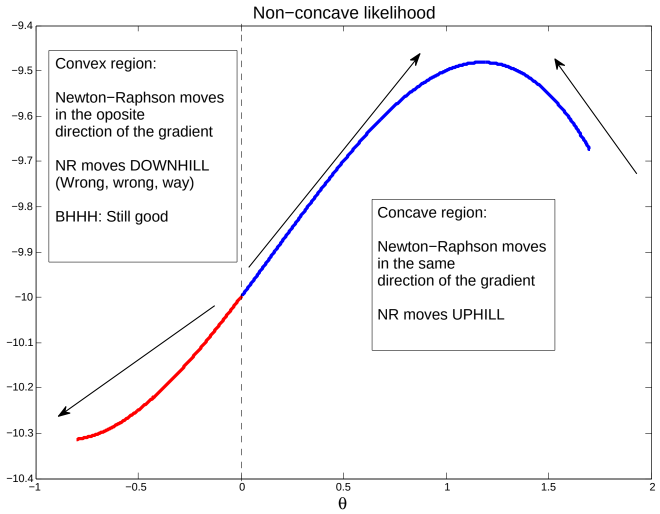
- Why is this valid? Information identity:

$$-E\left[H_i(\theta)\right] \quad = \quad E\left[s_i(\theta)\, s_i(\theta)'\right]$$

(valid for MLE if model is well specified)

STEP 6: BHHH

Some times linesearch may not help Newtons Method



STEP 6: BHHH

Advantages

- $\Sigma_i s_i s_i'$ is always positive definite
  I.e. it always moves uphill for λ small enough
- Does not rely on second derivatives

Disadvantages

- Only a good approximation
  - At the true parameters
  - for large N
  - for well specified models (in principle only valid for MLE)
- Only superlinear convergent - not quadratic

We can always use BHHH for first iterations and the switch to BFGS to update to get an even more accurate approximation to the hessian matrix as the iterations start to converge.