Structural estimation of discrete decision problems

Zurich Initiative on Computational Economics (ZICE) 2017

Fedor Iskhakov, Australian National University
John Rust, Georgetown University
Bertel Schjerning, University of Copenhagen

Saturday, 28 January 2017

Road Map: Discrete Decision Problems

Structural estimation of discrete decision problems

1. **PART I** The Nested Fixed Point Algorithm (NFXP)
   - Leading example: Rust's Engine replacement model
   - The nested fixed point algorithm (NFXP)
   - Counter factual simulations and implied demand (bottom up approach)
2. **PART II**: Mathematical Programming with Equilibrium Constraints
3. **PART III**: Nested Pseudo Likelihood and CCP estimator

Structural Estimation in Microeconomics

Methods for solving Dynamic Discrete Choice Models

- 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).
- Norets (2009): Bayesian Estimation (allows for serial correlation in ϵ)
- Su and Judd (2012): MLE using constrained optimization (MPEC)

PART I

The Nested Fixed Point Algorithm (NFXP)

Rust (ECTA, 1987):

OPTIMAL REPLACEMENT OF GMC BUS ENGINES:

AN EMPIRICAL MODEL OF HAROLD ZURCHER

Bertel Schjerning

Overview of Rust (1987)

The economic question: For how long one should continue to operate and maintain a bus before it is optimal to replace or rebuild the engine?

The model: The optimal replacement decision is the solution to a dynamic optimization problem that formalizes the trade-off between two conflicting objectives:

- *minimizing maintenance and replacement costs versus minimizing unexpected engine failures*
- (the productivity/efficiency of a machine declines over time, but replacing a machine is costly).

Empirical question: Did the decision maker (the superintendent of maintenance, Harold Zurcher) behave according to the optimal replacement rule implied by the theory model?

Structural estimation: Using data on *monthly mileage and engine replacements* for a sample of GMC busses, Rust estimate the structural parameters in the engine replacement model using NFXP

Overview of Rust (1987)

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

Main contributions

1. Development and implementation of Nested Fixed Point Algorithm
2. Formulation of assumptions, that makes dynamic discrete choice models tractable.
3. Bottom-up approach
4. An illustrative application in a simple model of engine replacement.
5. The first researcher to obtain ML estimates of discrete choice dynamic programming models

Policy experiments:

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

General Behavioral Framework

The decision problem

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

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

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 utitty function
- E summarizes expectations of future states given s_t and d_t

Rust's Assumptions

Assumption (CI)

State variables, $s_t = (x_t, \varepsilon_t)$ obeys a (conditional independent) controlled Markov process with probability density

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

Assumption (AS (additive separability))

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

Assumption (XV)

The unobserved state variables, ε_t are assumed to be multivariate iid. extreme value distributed

Object of interest: $\theta = (\beta, \theta_1, \theta_2, \theta_3)$
The vector of structural parameters to be estimated.

The Dynamic Programming Problem

- Under AS, the optimal decision solves the following DP problem

$$V_\theta(x_t, \varepsilon_t) = \max_{d \in C(x_t)} [u(x_t, d, \theta_1) + \varepsilon_t(d) + \beta E(V_\theta(x_{t+1}, \varepsilon_{t+1}) | x_t, \varepsilon_t, d)]$$

- Under (CI) and (XV) we can integrate out the unobserved state variables, such that the unknown function, EV_θ, no longer depends on ε_t.

$$EV_\theta(x, d) = \Gamma_\theta(EV_\theta)(x, d)$$

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

  - (CI): significantly reduces the dimension of the state space
  - (XV): allows us to integrate out the unobserved state variables, ε_t,

- Γ_θ is a *contraction mapping* with unique fixed point EV_θ, i.e. $\|\Gamma(EV) - \Gamma(W)\| \leq \beta \|EV - W\|$

Zurcher's Bus Engine Replacement Problem

- Choice set: Binary choice set, $C(x_t) = \{0, 1\}$. Each bus comes in for repair once a month and Zurcher chooses between ordinary maintenance ($d_t = 0$) and overhaul/engine replacement ($d_t = 1$).
- State variables: Harold Zurcher observes:
  - x_t: mileage at time t since last engine overhaul
  - $\varepsilon_t = [\varepsilon_t(d_t = 0), \varepsilon_t(d_t = 1)]$: other state variable
- Utility function:

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

- State variables process x_t (mileage since last replacement)

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

- If engine is replaced, state of bus regenerates to $x_t = 0$.

Likelihood Function

Likelihood

- Under assumption (CI) the likelihood function ℓ^f has the particular simple form

$$\ell^f\left(x_1, \ldots x_T, d_1, \ldots d_r | x_0, d_0, \theta\right) = \prod_{t=1}^{T} P\left(d_t | x_t, \theta\right) p\left(x_t | x_{t-1}, d_{t-1}, \theta_2\right)$$

  where $P\left(d_t | x_t, \theta\right)$ is the choice probability given the observable state variable, x_t.

How to compute the choice probability, $P\left(d_t | x_t, \theta\right)$

- Need to solve dynamic program

How to estimate the transition probability, $p\left(x_t | x_{t-1}, d_{t-1}, \theta_2\right)$

- Can be estimated non-parametrically or by NLS

Conditional Choice Probabilities

- Under the extreme value assumption choice probabilities are multinomial logistic

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

- The expected value function is given by the unique fixed point to the contraction mapping Γ_θ, defined by

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

- Structural Estimation: Rust's *Nested Fixed Point Algorithm* (NFXP)

Structural Estimation: The Nested Fixed Point Algorithm

Since the contraction mapping Γ always has a unique fixed point, the constraint $EV = \Gamma_\theta(EV)$ implies that the fixed point EV_θ is an *implicit function* of θ.

Hence, NFXP solves the *unconstrained* 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

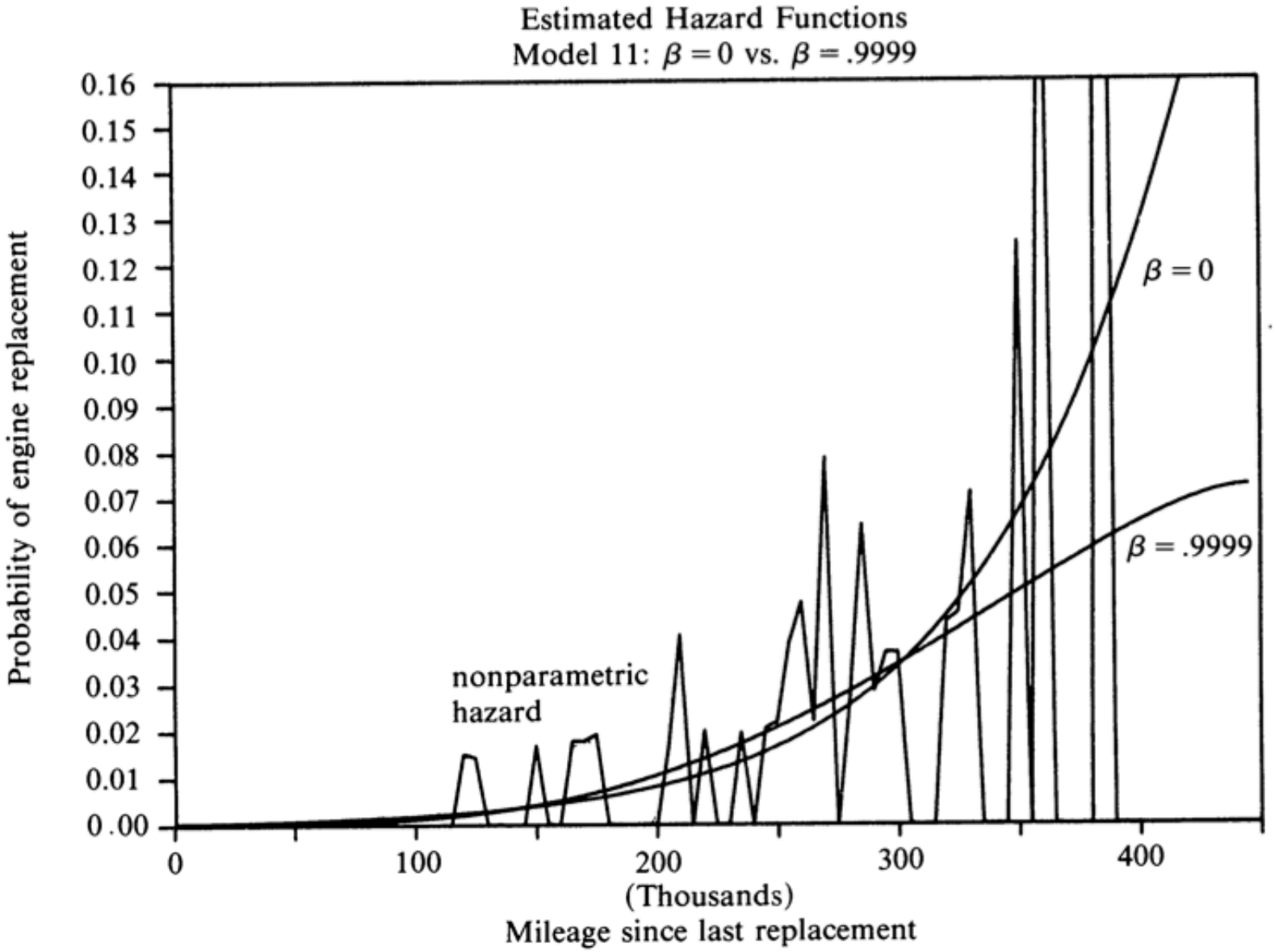
MATLAB implementation of the likelihood function

```
function [f,g,h]=ll(data, mp, P, options, theta)
    global ev0;
    % parse parameters
    mp.RC=theta(1);  mp.c=theta(2); mp.p=abs(theta(3:end));

    % Update u, du and P evaluated in grid points
    dc=0.001*mp.grid; cost=mp.c*0.001*mp.grid;
    P = nfxp.statetransition(mp.p, mp.n);

    % Solve model
    [ev0, pk, F]=nfxp.solve(ev0, P, cost, mp, options);

    % Evaluate log likelihood regarding replacement choice
    lp=pk(data.x);
    logl=log(lp+(1-2*lp).*(data.d));

    % add on log like for mileage process
    p=[mp.p; 1-sum(mp.p)];
    logl=logl + log(p(1+ data.dx1));

    % Objective function (negative mean log likleihood)
    f=mean(-logl);
```

Data

- Harold Zurcher's Maintenance records of 162 busses
- Monthly observations of mileage on each bus (odometer reading)
- Data on maintenance operations
  1. Routine, periodic maintenance (e.g. brake adjustments)
  2. Replacement or repair at time of failure
  3. Major engine overhaul and/or replacement
- Rust focus on 3)

Estimated Hazard Functions



Specification Search

TABLE VIII

SUMMARY OF SPECIFICATION SEARCH[a]

Cost Function	Bus Group 1, 2, 3	Bus Group 4	Bus Group 1, 2, 3, 4
Cubic $c(x, \theta_1) = \theta_{11}x + \theta_{12}x^2 + \theta_{13}x^3$	Model 1 −131.063 −131.177	Model 9 −162.885 −162.988	Model 17 −296.515 −296.411
quadratic $c(x, \theta_1) = \theta_{11}x + \theta_{12}x^2$	Model 2 −131.326 −131.534	Model 10 −163.402 −163.771	Model 18 −297.939 −299.328
linear $c(x, \theta_1) = \theta_{11}x$	Model 3 −132.389 −134.747	Model 11 −163.584 −165.458	Model 19 −300.250 −306.641
square root $c(x, \theta_1) = \theta_{11}\sqrt{x}$	Model 4 −132.104 −133.472	Model 12 −163.395 −164.143	Model 20 −299.314 −302.703
power $c(x, \theta_1) = \theta_{11}x^{\theta_{12}}$	Model 5[b] N.C. N.C.	Model 13[b] N.C. N.C.	Model 21[b] N.C. N.C.
hyperbolic $c(x, \theta_1) = \theta_{11}/(91-x)$	Model 6 −133.408 −138.894	Model 14 −165.423 −174.023	Model 22 −305.605 −325.700
mixed $c(x, \theta_1) = \theta_{11}/(91-x) + \theta_{12}\sqrt{x}$	Model 7 −131.418 −131.612	Model 15 −163.375 −164.048	Model 23 −298.866 −301.064
nonparametric $c(x, \theta_1)$ any function	Model 8 −110.832 −110.832	Model 16 −138.556 −138.556	Model 24 −261.641 −261.641

Structural Estimates, n=90

TABLE IX

STRUCTURAL ESTIMATES FOR COST FUNCTION $c(x, \theta_1) = .001\theta_{11}x$

FIXED POINT DIMENSION = 90

(Standard errors in parentheses)

Parameter		Data Sample			Heterogeneity Test	
Discount Factor	Estimates/ Log-Likelihood	Groups 1, 2, 3 3864 Observations	Group 4 4292 Observations	Groups 1, 2, 3, 4 8156 Observations	LR Statistic ($df = 4$)	Marginal Significance Level
$\beta = .9999$	RC	11.7270 (2.602)	10.0750 (1.582)	9.7558 (1.227)	85.46	1.2E − 17
	θ_{11}	4.8259 (1.792)	2.2930 (0.639)	2.6275 (0.618)		
	θ_{30}	.3010 (.0074)	.3919 (.0075)	.3489 (.0052)		
	θ_{31}	.6884 (.0075)	.5953 (.0075)	.6394 (.0053)		
	LL	−2708.366	−3304.155	−6055.250		
$\beta = 0$	RC	8.2985 (1.0417)	7.6358 (0.7197)	7.3055 (0.5067)	89.73	1.5E − 18
	θ_{11}	109.9031 (26.163)	71.5133 (13.778)	70.2769 (10.750)		
	θ_{30}	.3010 (.0074)	.3919 (.0075)	.3488 (.0052)		
	θ_{31}	.6884 (.0075)	.5953 (.0075)	.6394 (.0053)		
	LL	−2710.746	−3306.028	−6061.641		
Myopia test:	LR Statistic ($df = 1$)	4.760	3.746	12.782		
$\beta = 0$ vs. $\beta = .9999$	Marginal Significance Level	0.0292	0.0529	0.0035		

Structural Estimates, n=175

TABLE X

STRUCTURAL ESTIMATES FOR COST FUNCTION $c(x, \theta_1) = .001\theta_{11}x$

FIXED POINT DIMENSION = 175

(Standard errors in parentheses)

Parameter		Data Sample			Heterogeneity Test	
Discount Factor	Estimates Log-Likelihood	Groups 1, 2, 3 3864 Observations	Group 4 4292 Observations	Groups 1, 2, 3, 4 8156 Observations	LR Statistic (df = 6)	Marginal Significance Level
$\beta = .9999$	RC	11.7257 (2.597)	10.896 (1.581)	9.7687 (1.226)	237.53	1.89E−48
	θ_{11}	2.4569 (.9122)	1.1732 (0.327)	1.3428 (0.315)		
	θ_{30}	.0937 (.0047)	.1191 (.0050)	.1071 (.0034)		
	θ_{31}	.4475 (.0080)	.5762 (.0075)	.5152 (.0055)		
	θ_{32}	.4459 (.0080)	.2868 (.0069)	.3621 (.0053)		
	θ_{33}	.0127 (.0018)	.0158 (.0019)	.0143 (.0013)		
	LL	−3993.991	−4495.135	−8607.889		
$\beta = 0$	RC	8.2969 (1.0477)	7.6423 (.7204)	7.3113 (0.5073)	241.78	2.34E−49
	θ_{11}	56.1656 (13.4205)	36.6692 (7.0675)	36.0175 (5.5145)		
	θ_{30}	.0937 (.0047)	.1191 (.0050)	.1070 (.0034)		
	θ_{31}	.4475 (.0080)	.5762 (.0075)	.5152 (.0055)		
	θ_{32}	.4459 (.0080)	.2868 (.0069)	.3622 (.0053)		
	θ_{33}	.0127 (.0018)	.0158 (.0019)	.0143 (.0143)		
	LL	−3996.353	−4496.997	−8614.238		
Myopia tests:	LR Statistic ($df = 1$)	4.724	3.724	12.698		
$\beta = 0$ vs. $\beta = .9999$	Marginal Significance Level	0.0297	0.0536	.00037		

Estimating parameters, bustypes 1,2,3,4 (model 19)

```
Output from run_nfxp.m

Beta           =    0.99990
n              =  175.00000
Sample size    = 8156.00000

Parameter Estimates
------------------------------------------------------------------------
                            RC                C     Likelihood
------------------------------------------------------------------------
         param           9.769           1.343         -8607.9
          s.e.           1.226           0.315
------------------------------------------------------------------------
```

Identification - scale of cost function

Identification problem?

- We only identify RC/σ and $c(x, \theta_1)/\sigma = 0.001 * \theta_1/\sigma * x$, (where σ is parameter that index the scale of the cost function).
- σ is unidentified form mileage and replacement data

How to deal with identification problem related to scale of utility?

- Using replacement cost data and structural estimates we can obtain a scale estimate
- Scale the estimates with observed average replacement costs

Average Engine Replacement Costs

TABLE III

AVERAGE ENGINE REPLACEMENT COSTS[a]

	Bus Group		
Operation	1, 2, 3	4	1, 2, 3, 4
Labor time[b] to drop engine	$ 150	$ 150	$ 150
Labor time[b] to overhaul engine	3373	2870	3032
Parts required to overhaul engine	5826	4343	4730
Labor time[b] to re-install engine	150	150	150
Total cost of replacement	$9499	$7513	$8062

- Replacement costs are *higher* for group 1,2,3 although engine replacements for this group occur 57.000 miles and 15 month *earlier*
- Presumably operating and maintenance costs for these busses increase much faster

Identification - scale of cost function

- Using replacement cost data (prev. slide) and structural estimates from Table IX (next slide) we can obtain a scale estimate

$$\begin{aligned} \sigma_{bus\ 1,2,3} &= \frac{RC}{RC/\sigma} \\ &= \$9499/11.7257 \\ \sigma_{bus\ 4} &= \$7513/10.0750 \end{aligned}$$

- We can the obtain a dollar estimate of $c(x, \theta_1)$ (i.e monthly maintenance costs per accumulated 5000 miles)

$$\begin{aligned} c(x,\theta_1)_{bus\ 1,2,3} &= \sigma * 0.001\theta_{11}/\sigma * x \\ &= \$9499/11.725 * 0.001 * 4.82 * x = \$3.9 * x \\ c(x,\theta_1)_{bus\ 4} &= \$7513/10.0750 * 0.001 * 2.2930 * x = \$1.7 * x \end{aligned}$$

- Hence, a bus with mileage of 300.000 (i.e. $x = 300.000/5.000$) is $(3.9 - 1.7) * 300000/5000 = 132\$$ more expensive to operate per month

Structural Estimates

TABLE IX

STRUCTURAL ESTIMATES FOR COST FUNCTION $c(x, \theta_1) = .001\theta_{11}x$

FIXED POINT DIMENSION = 90

(Standard errors in parentheses)

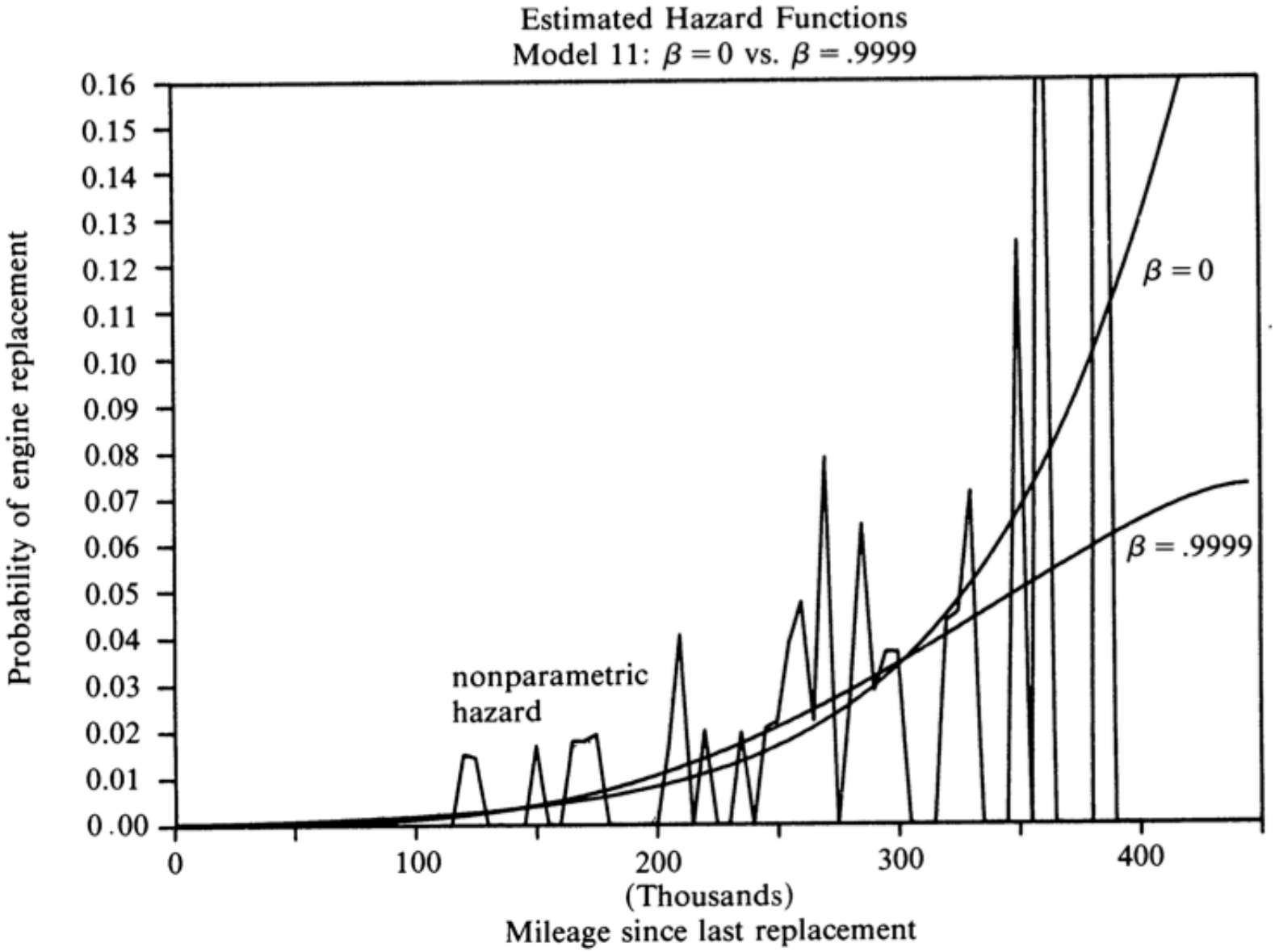
Parameter		Data Sample			Heterogeneity Test	
Discount Factor	Estimates/ Log-Likelihood	Groups 1, 2, 3 3864 Observations	Group 4 4292 Observations	Groups 1, 2, 3, 4 8156 Observations	LR Statistic ($df = 4$)	Marginal Significance Level
$\beta = .9999$	RC	11.7270 (2.602)	10.0750 (1.582)	9.7558 (1.227)	85.46	1.2E − 17
	θ_{11}	4.8259 (1.792)	2.2930 (0.639)	2.6275 (0.618)		
	θ_{30}	.3010 (.0074)	.3919 (.0075)	.3489 (.0052)		
	θ_{31}	.6884 (.0075)	.5953 (.0075)	.6394 (.0053)		
	LL	−2708.366	−3304.155	−6055.250		
$\beta = 0$	RC	8.2985 (1.0417)	7.6358 (0.7197)	7.3055 (0.5067)	89.73	1.5E − 18
	θ_{11}	109.9031 (26.163)	71.5133 (13.778)	70.2769 (10.750)		
	θ_{30}	.3010 (.0074)	.3919 (.0075)	.3488 (.0052)		
	θ_{31}	.6884 (.0075)	.5953 (.0075)	.6394 (.0053)		
	LL	−2710.746	−3306.028	−6061.641		
Myopia test:	LR Statistic ($df = 1$)	4.760	3.746	12.782		
$\beta = 0$ vs. $\beta = .9999$	Marginal Significance Level	0.0292	0.0529	0.0035		

Why a dynamic model?

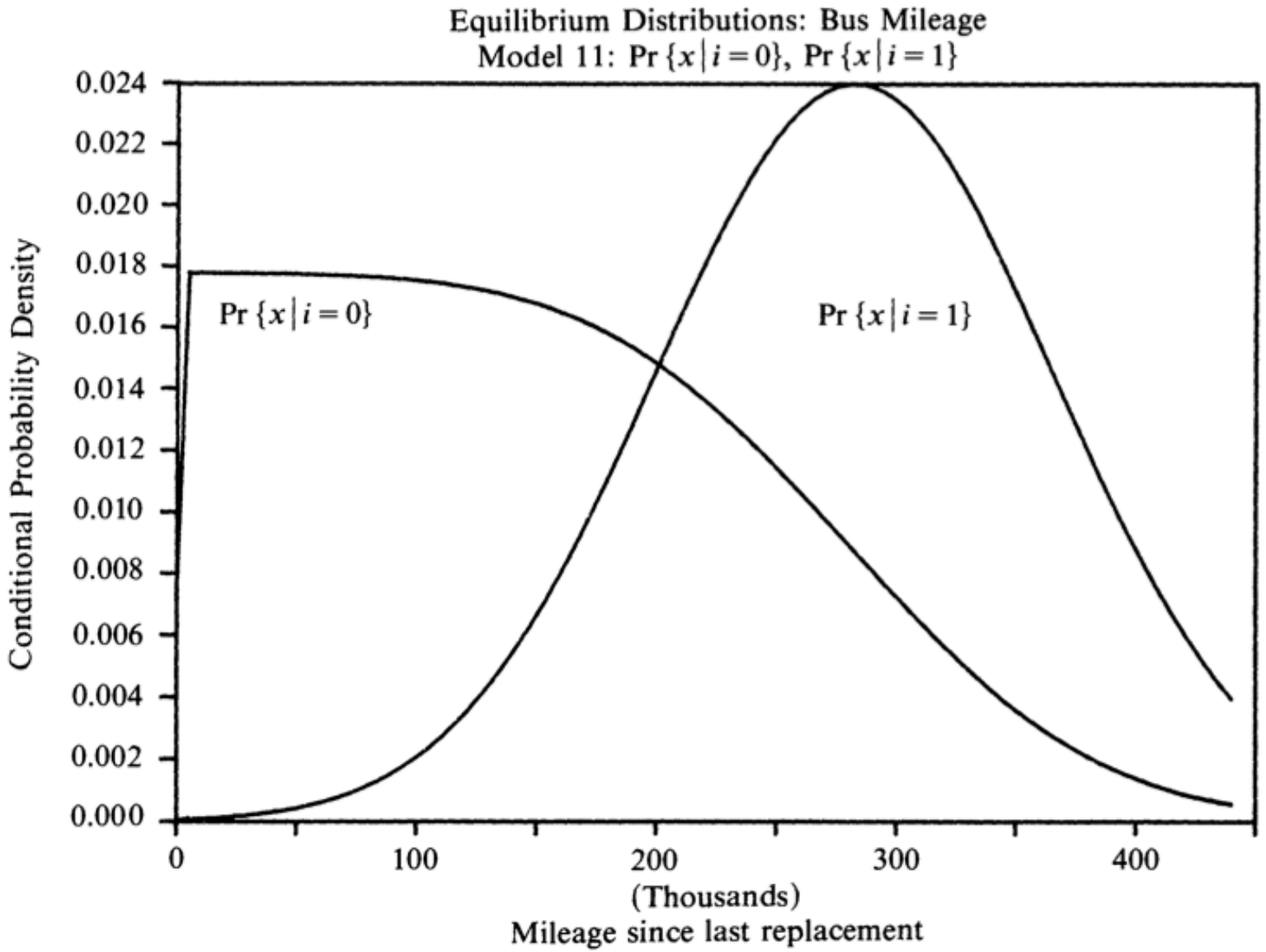
Suppose the "true" β is > 0, but we estimate the model with $\beta = 0$

- Our estimate of the replacement cost function will be biased.
- Parameters RC and θ_1 would be biased too
  (RC is upward biased and θ_1 is downward biased.)
- Predictions using the estimated model will be biased for two reasons:
  1. parameter estimates are biased
  2. the static model is not correct.
- Though the biases introduced by (1) and (2) might partly compensate each other, it will be a very unlikely coincidence that they compensate each other to make the bias negligible.
- Effect on equilibrium demand and hazard functions are very different!

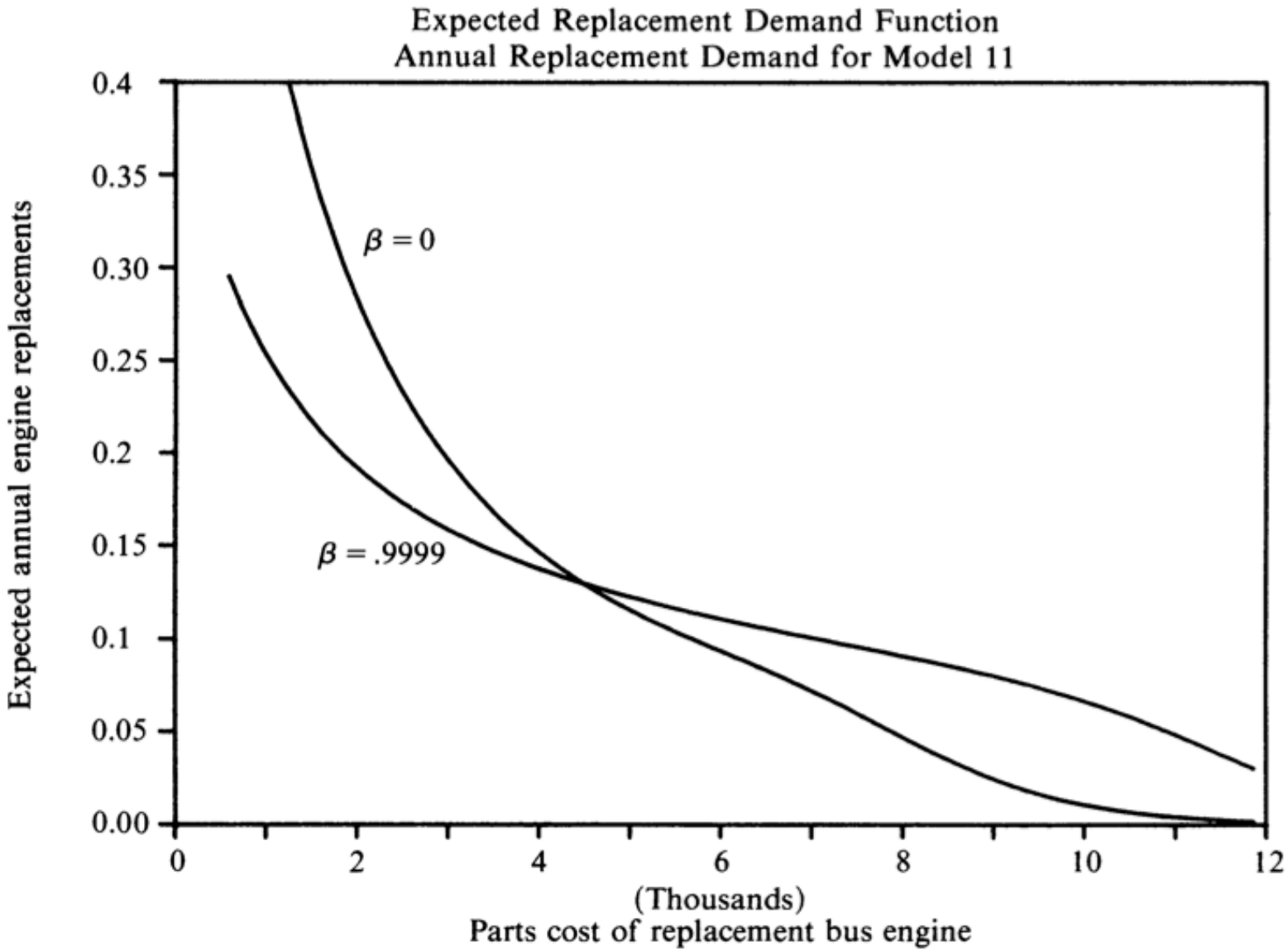
Estimated Hazard Functions



Bus mileage, bus group 4



Demand Function, bus group 4



Why not a reduced form for demand?

Reduced form

- Regress engine replacements on replacement costs

Problem: Lack of variation in replacement costs

- Data would be clustered around the intersection of the demand curves for $\beta = 0$ and $\beta = 0.9999$
  (both models predict that *RC* is around the actual RC of \$4343)
- Demand also depends on how operating costs varies with mileage
- Need exogenous variation in RC
  .... that doesn't vary with operating costs
- Even if we had exogenous variation, this does not help us to understand the underlying economic incentives

Structural Approach

Attractive features

- structural parameters have a transparent interpretation
- evaluation of (new) policy proposals by counterfactual simulations.
- economic theories can be tested directly against each other.
- economic assumptions are more transparent and explicit (compared to statistical assumptions)

Less attractive features

- We impose more structure and make more assumptions
- Truly "structural" (policy invariant) parameters may not exist
- The curse of dimensionality
- The identification problem
- The problem of multiplicity and indeterminacy of equilibria
- Intellectually demanding and a huge amount of work

PART II

Mathematical Programming with Equilibrium Constraints (MPEC)

Bertel Schjerning

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

Zurcher's Bus Engine Replacement Problem

- Choice set: Each bus comes in for repair once a month and Zurcher chooses between ordinary maintenance ($d_t = 0$) and overhaul/engine replacement ($d_t = 1$)
- State variables: Harold Zurcher observes:
  - x_t: mileage at time t since last engine overhaul
  - $\varepsilon_t = [\varepsilon_t(d_t = 0), \varepsilon_t(d_t = 1)]$: other state variable
- Utility function:

$$u(x_t, d, \theta_1) + \varepsilon_t(d_t) = \begin{cases} -RC - c(0, \theta_1) + \varepsilon_t(1) & \text{if } d_t = 1 \\ -c(x_t, \theta_1) + \varepsilon_t(0) & \text{if } d_t = 0 \end{cases} \quad (3)$$

- State variables process x_t (mileage since last replacement)

$$p(x_{t+1}|x_t, d_t, \theta_2) = \begin{cases} g(x_{t+1} - 0, \theta_2) & \text{if } d_t = 1 \\ g(x_{t+1} - x_t, \theta_2) & \text{if } d_t = 0 \end{cases} \quad (4)$$

- If engine is replaced, state of bus regenerates to $x_t = 0$.

Structural Estimation

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

Likelihood function

$$\ell_i^f(\theta) = \sum_{t=2}^{T_i} log(P(d_{i,t}|x_{i,t}, \theta)) + \sum_{t=2}^{T_i} log\left(p(x_{i,t}|x_{i,t-1}, d_{i,t-1}, \theta_2)\right)$$

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

$$\begin{aligned} EV_\theta(x, d) &= \Gamma_\theta(EV_\theta)(x, d) \\ &= \int_y \ln \left[\sum_{d' \in \{0,1\}} \exp[u(y, d'; \theta_1) + \beta EV_\theta(y, d')] \right] p(dy|x, d, \theta_2) \end{aligned}$$

The Nested Fixed Point Algorithm

NFXP solves the *unconstrained* 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

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

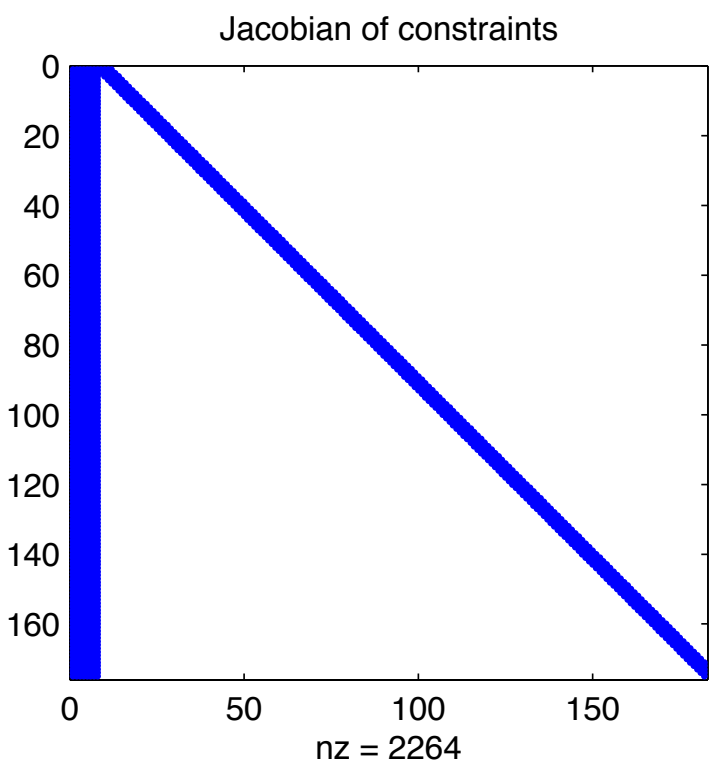
MPEC/MATLAB:

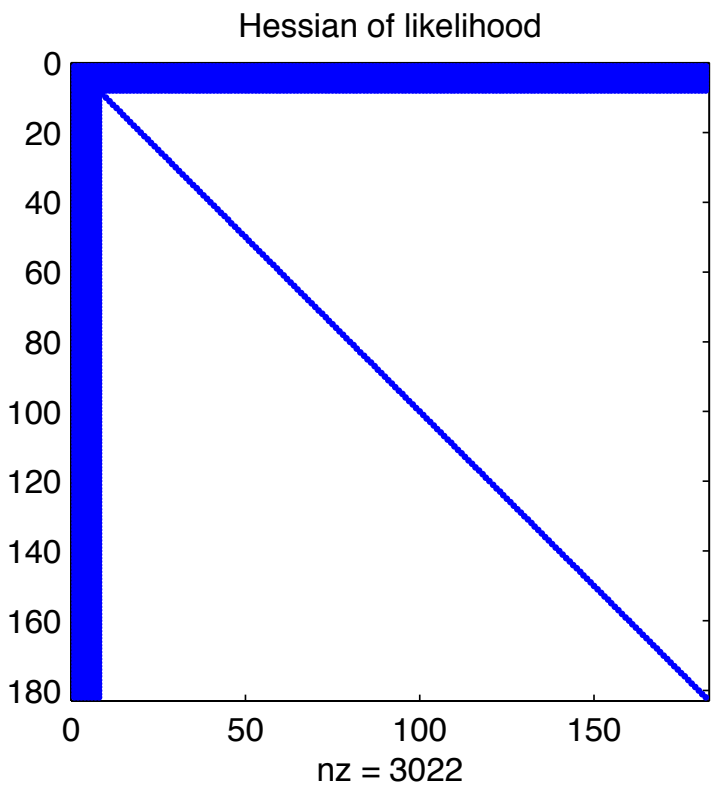
- User need to supply Jacobians, Hessian, and Sparsity Patterns
- Su and Judd do not supply analytical derivatives.
- ktrlink provides link between MATLAB and KNITRO solvers.

Sparsity patterns for MPEC

Two key factors in efficient implementations:

- Provide analytic-derivatives (huge improvement in speed)
- Exploit sparsity pattern in constraint Jacobian (huge saving in memory requirement)





Zurcher's Bus Engine Replacement Problem

Discretize the mileage state space x into n grid points

$$\hat{X} = \{\hat{x}_1, ..., \hat{x}_n\} \text{ with } \hat{x}_1 = 0$$

Mileage transition probability: for $j = 1, ..., J$

$$p(x'|\hat{x}_k, d, \theta_2) = \begin{cases} Pr\{x' = \hat{x}_{k+j}|\theta_2\} = \theta_{2j} \text{ if } d = 0 \\ Pr\{x' = \hat{x}_{1+j}|\theta_2\} = \theta_{2j} \text{ if } d = 1 \end{cases}$$

Mileage in the next period x' can move up at most J grid points. J is determined by the distribution of mileage.

Choice-specific expected value function for $\hat{x} \in \hat{X}$

$$EV_\theta(\hat{x}, d) = \hat{\Gamma}_\theta(EV_\theta)(\hat{x}, d)$$

$$= \sum_j^J \ln \left[\sum_{d' \in D(y)} \exp[u(x', d'; \theta_1) + \beta EV_\theta(x', d')] \right] p(x'|\hat{x}, d, \theta_2)$$

Bellman equation in matrix form

Choice-specific expected value function in compact matrix form

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

where $EV(d) = [EV(1,d), .., EV(n,d)]$ and $u(d) = [u(1,d), .., u(n,d)]$

Transition matrix for mileage is sparse

$$\Pi_{nxn} = \begin{pmatrix} \pi_0 & \pi_1 & \pi_2 & 0 & \cdot & \cdot & \cdot & 0 \\ 0 & \pi_0 & \pi_1 & \pi_2 & 0 & \cdot & \cdot & 0 \\ 0 & 0 & \pi_0 & \pi_1 & \pi_2 & 0 & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \\ 0 & & & & \pi_0 & \pi_1 & \pi_2 & 0 \\ 0 & & & & & \pi_0 & \pi_1 & \pi_2 \\ 0 & & & & & & \pi_0 & 1-\pi_0 \\ 0 & 0 & & & & & & 1 \end{pmatrix}$$

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 $J = 4$ grid points
- True parameter values:
  - $\theta_1 = 2 : 457$
  - $RC = 11.726$
  - $(\theta_{21}, \theta_{22}, \theta_{23}, \theta_{24}) = (0.0937, 0.4475, 0.4459, 0.0127)$
- 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?

Su and Judd (Econometrica, 2012)

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: Solve for fixed point using Newton Iterations
Step 3: Recenter Bellman equation
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

Main references

Rust (1987): "Optimal Replacement of GMC Bus Engines: An Empirical Model of Harold Zurcher" *Econometrica* 55-5 999-1033.

Rust (2000): "Nested Fixed Point Algorithm Documentation Manual: Version 6"
`https://editorialexpress.com/jrust/nfxp.html`

Nested Fixed Point Algorithm

NFXP Documentation Manual version 6, (Rust 2000, page 18):

Formally, one can view the nested fixed point algorithm as solving the following constrained optimization problem:

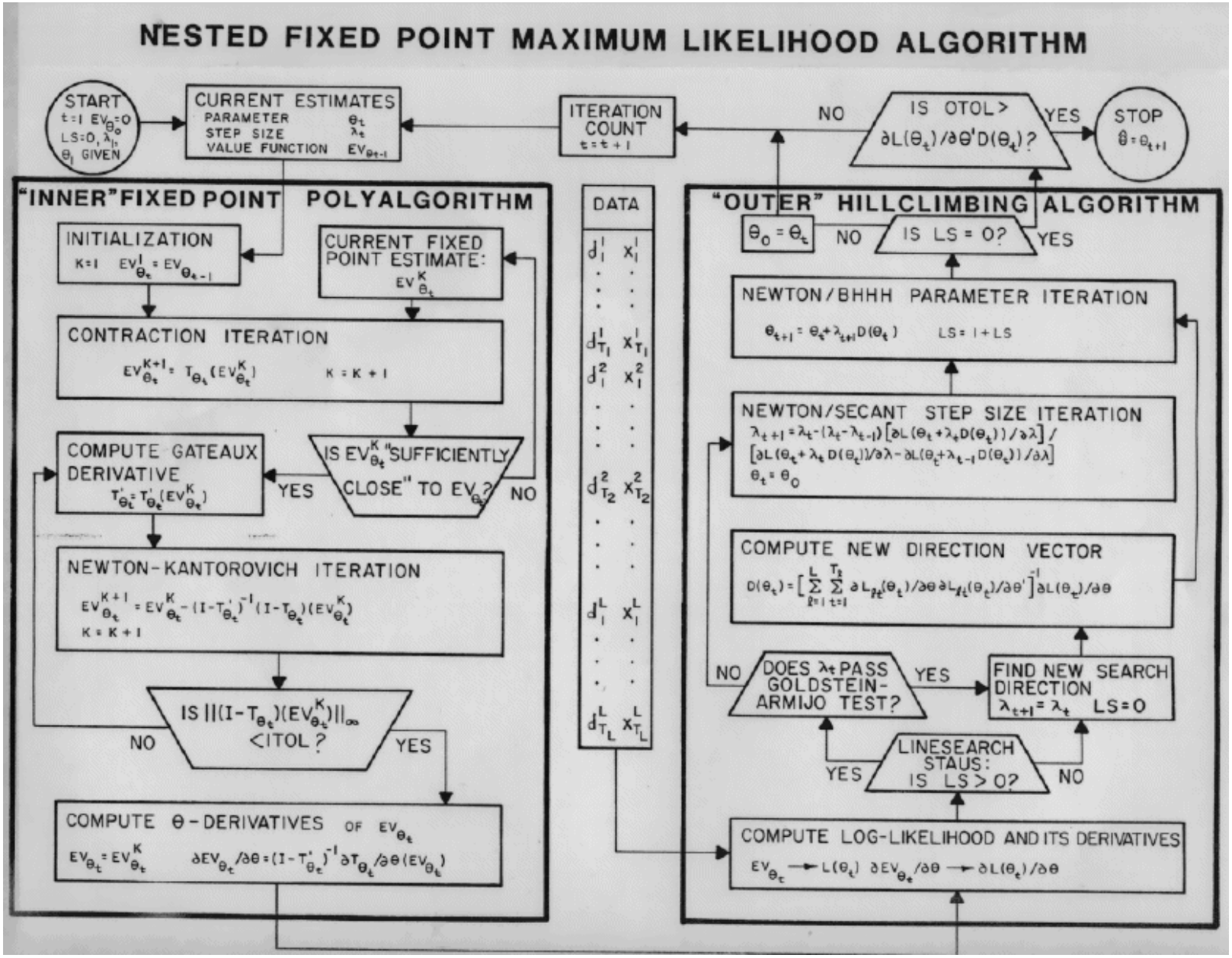
$$\max_{\theta, EV} L(\theta, EV) \text{ subject to } EV = \Gamma_\theta(EV) \tag{5}$$

Since the contraction mapping Γ *always has a unique fixed point, the constraint* $EV = \Gamma_\theta(EV)$ *implies that the fixed point* EV_θ *is an implicit function of* θ*. Thus, the constrained optimization problem (5) reduces to the unconstrained optimization problem*

$$\max_{\theta} L(\theta, EV_\theta) \tag{6}$$

where EV_θ *is the implicit function defined by* $EV_\theta = \Gamma(EV_\theta)$*.*

NFXP pocket guide



STEP 2: Newton-Kantorovich Iterations

- Problem: Find fixed point of the contraction mapping

$$EV = \Gamma(EV)$$

- Error bound on successive contraction iterations:
  $||EV_{k+1} - EV|| \leq \beta||EV_k - EV||$
  linear convergence $\rightarrow$ slow when β close to 1
- Newton-Kantorovich:
  Solve $[I - \Gamma](EV_\theta) = 0$ using Newtons method
  $||EV_{k+1} - EV|| \leq A||EV_k - EV||^2$
  quadratic convergence around fixed point, EV

STEP 2: Newton-Kantorovich Iterations

Newton-Kantorovich iteration:

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

where I is the identity operator on B, and 0 is the zero element of B (i.e. the zero function). The nonlinear operator $I - \Gamma$ has a Fréchet derivative $I - \Gamma'$ which is a bounded linear operator on B with a bounded inverse.

The Fixed Point (poly) Algorithm

1. Successive contraction iterations
   (until EV is in domain of attraction)
2. Newton-Kantorovich (until convergence)

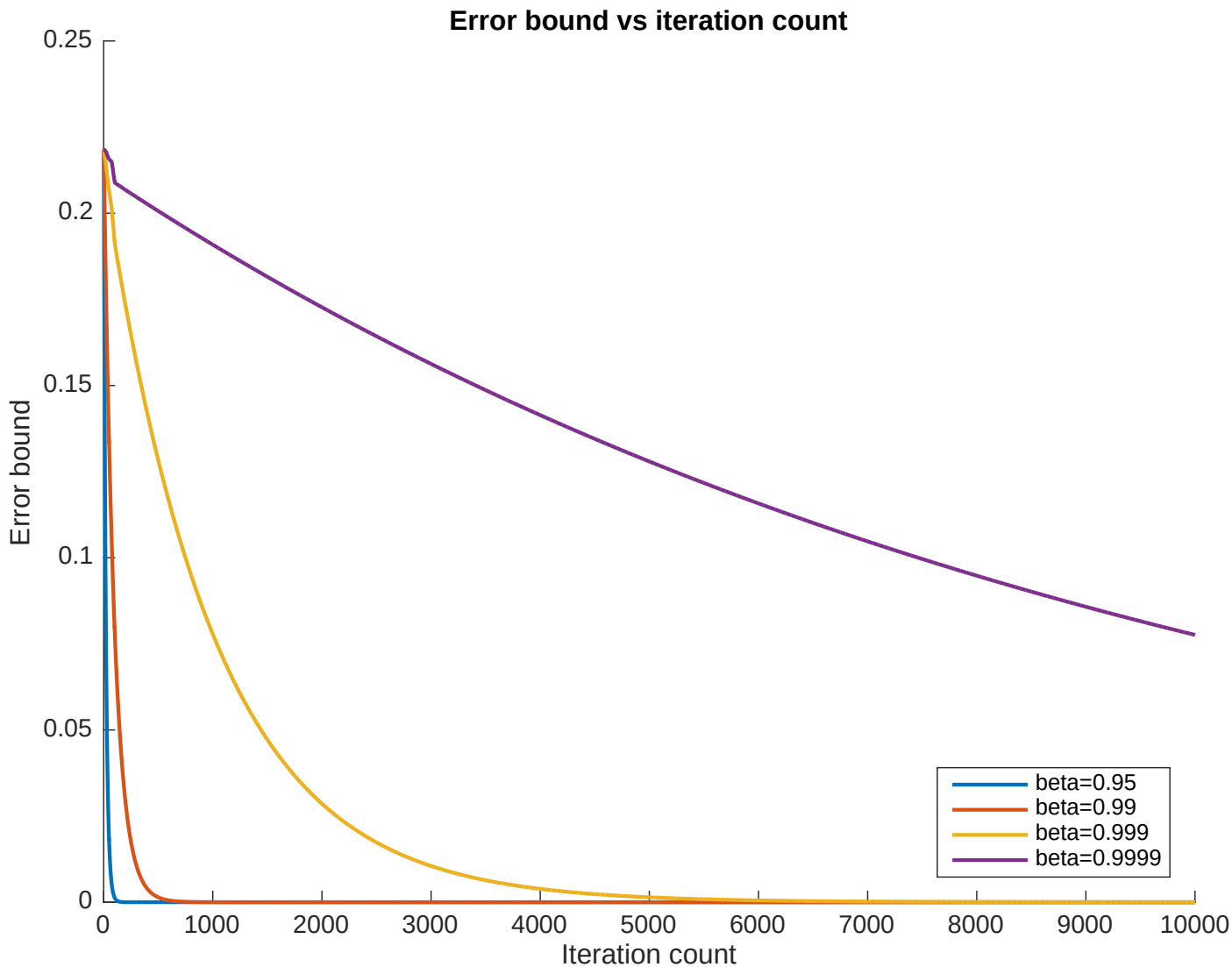
STEP 2: Newton-Kantorovich Iterations, $\beta = 0.9999$

Successive Approximations, VERY Slow

```
Begin contraction iterations
   j           tol         tol(j)/tol(j-1)
   1         0.24310300        0.24310300
   2         0.24307590        0.99988851
   3         0.24304810        0.99988564
   :              :                :
 9998        0.08185935        0.99990000
 9999        0.08185116        0.99990000
 10000       0.08184298        0.99990000
Elapsed time: 1.44752 (seconds)

Begin Newton-Kantorovich iterations
  nwt           tol
   1       9.09494702e-13
Elapsed time: 1.44843 (seconds)

Convergence achieved!
```

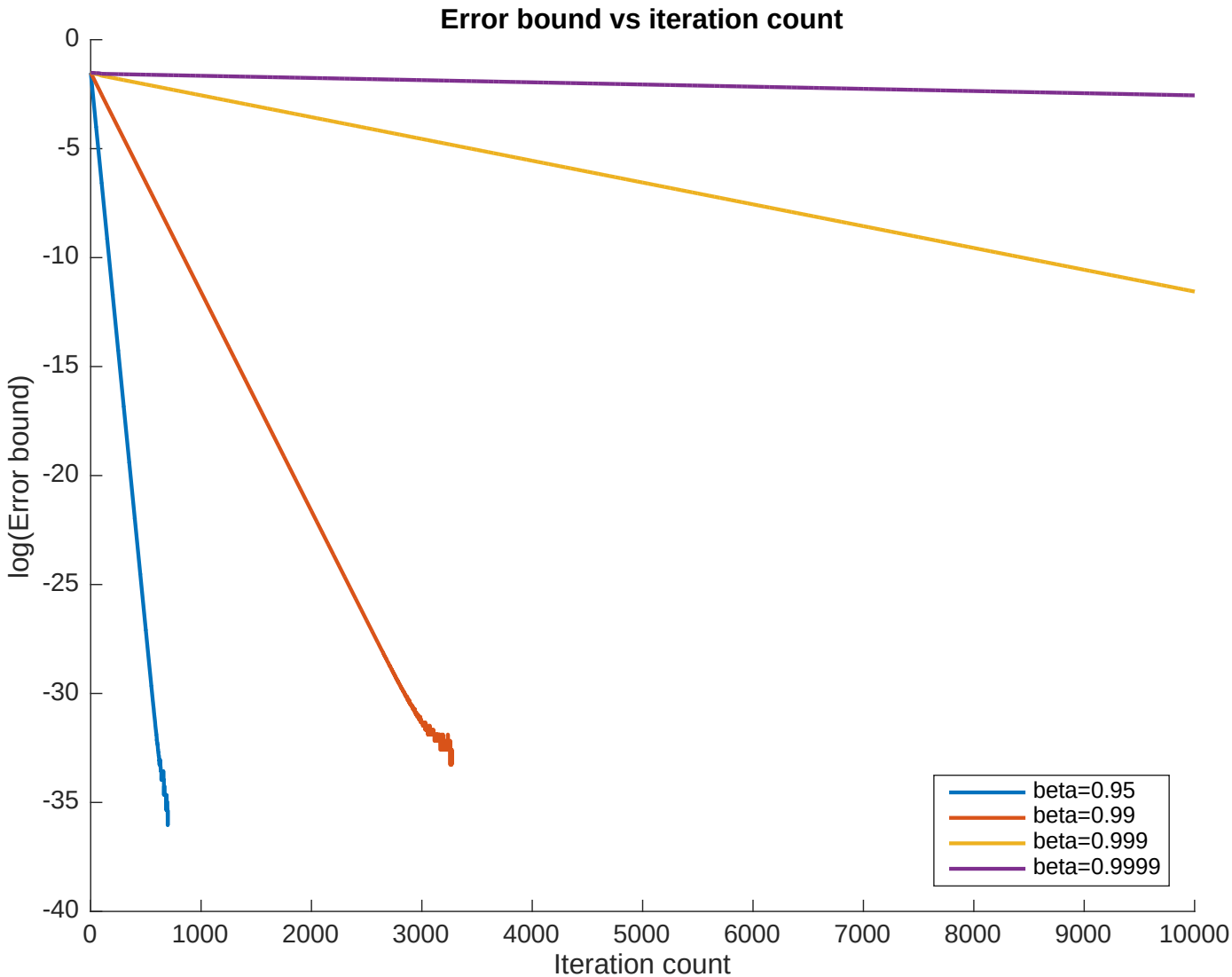
STEP 2: Newton-Kantorovich Iterations

Successive Approximations, VERY Slow



STEP 2: Newton-Kantorovich Iterations

Successive Approximations, Linear convergence



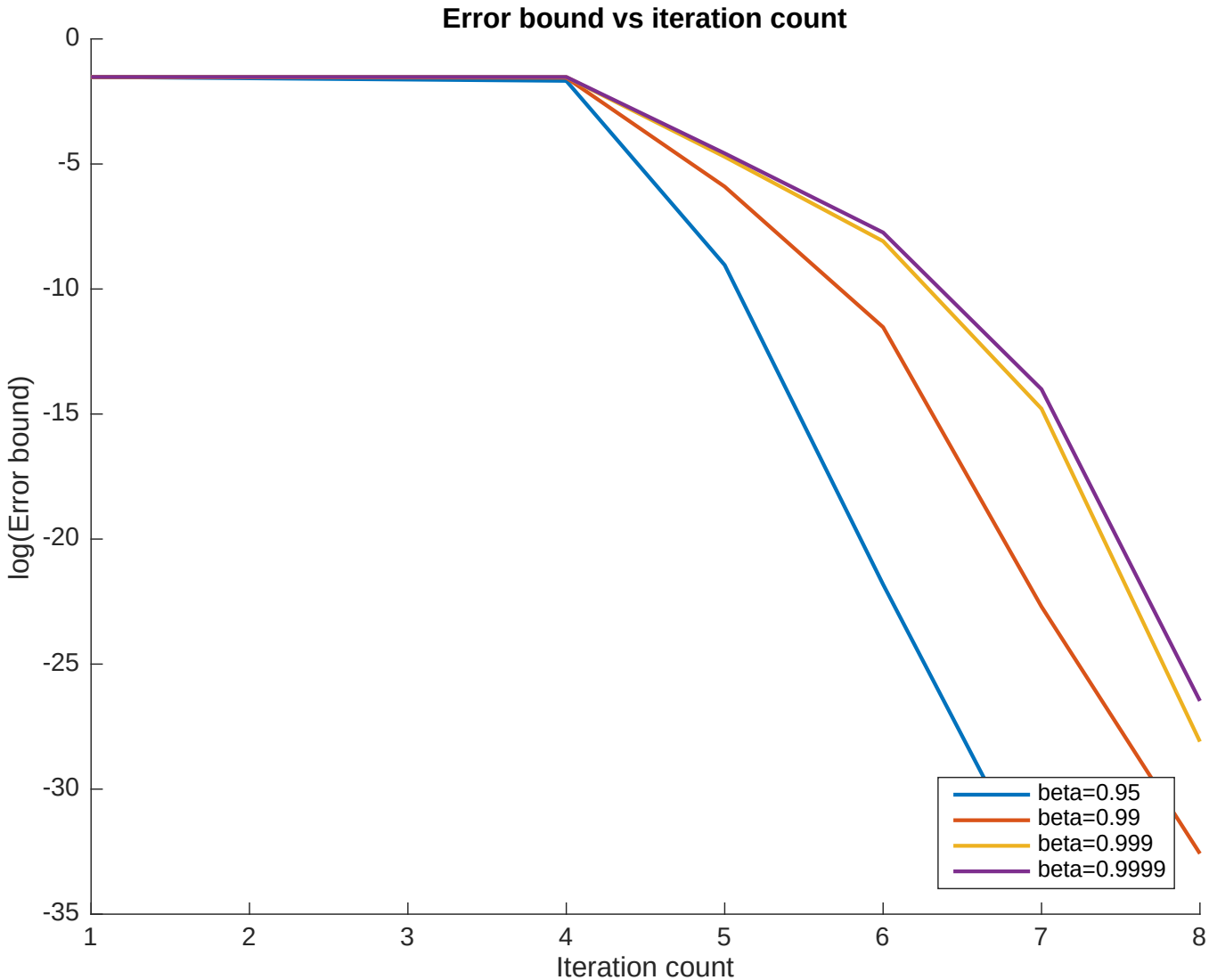
STEP 2: Newton-Kantorovich Iterations, $\beta = 0.9999$

Quadratic convergence!

```
Begin contraction iterations
   j           tol         tol(j)/tol(j-1)
   1         0.21854635        0.21854635
   2         0.21852208        0.99988895
Elapsed time: 0.00056 (seconds)

Begin Newton-Kantorovich iterations
  nwt           tol
   1       1.03744352e-02
   2       4.40564315e-04
   3       8.45941486e-07
   4       3.63797881e-12
Elapsed time: 0.00326 (seconds)

Convergence achieved!
```

STEP 2: Newton-Kantorovich Iterations

NR: Quadratic convergence!



STEP 2: When to switch to Newton-Kantorovich

Observations:

- $tol_k = \|EV_{k+1} - EV_k\| < \beta\|EV_k - EV\|$
- tol_k quickly slow down and declines very slowly for β close to 1
- Relative tolerance tol_{k+1}/tol_k approach β

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

STEP 3: Recenter to ensure numerical stability

Logit formulas must be reentered.

$$\begin{aligned} P_i &= \frac{\exp(V_i)}{\sum_{j\in D(y)} \exp(V_j)} \\ &= \frac{\exp(V_i - V_0)}{\sum_{j\in D(y)} \exp(V_j - V_0)} \end{aligned}$$

and "log-sum" must be recenteret too

$$\begin{aligned} EV_\theta &= \int_y \ln \sum_{j'\in D(y)} \exp(V_j) p(dy|x,d,\theta_2) \\ &= \int_y \left(V_0 + \ln \sum_{j'\in D(y)} \exp(V_j - V_0) \right) p(dy|x,d,\theta_2) \end{aligned}$$

If V_0 is chosen to be $V_0 = \max_j V_j$ we can avoid numerical instability due to overflow/underflow

STEP 3: MATLAB implementation of Bellman Operator

```
% Bellman operator
function [ev1, pk]=bellman(ev, P, c, mp)
   VK=-c+mp.beta*ev;  % Value off keep
   VR=-mp.RC-c(1)+mp.beta*ev(1);  % Value of replacing

   % Recenter by Bellman by subtracting max(VK, VR)
   maxV=max(VK, VR);
   ev1=P*(maxV + log(exp(VK-maxV)  +  exp(VR-maxV)));

   if nargout>1 % Choice probability
    pk=1./(1+exp((VR-VK)));
   end
end
```

STEP 4: Fréchet derivative of Bellman operator

Fréchet derivative

- For NK iteration we need Γ'

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

- In terms of its finite-dimensional approximation, Γ'_θ takes the form of an $N \times N$ matrix equal to the partial derivatives of the $N \times 1$ vector $\Gamma_\theta(EV_\theta)$ with respect to the $N \times 1$ vector EV_θ
- Γ'_θ is simply β times the transition probability matrix for the controlled process $\{d_t, x_t\}$
- Two lines of code in MATLAB

STEP 4: MATLAB implementation of Fréchet derivative

```
% Frechet derivative of Bellman operator
function dGamma_dEv=dbellman(pk,  P, mp)
    dGamma_dEv=mp.beta*bsxfun(@times, P, pk');

    % Add additional term for dGamma/dEv(1),
    % since Ev(1) enter logsum for all states
    dGamma_dEv(:,1)=dGamma_dEv(:,1)+mp.beta*P*(1-pk);
end % end of NFXP.dbellman
```

STEP 5: Provide analytical gradients of likelihood

Gradient similar to the gradient for the conventional logit

$$\partial \ell_i^1(\theta)/\partial\theta = [d_{it} - P(d_{it}|x_{it},\theta)] \times \partial(v_{repl.} - v_{keep})/\partial\theta$$

- Only thing that differs is the inner derivative of the choice specific value function that besides derivatives of current utility also includes $\partial EV_\theta/\partial\theta$ wrt. θ
- By the implicit function theorem we obtain

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

- By-product of the N-K algorithm: $[I - \Gamma'_\theta]^{-1}$

STEP 5: MATLAB implementation of scores

```
1
2  % step 1: compute derivative of contraction operator wrt. parameters
3  dtdmp(:, 1)=P*pk-1; % Derivative wrt RC
4  dtdmp(:, 2)=-(P*dc).*pk; % Derivative wrt c
5
6  % step 2: compute derivative of ev wrt. parameters
7  devdmp=F\dtdmp;  % F=I-Gamma' is by-product of NK-iteration
8
9  % step 3: compute derivative of log-likelihood wrt. parameters
10 score=bsxfun(@times, (lp- 1 + data.d), ...
11 [-ones(N,1) dc(data.x,:)]  + (devdmp(ones(N,1),:)-devdmp(data.x,:)) ..
12 );
```

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 *outer product of scores* as approx. to Hessian

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

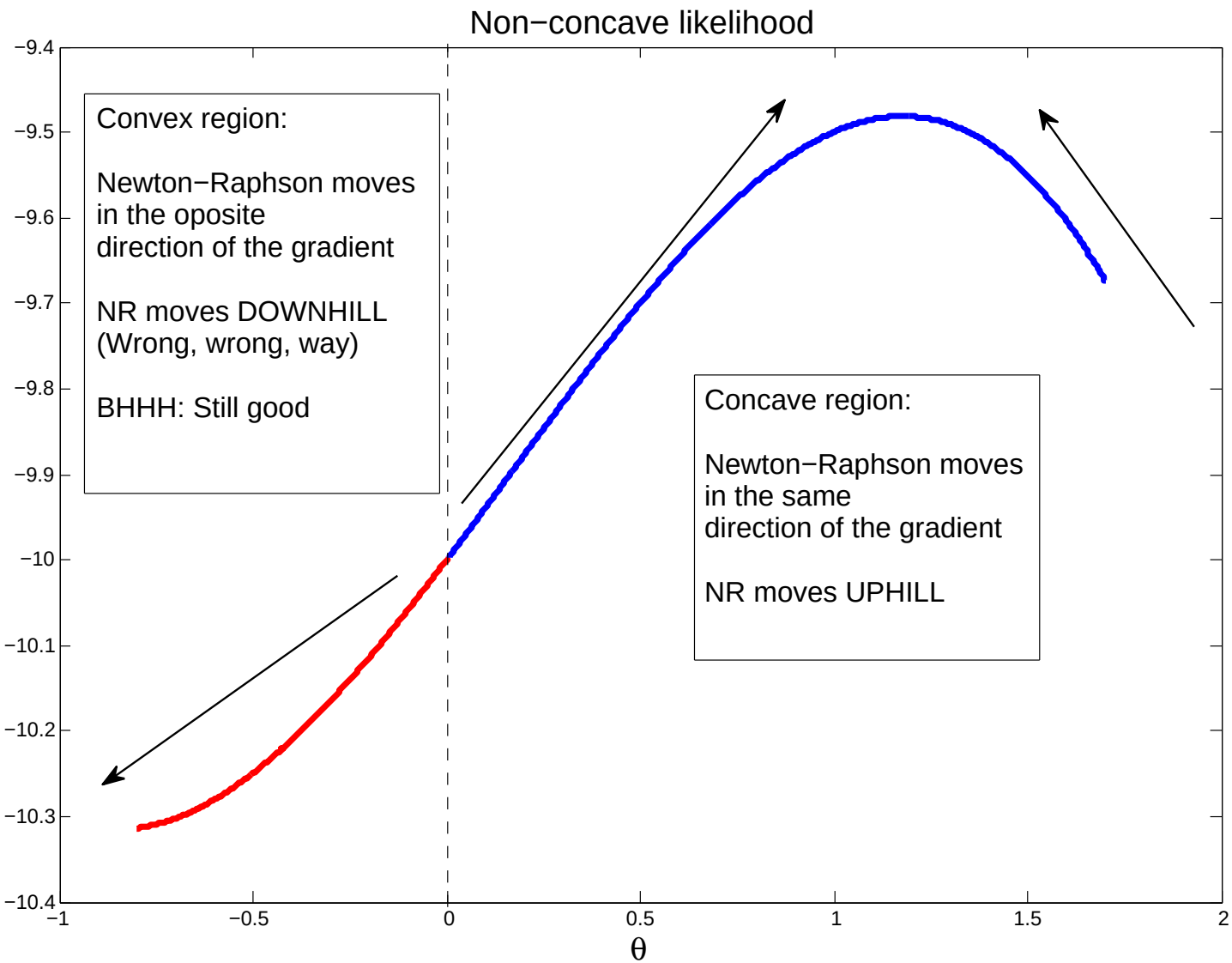
- Why is this valid? Information identity:

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

  (only valid for MLE and CMLE)

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

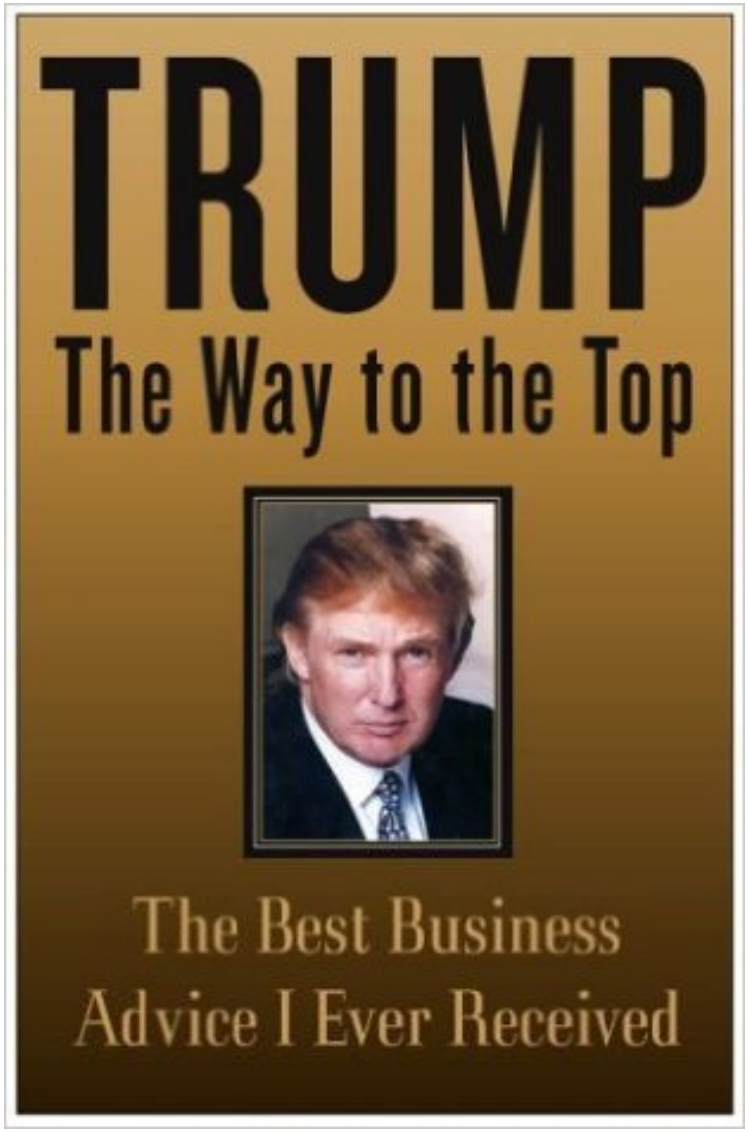
- Only a good approximation
  - At the true parameters
  - for large N
  - for well specified models (in principle only valid for MLE)
- Only superlinear convergent - not quadratic

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.

STEP 6: BHHH

"The road ahead will be long. Our climb will be steep. We may not get there in one year or even in one term. But, America, I have never been more hopeful than I am tonight that we will get there. I promise you, we as a people will get there." (Barack Obama, Nov. 2008)

STEP 6: Ooups, new sheriff in town



Convergence!

β=0.9999

```
1  ---------------------------------------------------------------------
2  ***                    Convergence Achieved
   ***
3  ---------------------------------------------------------------------
4                                 _
5                                \`\
6                                |= |
7                               /-  ;.---.
8                          _ __.'      (_____)
9                           `          (______)
10                         _'   ._ .'  (_____)
11                          `          (____)
12                         --`'------'`
13 Number of iterations: 9
14 grad*direc             0.00003
15 Log-likelihood     -276.74524
16
17     Param.              Estimates          s.e.         t-stat
18 ---------------------------------------------------------------------
19     RC                    11.1525        0.9167        12.1655
20     c                      2.3298        0.3288         7.0856
21 ---------------------------------------------------------------------
```

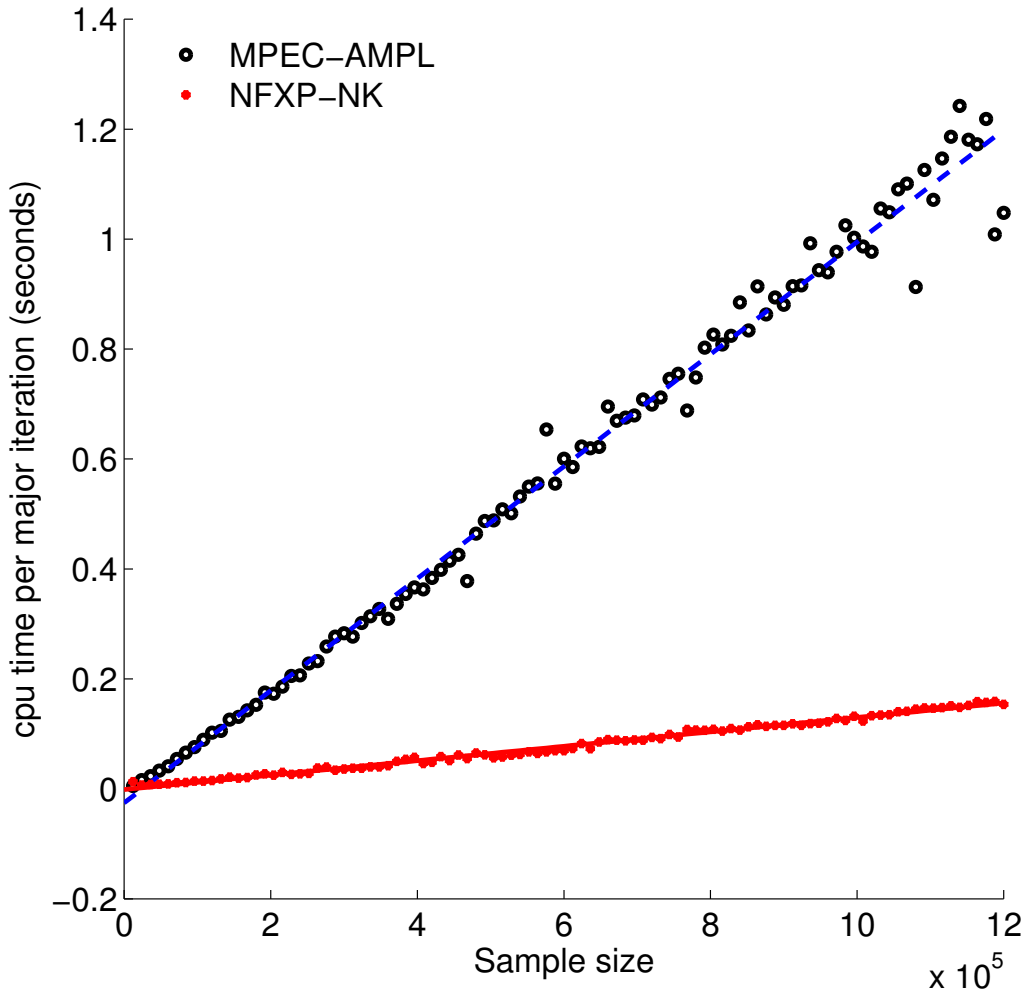
MPEC versus NFXP-NK: sample size 6,000

β	Converged (out of 1250)	CPU Time (in sec.)	# of Major Iter.	# of Func. Eval.	# of Bellm. Iter.	# of N-K Iter.
			MPEC-Matlab			
0.975	1247	1.677	60.9	69.9		
0.985	1249	1.648	62.9	70.1		
0.995	1249	1.783	67.4	74.0		
0.999	1249	1.849	72.2	78.4		
0.9995	1250	1.967	74.8	81.5		
0.9999	1248	2.117	79.7	87.5		
			MPEC-AMPL			
0.975	1246	0.054	9.3	12.1		
0.985	1217	0.078	16.1	44.1		
0.995	1206	0.080	17.4	49.3		
0.999	1248	0.055	9.9	12.6		
0.9995	1250	0.056	9.9	11.2		
0.9999	1249	0.060	11.1	13.1		
			NFXP-NK			
0.975	1250	0.068	11.4	13.9	155.7	51.3
0.985	1250	0.066	10.5	12.9	146.7	50.9
0.995	1250	0.069	9.9	12.6	145.5	55.1
0.999	1250	0.069	9.4	12.5	141.9	57.1
0.9995	1250	0.078	9.4	12.5	142.6	57.5
0.9999	1250	0.070	9.4	12.6	142.4	57.7

MPEC versus NFXP-NK: sample size 60,000

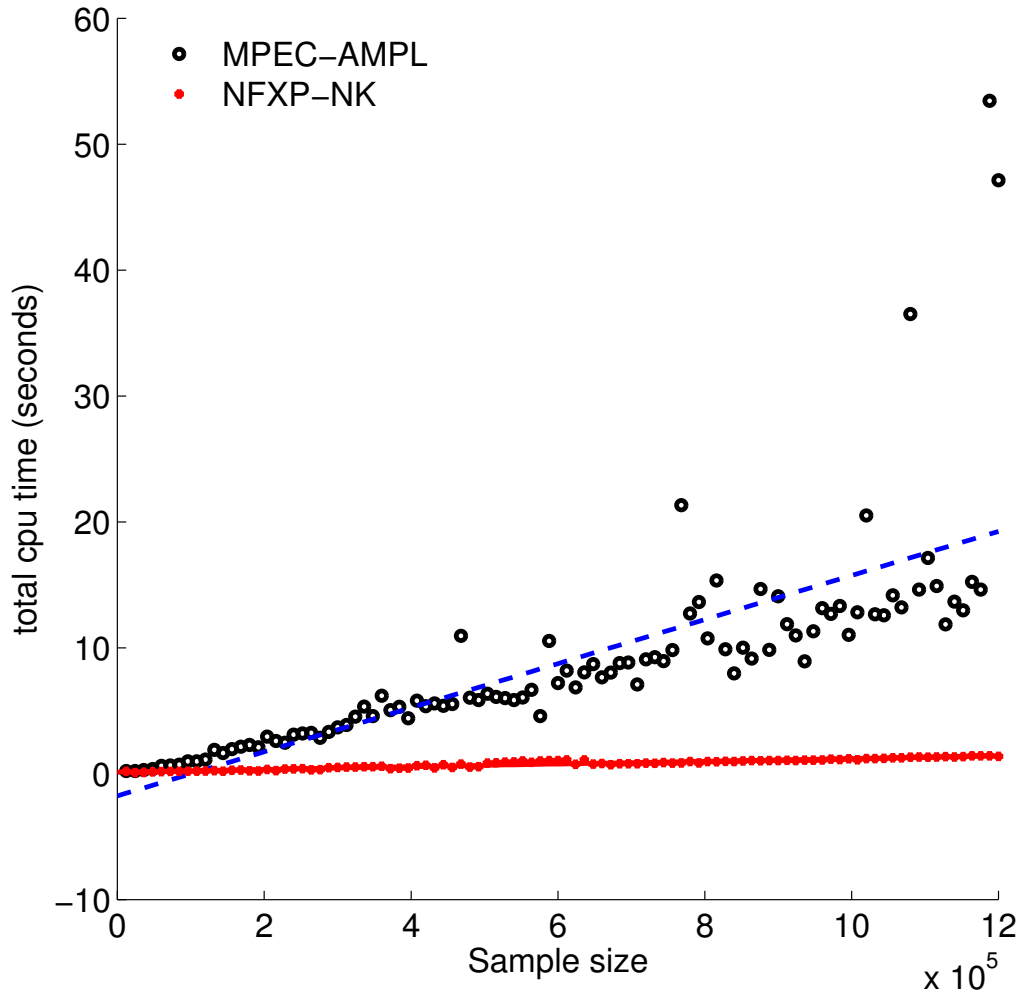
β	Converged (out of 1250)	CPU Time (in sec.)	# of Major Iter.	# of Func. Eval.	# of Bellm. Iter.	# of N-K Iter.
			MPEC-AMPL			
0.975	1247	0.53	9.2	11.7		
0.985	1226	0.76	13.9	32.6		
0.995	1219	0.74	14.2	30.7		
0.999	1249	0.56	9.5	11.1		
0.9995	1250	0.59	9.9	11.2		
0.9999	1250	0.63	11.0	12.7		
			NFXP-NK			
0.975	1250	0.15	8.2	11.3	113.7	43.7
0.985	1250	0.16	8.4	11.4	124.1	46.2
0.995	1250	0.16	9.4	12.1	133.6	52.7
0.999	1250	0.17	9.5	12.2	133.6	55.2
0.9995	1250	0.17	9.5	12.2	132.3	55.2
0.9999	1250	0.17	9.5	12.2	131.7	55.4

CPU time is linear sample size



$T_{NFXP} = 0.001 + 0.13x$ $(R^2 = 0.991)$, $T_{MPEC} = -0.025 + 1.02x$ $(R^2 = 0.988)$.

CPU time is linear sample size



$$T_{NFXP} = 0.129 + 1.07x \ (R^2 = 0.926) \ , \ T_{MPEC} = -1.760 + 17.51x \ (R^2 = 0.554).$$

Summary of findings

Su and Judd (Econometrica, 2012) used an inefficient version of NFXP

- that solely relies on the method of successive approximations to solve the fixed point problem.

Using the efficient version of NFXP proposed by Rust (1987) we find:

- MPEC and NFXP-NK are similar in performance when the sample size is relatively small.
- In problems with large sample sizes, NFXP-NK outperforms MPEC by a significant margin.
- NFXP does not slow down as $\beta \rightarrow 1$
- It is non-trivial to compute standard error using MPEC, whereas they are a natural by-product of NFXP.

Desirable features of MPEC

- Ease of use by people who are not interested in devoting time to the special-purpose programming necessary to implement NFXP-NK.
- Can easily be implemented in the intuitive AMPL language.

MPEC does not seem appropriate when estimating life cycle models.

PART III

Sequential Estimation: Nested Pseudo Likelihood and CCP estimator

Bertel Schjerning

Main Contributions of Aguirregabiria and Mira (2002)

Nested Pseudo Likelihood (NPL) algorithm

- Solution of the DP problem in choice probability space (rather than value functions space)

Statistical and computational properties of the estimator.

- When NPL is initialized with consistent nonparametric estimates of conditional choice probabilities, successive iterations return a sequence of estimators of the structural parameters which we call K-stage policy iteration estimators.
- The sequence includes as extreme cases a Hotz-Miller estimator (for K = 1) and Rust's nested fixed point MLE estimator (in the limit when $K \to \infty$).

Monte Carlo experiments

- Monte Carlo based on Rust's bus replacement model.
- Reveal a trade-off between finite sample precision and computational cost in the sequence of policy iteration estimators.

The General Problem

Bellman equation

$$V(s;\theta) = \max_{a \in \mathcal{A}(s)} \{u(s,a;\theta_u) + \beta \int V(s';\theta) p(s'|s,a;\theta_g,\theta_f) ds'\}$$

u and p: known up to a set of parameters, $\theta = (\theta_u, \theta_g, \theta_f)$

- **The agent's problem:** Maximize expected sum of current and future discounted utilities
  - a : Discrete control variable, $a \in \mathcal{A}(s) = \{1, 2, .., J\}$.
  - s : Current state, fully observed by agent
  - $s\prime$: Future state; possibly continuous and subject to uncertainty
- **The agents beliefs about s′:**
  - Obeys a (controlled) Markov transition probability $p(s_{t+1}|s_t, a_t; \theta_g, \theta_f)$
- **Model solution,** $V(s;\theta)$
  - Find the fixed point for the Bellman equation

A&M maintain Rust's Assumptions

Assumption (Conditional Independence (CI))

State variables, $s_t = (x_t, \varepsilon_t)$ obeys a (conditional independent) controlled Markov process with probability density

$$p(x_{t+1}, \varepsilon_{t+1} | x_t, \varepsilon_t, a, \theta_g, \theta_f) = g(\varepsilon_{t+1} | x_{t+1}, \theta_g) f(x_{t+1} | x_t, a, \theta_f)$$

Assumption (Additive Separability (AS))

$$U(s_t, a) = u(x_t, a; \theta_u) + \varepsilon_t(a)$$

Assumption (Finite Domain of Observable State Variables)

$$x \in X = \{x^1, x^1, ..., x^m\}$$

Assumption (XV)

The unobserved state variables, ε_t are assumed to be multivariate iid. extreme value distributed

Bellman equation and choice probabilities

Define the smoothed value function $V_\sigma(x) = \int V(x,\epsilon)g(\epsilon|x)d\epsilon$ where σ represents parameters that index the distribution of the $\epsilon's$.
($\sigma = \theta_2$ in Rust notation)

Under assumptions CI, AS and finite domain of x, we can summarize the solution by the smoothed Bellman operator, $\Gamma_\sigma(V_\sigma)$

$$V_\sigma(x) = \int \max_{a\in\mathcal{A}(x)} \left\{ u(x,a) + \epsilon(a) + \beta \sum_{x'} V_\sigma(x') f(x'|x,a) \right\} g(\epsilon|x)d\epsilon$$

The conditional choice probability (CCP)

$$P(a|x) = \int I\{a = \arg\max_{j\in\mathcal{A}(x)} \{v(j,x) + \epsilon(j)\}\, g(\epsilon|x)d\epsilon$$

where $v(x,a) = u(x,a) + \beta \sum_{x'} V_\sigma(x') f(x'|x,a)$ is the choice-specific value function

From Conditional Choice Probabilities to Value Functions

- $P(a|x)$ is uniquely determined by the vector of normalized value function differences $\tilde{v}(x,a) = v(x,a) - v(x,1)$
- That is, there exists a vector mapping $Q_x(.)$ such that $\{P(a|x) : a > 1\} = Q_x(\tilde{v}(x,a) : a > 1)$, where, without loss of generality, we exclude the probability of alternative one.
- In general

$$Q_x^j(\tilde{v}(x,a) : a > 1) = \partial S([0, \{\tilde{v}(x,a) : a > 1\}], x)/\partial \tilde{v}(x,j)$$

where

$$S(\{v(x,a) : a \in A\}, x) = \int \max[v(x,a) + \epsilon(a)]g(d\epsilon|x)$$

is McFadden's social surplus function.

From Conditional Choice Probabilities to Value Functions

- Under assumption (XV), social surplus function is the well known "log-sum" formula

$$S(\{v(x,a) : a \in A\}, x) = \int \max_{a \in A}[v(x,a) + \epsilon(a)]g(d\epsilon|x)$$
$$= \sigma \log \sum_{j \in A} exp(v(x,j)/\sigma)$$

  the j'th component Q_x takes the well known logistic form

$$Q_x^j(\tilde{v}(x,a)) = \frac{\exp(\tilde{v}(x,a)/\sigma)}{1 + \sum_{j=2}^{A} \exp(\tilde{v}(x,j)/\sigma)}$$

- NOTE, it's not hard to invert Q_x^j in this case

From Conditional Choice Probabilities to Value Functions

The Smooth Bellman equation can be re-written as

$$V_\sigma(x) = \sum_{a \in A} P(a|x) \left\{ u(x,a) + E[\epsilon(a)|x,a] + \beta \sum_{x'} V_\sigma(x') f(x'|x,a) \right\}$$

where $E[\epsilon(a)|x,a]$ is the expectation of the unobservable ϵ conditional on the optimal choice of alternative a:

$$E[\epsilon(a)|x,a] = [P(a|x)]^{-1} \int \epsilon(a) I\{\tilde{v}(x,a) + \epsilon(a) > \tilde{v}(x,k) + \epsilon(j), j \in A(x)\} g(d\epsilon|x)$$

$E[\epsilon(a)|x,a]$ clearly depends on $\tilde{v}(x,a)$, but due to the invertibility of Q_x we can express it probability space

$$E[\epsilon(a)|x,a] = e_x(a, \{P(j|x)\}_{j \in A}).$$

Under XV we have $E[\epsilon(a)|x,a] = \gamma + \ln P(a|x)$ where $\gamma = 0.5772156649...$ is Euler's constant

From Conditional Choice Probabilities to Value Functions

In compact matrix notation we can write

$$V_\sigma = \sum_{a \in A} P(a) * \{u(a) + e(a, P) + \beta F(a) V_\sigma\}$$

where $*$ is the element by element product and $P(a)$, $u(a)$, $e(a, P)$ and V_σ are all $M \times 1$ vectors and $F(a)$ is the $M \times M$ matrix of conditional transition probabilities $f(x'|x, a)$

This system of fixed point equations can be solved for the value function to obtain V_σ as a function of P:

$$V_\sigma = \psi(P) = [I - \beta F^U(P)]^{-1} \sum_{a \in A} \{P(a) * [u(a) + e(a, P)]\}$$

where $F^U(P) = \sum_{a \in A} P(a) F(a)$ is the $M \times M$ matrix of unconditional transition probabilities induced by P.

The Fixed Point Problem in Probability Space

Recall that

$$V_\sigma = \psi(P) = [I - \beta F^U(P)]^{-1} \sum_{a \in A} \{P(a) * [u(a) + e(a, P)]\} \qquad (7)$$

and

$$P(a|x) = \int I\{a = \arg\max_{j \in \mathcal{A}(x)} \{v(j, x) + \epsilon(j)\}\, g(\epsilon|x) d\epsilon \qquad (8)$$

where $v(x, a) = u(x, a) + \beta \sum_{x'} V_\sigma(x') f(x'|x, a)$

The policy iteration operator, Ψ

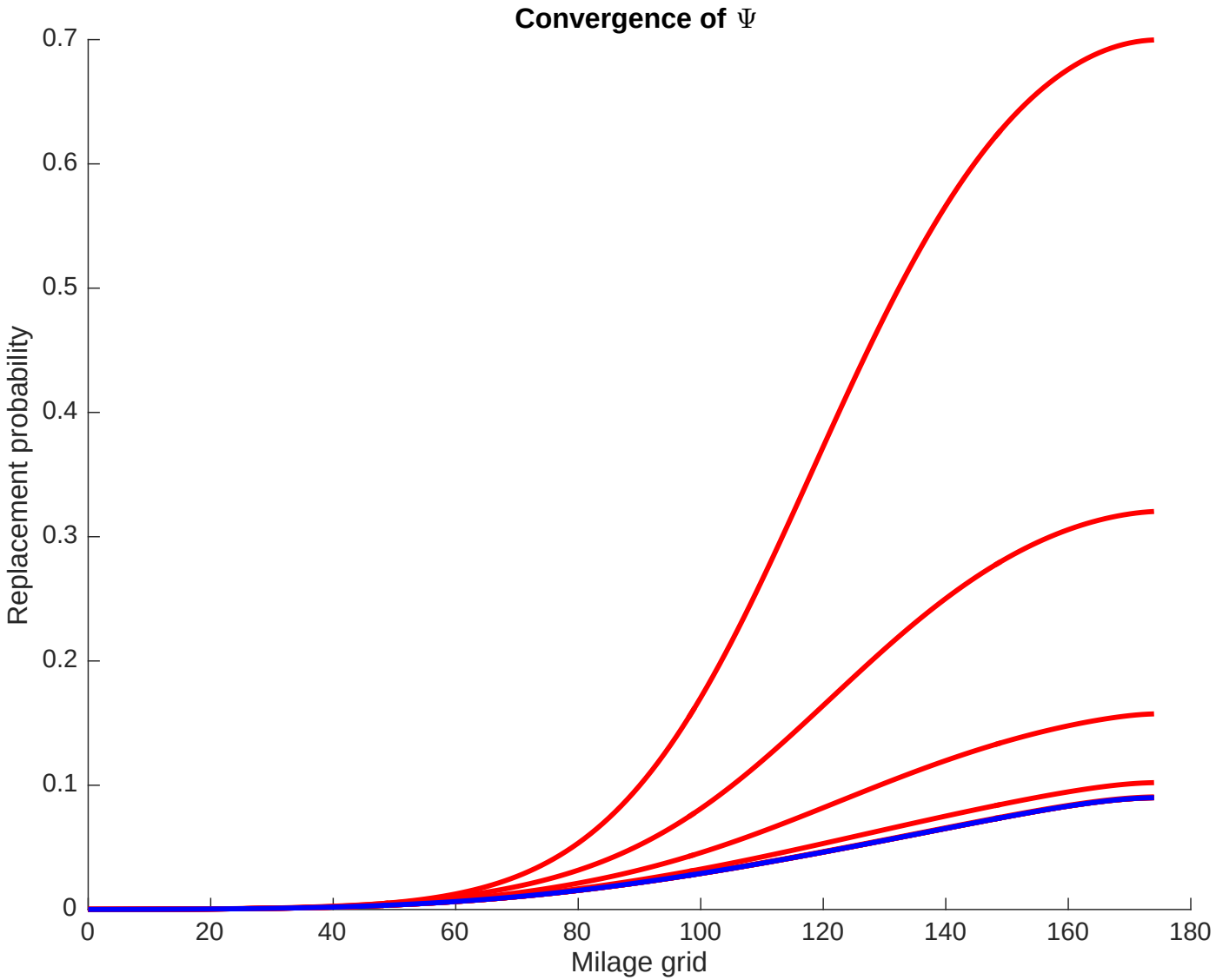
Substituting the *policy valuation operator*, $\psi(P)$ defined by (7) into the formula for CCP's in (8) we obtain the cornerstone of NPL algorithm algorithm:

$$P = \Psi(P) = \Lambda(\psi(P))$$

- Hence, the optimal choice probabilities P is a fixed point of Ψ.
- Thus the original fixed point problem in "value space" can be reformulated as a fixed point problem in "probability space"

Finding fixed point, $P = \Psi(P)$

Fast convergence of successive approximations, $P_{k+1} = \Psi(P_k)$



Likelihood function

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

Likelihood function

$$\ell_i^f(\theta) = \ell_i^1(\theta) + \ell_i^2(\theta_f) = \sum_{t=2}^{T_i} log(P_\theta(a_{i,t}|x_{i,t})) + \sum_{t=2}^{T_i} log\left(f_{\theta_f}(x_{i,t}|x_{i,t-1}, a_{i,t-1})\right)$$

Two Step-Estimator

- Consistent estimates of the conditional transition probability parameters θ_f can be obtained from transition data without having to solve the Markov decision model.
- We focus on the estimation of $\alpha = (\theta_u, \theta_g)$ given initial consistent estimates of θ_f obtained from maximizing the partial log-likelihood $\ell^2(\theta_f) = \sum_i \ell_i^2(\theta_f)$
- Originally suggested in Rust(1987)

Nested Pseudo Likelihood Algorithm

Initialization

- Let $\hat{\theta}_f$ be an estimate of θ_f .
- Start with an initial guess for the conditional choice probabilities, $P^0 \in [0,1]^{MJ}$.

At iteration $K \geq 1$, apply the following steps:

- Step 1: Obtain a new pseudo-likelihood estimate of α, α^K, as

$$\alpha^K = \arg\max_{\alpha \in \Theta} \sum_{i=1}^{n} \ln \Psi_{\alpha,\hat{\theta}_f}(P^{K-1})(a_i|x_i) \tag{9}$$

  where $\Psi_\theta(P)(a|x)$ is the (a,x)'s element of $\Psi_\theta(P)$.
- Step 2: Update P using the arg max from step 1, i.e.

$$P^K = \Psi_{(\alpha^K,\hat{\theta}_f)}(P^{K-1}) \tag{10}$$

- Iterate in K until convergence in P (and α) is reached.

Sequential Policy Iteration Estimators

- Performing one, two, and in general K iterations of the NPL algorithm yields a sequence $\{\hat{\alpha}_1, \hat{\alpha}_1, ..., \hat{\alpha}_K\}$ of statistics that can be used as estimators of the true value of α, α^*
- A&M call them sequential policy iteration (PI) estimators.

The K-stage PI estimator is defined as:

$$\hat{\alpha}^K = \arg\max_{\alpha \in \Theta} \sum_{i=1}^{n} \ln \Psi(P^{K-1})(a_i|x_i)$$

where $P^K = \Psi_{(\hat{\alpha}^K, \hat{\theta}_f)}(P^{K-1})$ and P^0 is a consistent, non-parametric estimate of the true conditional choice probabilities

Statistical properties of K-PI estimator

For any K

- $\hat{\alpha}^K$ is asymptotically equivalent to MLE
- $\hat{\alpha}^K$ is $\sqrt{n}$ consistent
- $\hat{\alpha}^K$ is asymptotic normal with known variance-covariance matrix (A&M has an expression that accounts for first step estimation error)

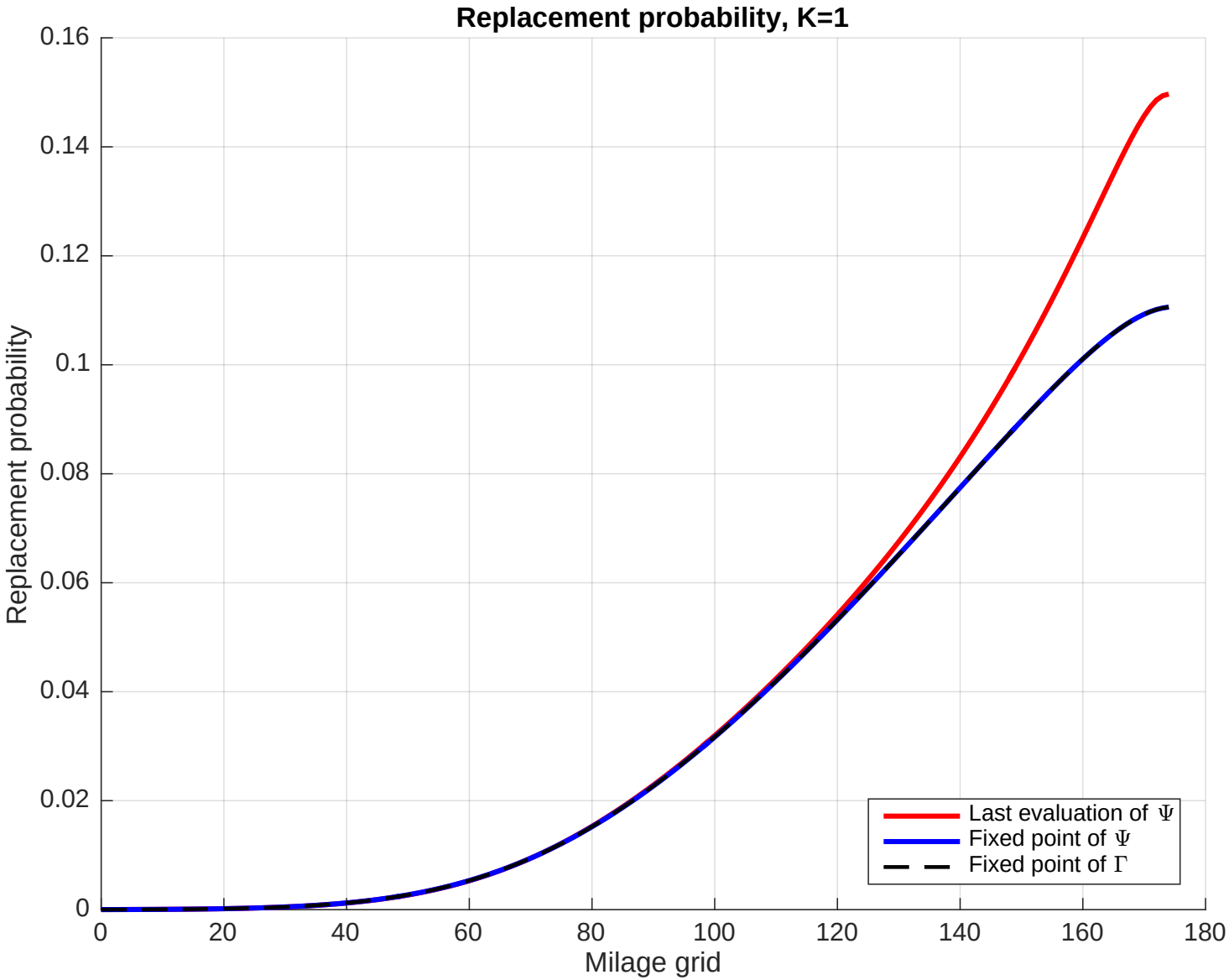
For $K = 1$

- $\hat{\alpha}^K$ encompasses Hotz-Miller (1993) estimator

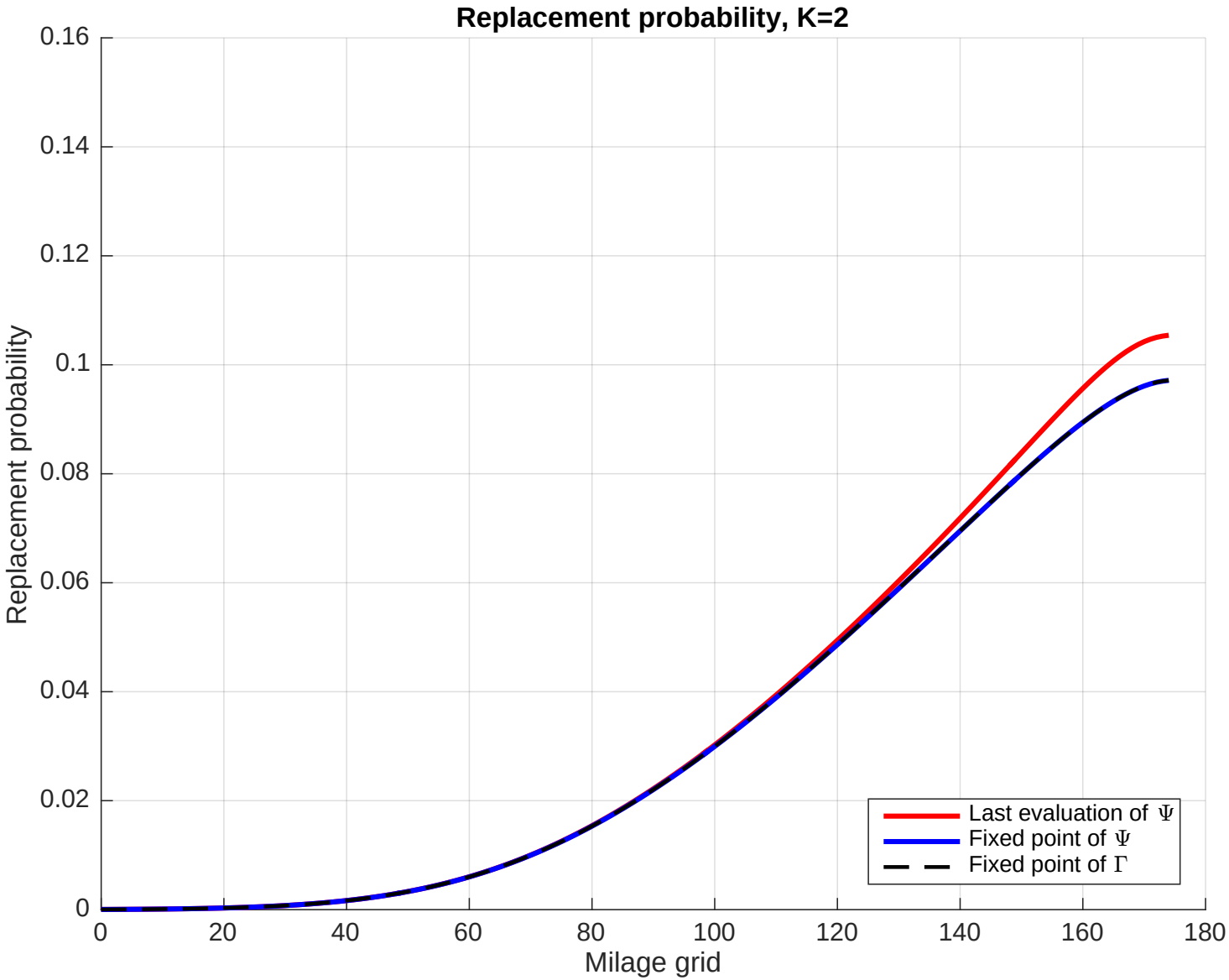
As $K \to \infty$

- $\hat{\alpha}^K$ converges to the MLE estimator obtained by NFXP
- Standard inference.

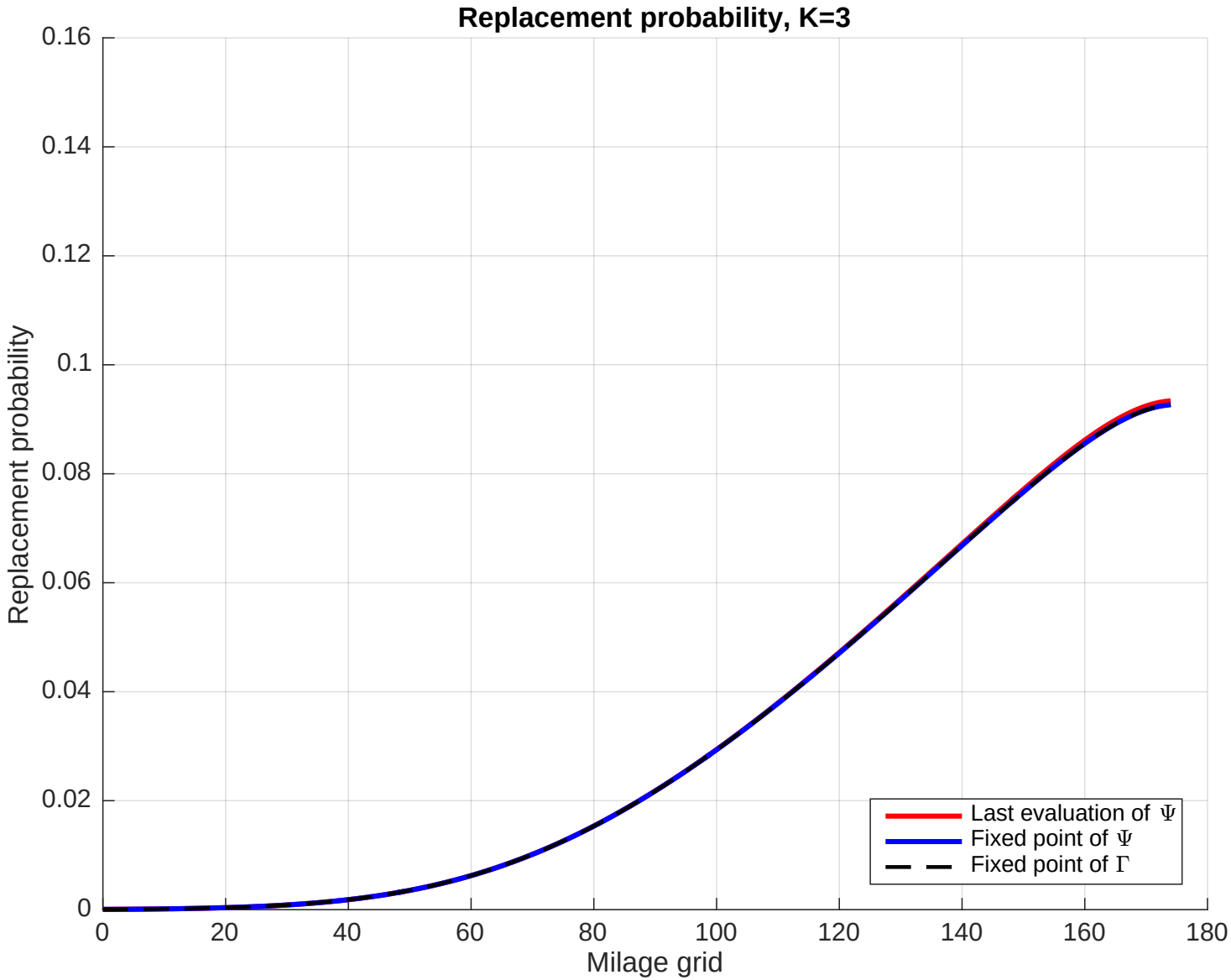
Replacement probability, K=1



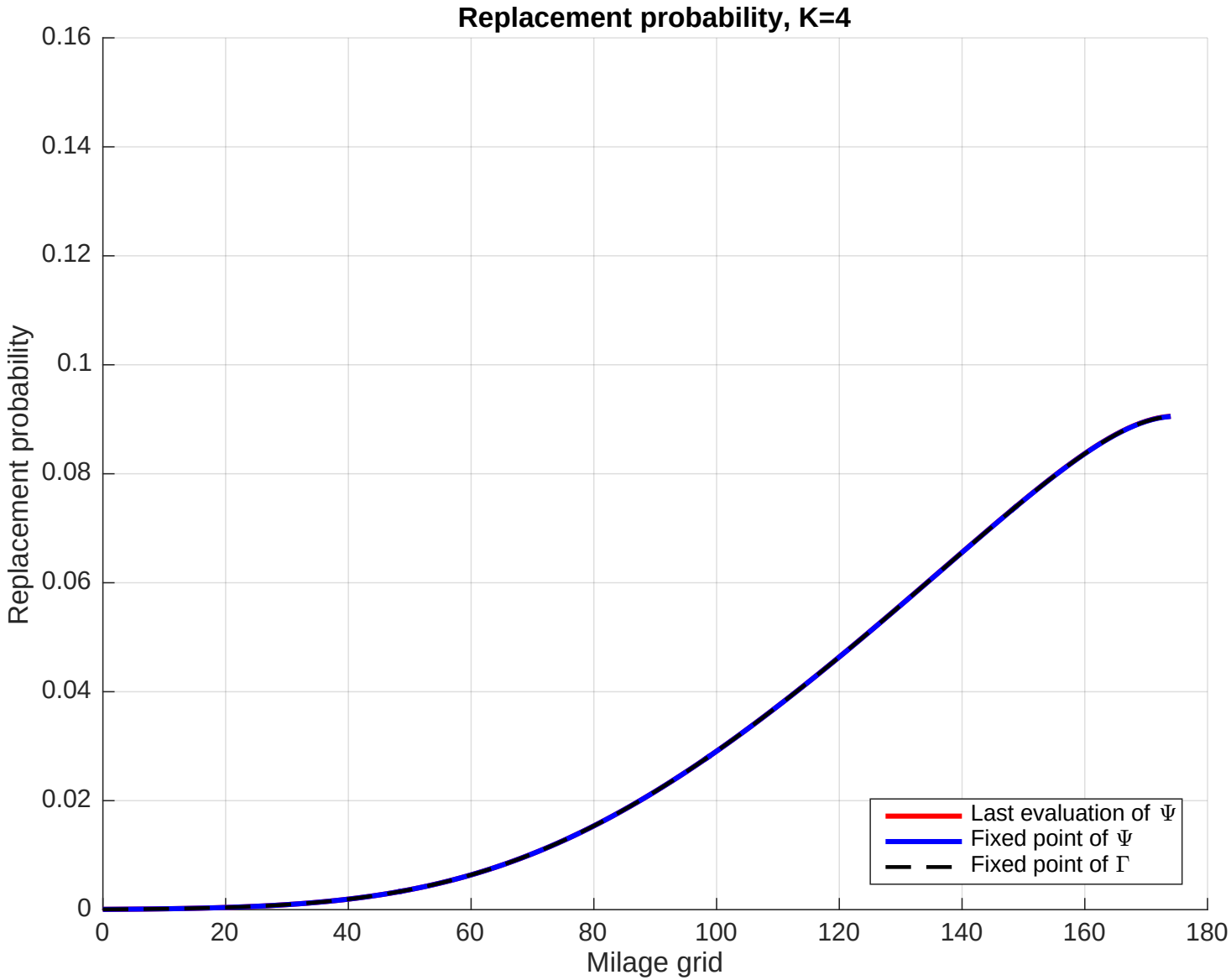
Replacement probability, K=2



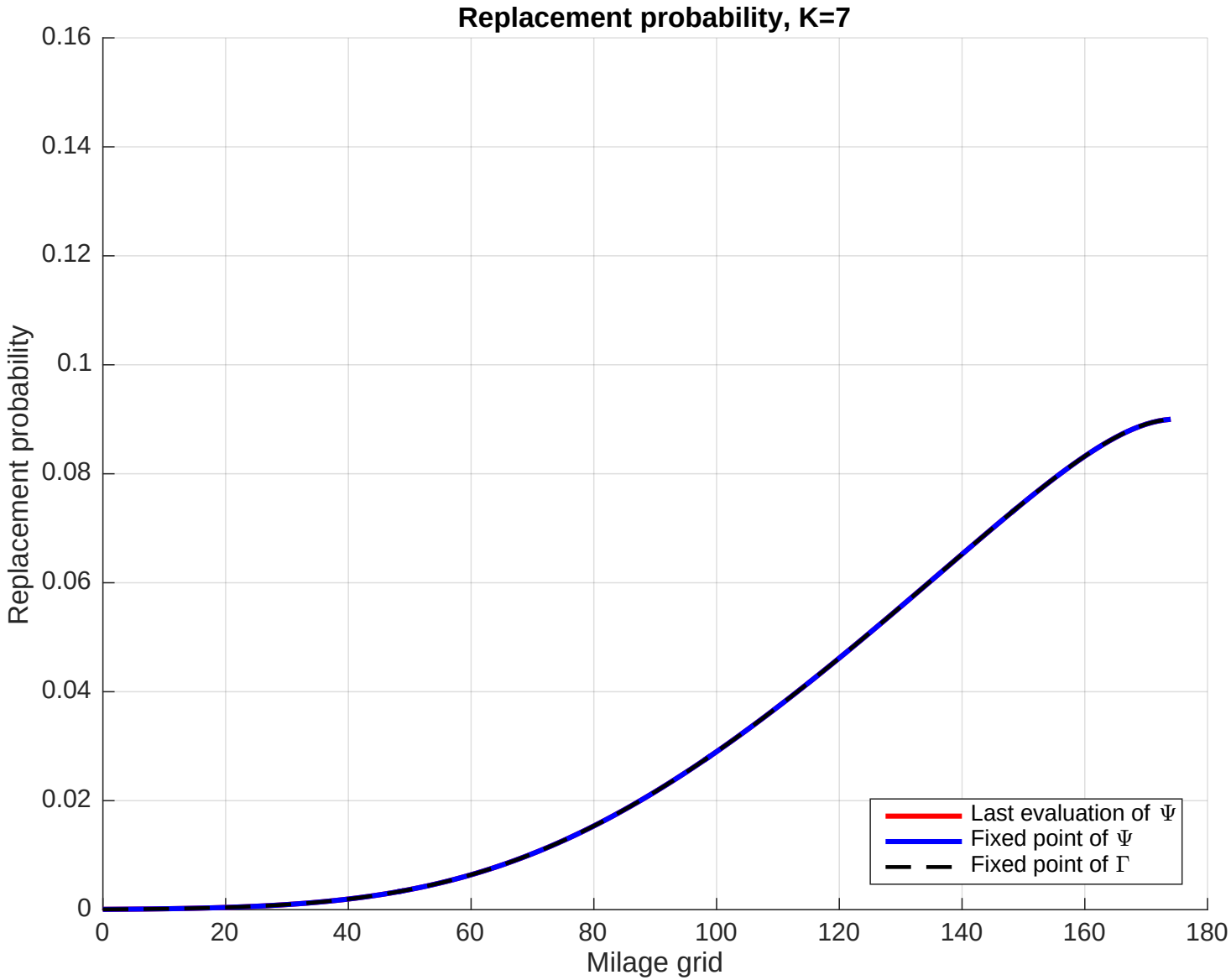
Replacement probability, K=3



Replacement probability, K=4



Replacement probability, MLE



Hotz-Miller's two step estimator

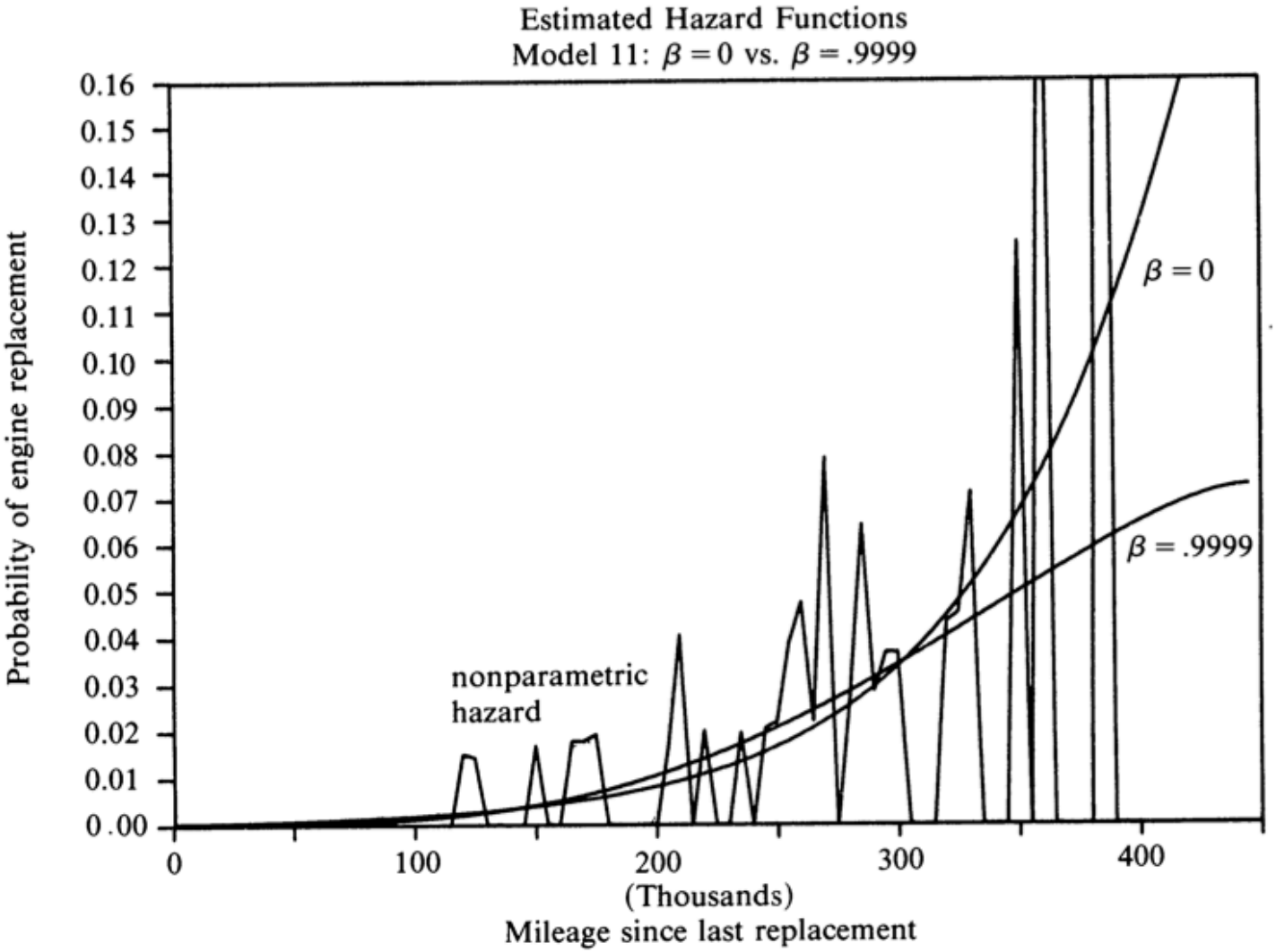
- The CCP estimators were defined as the values of α that solve systems of equations of the form

$$\arg\min_{\alpha\in\Theta} \sum_{i=1}^{N}\sum_{j=1}^{J} Z_i^j \left[I(a_i = j) - \tilde{P}_{(\alpha,\hat{\theta}_f)}(P^0)(j|x) \right]$$

  where Z_i is are vectors of instrumental variables (e.g.) functions of x_i
- Easy to show that the 1-stage PI estimator is a CCP estimator with $Z_i = \partial\Psi(P^0)(a_i|x_i)/\partial\alpha$ is used as instrument.

Small sample problems

Sometimes it can be hard to get a precise nonparametric estimate of CCP



The Precision of PI Estimators: A Monte Carlo Evidence

TABLE I

MONTE CARLO EXPERIMENT

	Experiment design
Model:	Bus engine replacement model (Rust)
Parameters:	$\theta_0 = 10.47$; $\theta_1 = 0.58$; $\beta = 0.9999$
State space:	201 cells
Number observations:	1000
Number replications:	1000
Initial probabilities:	Kernel estimates

Monte Carlo distribution of MLE
(In parenthesis, percentages over the true value of the parameter)

	θ_0	θ_1
Mean Absolute Error:	2.08 (19.9%)	0.17 (29.0%)
Median Absolute Error:	1.56 (14.9%)	0.13 (22.7%)
Std. dev. estimator:	2.24 (21.4%)	0.16 (26.9%)
Policy iterations (avg.):	6.2	

Monte Carlo distribution of PI estimators (relative to MLE)
(All entries are 100* (K-PI statistic-MLE statistic)/MLE statistic)

		Estimators		
Parameter	Statistics	1-PI	2-PI	3-PI
θ_0	Mean AE	4.7%	1.6%	0.3%
	Median AE	14.2%	0.2%	−0.3%
	Std. dev.	6.8%	1.2%	0.3%
θ_1	Mean AE	18.7%	1.5%	0.2%
	Median AE	25.1%	0.7%	0.6%
	Std. dev.	11.0%	1.3%	0.2%

The Precision of PI Estimators: A Monte Carlo Evidence

TABLE II

RATIO BETWEEN ESTIMATED STANDARD ERRORS AND STANDARD DEVIATION OF MONTE CARLO DISTRIBUTION

Parameters	Statistics	Estimators			
		1-PI	2-PI	3-PI	MLE
θ_0	Ratio	0.801	1.008	1.027	1.022
θ_1	Ratio	0.666	1.043	1.076	1.065

Relative Speed of NPL and NFXP

- For most problems the fixed point iterations (i.e., policy iterations) are much more expensive than likelihood and pseudo-likelihood "hill" climbing iterations.
- The size of the state space does not affect the number of policy iterations in any of the two algorithms.
- Both algorithms were initialized with Hotz-Miller Estimates.
- A&M found that NPL around 5 and 10 times faster than NFXP