Solving and estimating discrete choice models

The Nested Fixed Point Algorithm (NFXP)
vs.
Mathematical Programming with Equilibrium Constraints (MPEC)

Fedor Iskhakov, University of New South Wales
Jinhyuk Lee, Ulsan National Institute of Science and Technology
John Rust, Georgetown University
Kyoungwon Seo, Korea Advanced Institute of Science and Techn.
Bertel Schjerning, University of Copenhagen

June 11th, 2015

Road Map

This Lecture

1. Dynamic Discrete Choice Problems, Infinite Horizon Case
   - General Behavioral framework
   - Structural Estimation by MPEC
   - Structural Estimation by NFXP
   - Example: Rust's model
   - Death to good old NFXP?

2. Maximum Likelihood Estimation of Discrete Markov Decision Models by Sieve Approximations
   - Approximation of Expected Value Function
   - Another application of logit smooting: DP Mixed Logit
   - Approximation errors: Implications for statistical inference

PART I

Dynamic Discrete Choice Problems, Infinite Horizon Case

MATLAB code

CODE: `www.goo.gl/tFDzKx`

(link will only be active today)

Rust (Econometrica, 1987)

Econometrica, Vol. 55, No. 5 (September, 1987), 999-1033

OPTIMAL REPLACEMENT OF GMC BUS ENGINES: AN EMPIRICAL MODEL OF HAROLD ZURCHER

BY JOHN RUST[1]

This paper formulates a simple *regenerative optimal stopping* model of bus engine replacement to describe the behavior of Harold Zurcher, superintendent of maintenance at the Madison (Wisconsin) Metropolitan Bus Company. The null hypothesis is that Zurcher's decisions on bus engine replacement coincide with an optimal stopping rule: a strategy which specifies whether or not to replace the current bus engine each period as a function of observed and unobserved state variables. The optimal stopping rule is the solution to a stochastic dynamic programming problem that formalizes the trade-off between the conflicting objectives of minimizing maintenance costs versus minimizing unexpected engine failures. The model depends on unknown "primitive parameters" which specify Zurcher's expectations of the future values of the state variables, the expected costs of regular bus maintenance, and his perceptions of the customer goodwill costs of unexpected failures. Using ten years of monthly data on bus mileage and engine replacements for a subsample of 104 buses in the company fleet, I estimate these primitive parameters and test whether Zurcher's behavior is consistent with the model. Admittedly, few people are likely to take particular interest in Harold Zurcher and bus engine replacement *per se*. I focus on a specific individual and capital good because it provides a simple, concrete framework to illustrate two ideas: (i) a "bottom-up" approach for modelling replacement investment, and (ii) a "nested fixed point" algorithm for estimating dynamic programming models of discrete choice.

KEYWORDS: Optimal replacement, regenerative optimal stopping models, dynamic programming, controlled stochastic processes, nested fixed point algorithm.

Su and Judd (Econometrica, 2012)

CONSTRAINED OPTIMIZATION APPROACHES TO ESTIMATION OF STRUCTURAL MODELS

BY CHE-LIN SU AND KENNETH L. JUDD[1]

Estimating structural models is often viewed as computationally difficult, an impression partly due to a focus on the nested fixed-point (NFXP) approach. We propose a new constrained optimization approach for structural estimation. We show that our approach and the NFXP algorithm solve the same estimation problem, and yield the same estimates. Computationally, our approach can have speed advantages because we do not repeatedly solve the structural equation at each guess of structural parameters. Monte Carlo experiments on the canonical Zurcher bus-repair model demonstrate that the constrained optimization approach can be significantly faster.

KEYWORDS: Structural estimation, dynamic discrete choice models, constrained optimization.

Death to NFXP?

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.

Monte Carlo study demonstrates the uses of parametric bootstrap to compute

Structural Estimation in Microeconomics

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 (2013): 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:

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

- 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 $x_t = 0$.

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

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

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 $\hat{x} \in \hat{X}$

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

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

Parameter Estimates, Rust (1987)

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		

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

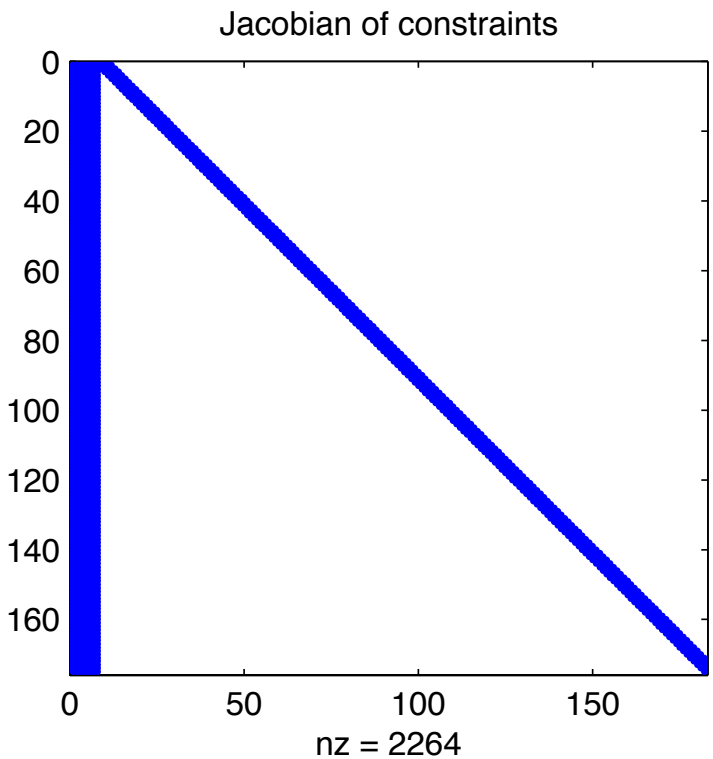
MPEC/MATLAB:

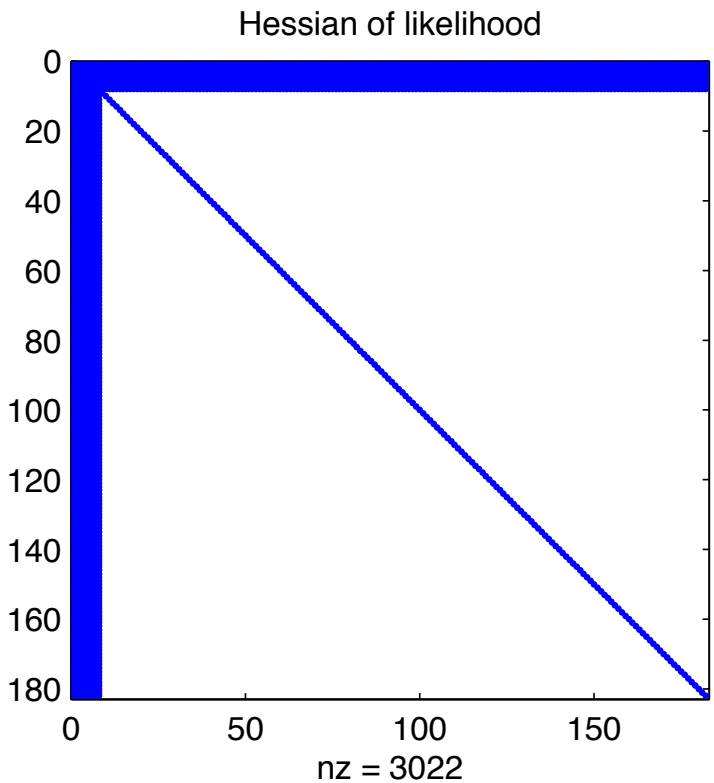
- User need to supply Jacobians, Hessian, and Sparsity Patterns
- Su and Judd do not supply analytical second order derivatives.
- ktrlink provides link between MATLAB and KNITRO solvers.

Sparsity patterns for MPEC

Two key factors in efficient implementations:

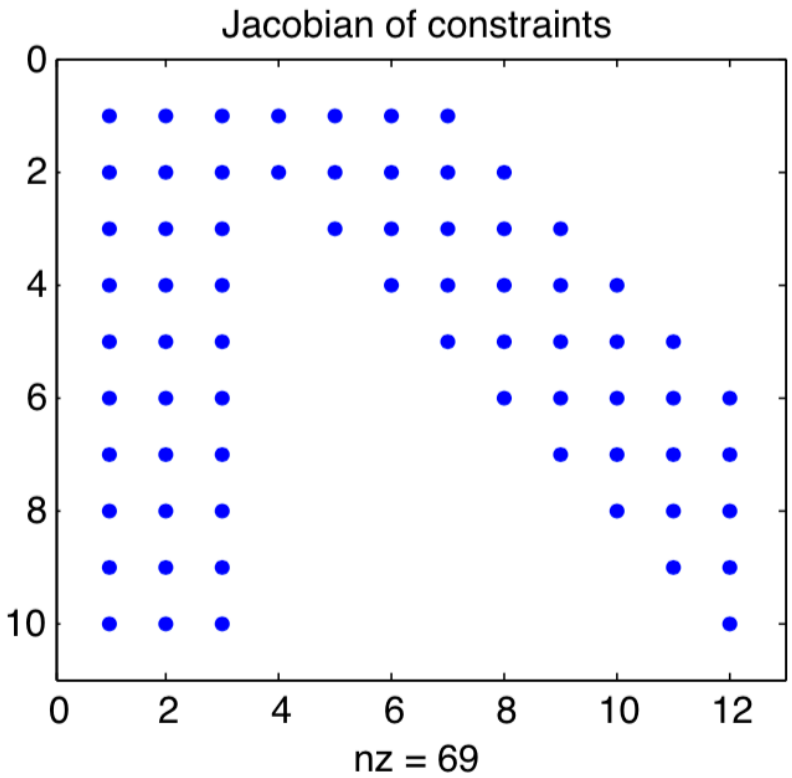
- Provide analytical-derivatives (huge improvement in speed)
- Exploit sparsity pattern in constraint Jacobian (huge saving in memory requirement)

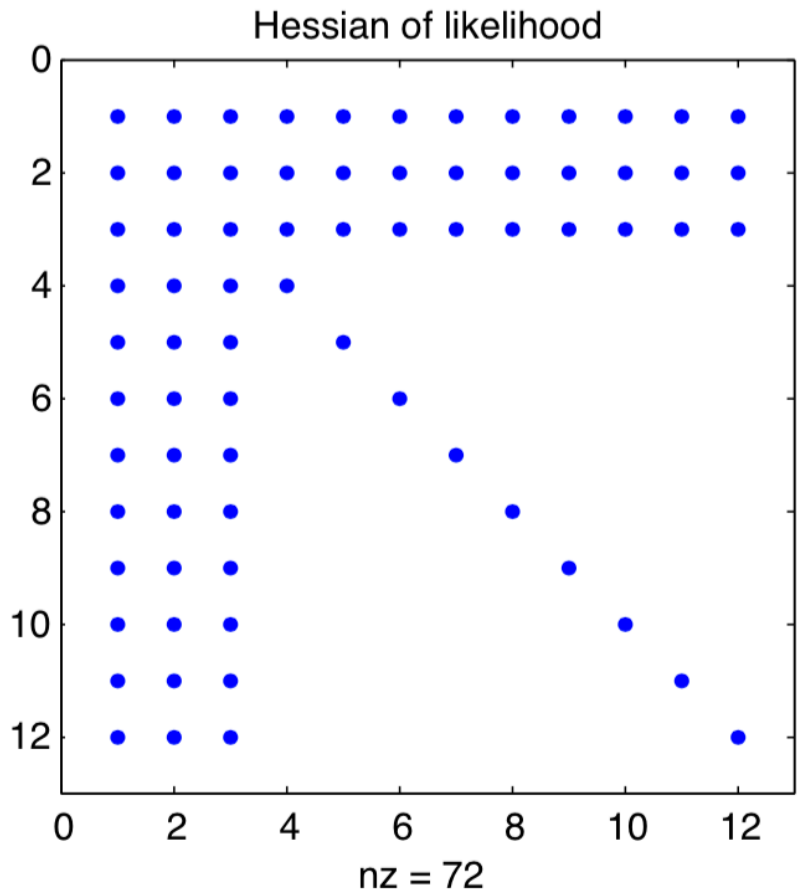




Sparsity patterns for MPEC

Number of gird points, N=10
Number of structural parameters, J=2
Number of parameters in mileage transition probability, J=4





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

Death to NFXP?

Su and Judd (Econometrica, 2012)

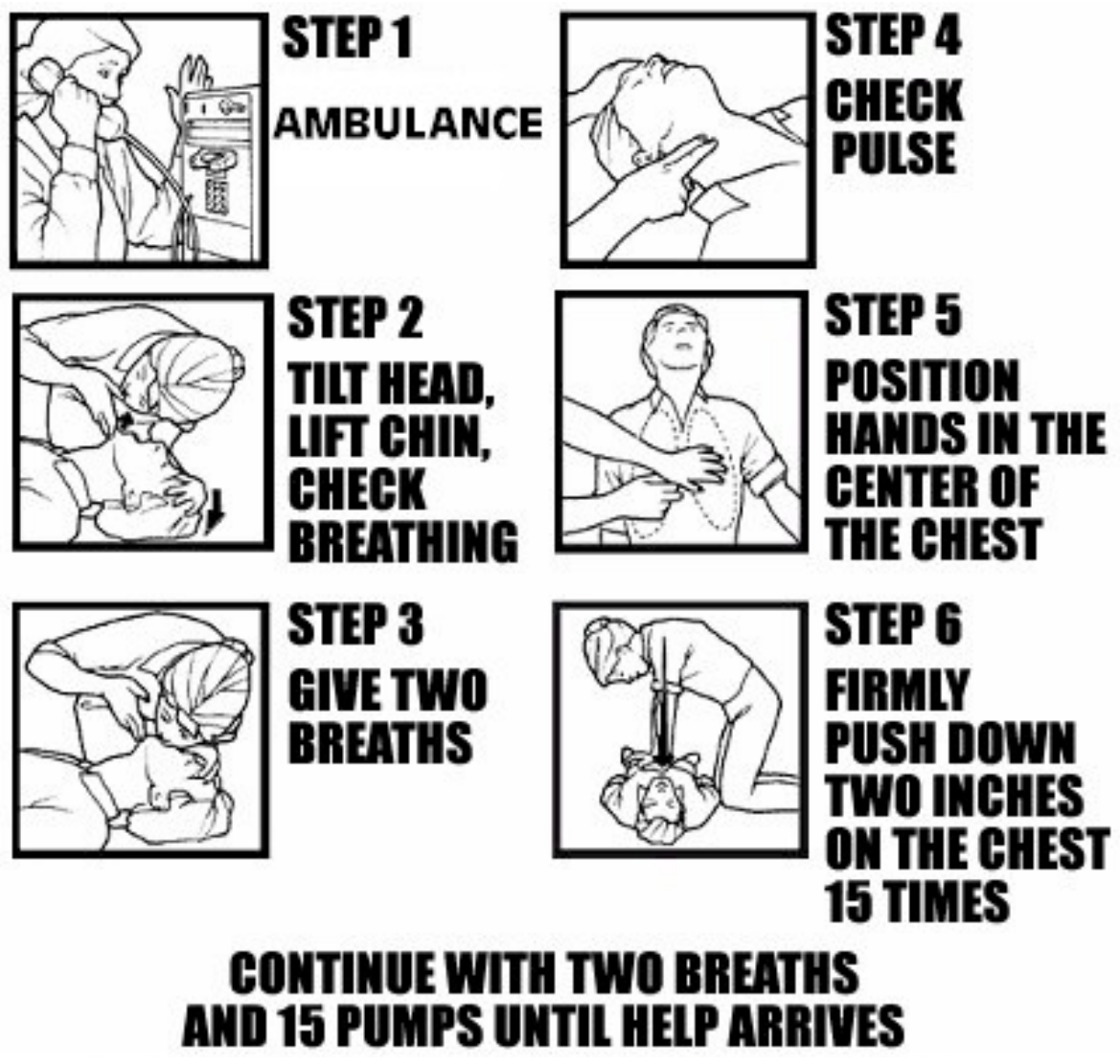
TABLE II

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.

How to do CPR


STEP 1
AMBULANCE
STEP 4
CHECK PULSE
STEP 2
TILT HEAD, LIFT CHIN, CHECK BREATHING
STEP 5
POSITION HANDS IN THE CENTER OF THE CHEST
STEP 3
GIVE TWO BREATHS
STEP 6
FIRMLY PUSH DOWN TWO INCHES ON THE CHEST 15 TIMES
CONTINUE WITH TWO BREATHS AND 15 PUMPS UNTIL HELP ARRIVES

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.)

If NFXP heartbeat is still weak:
Read NFXP pocket guide until help arrives!

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:

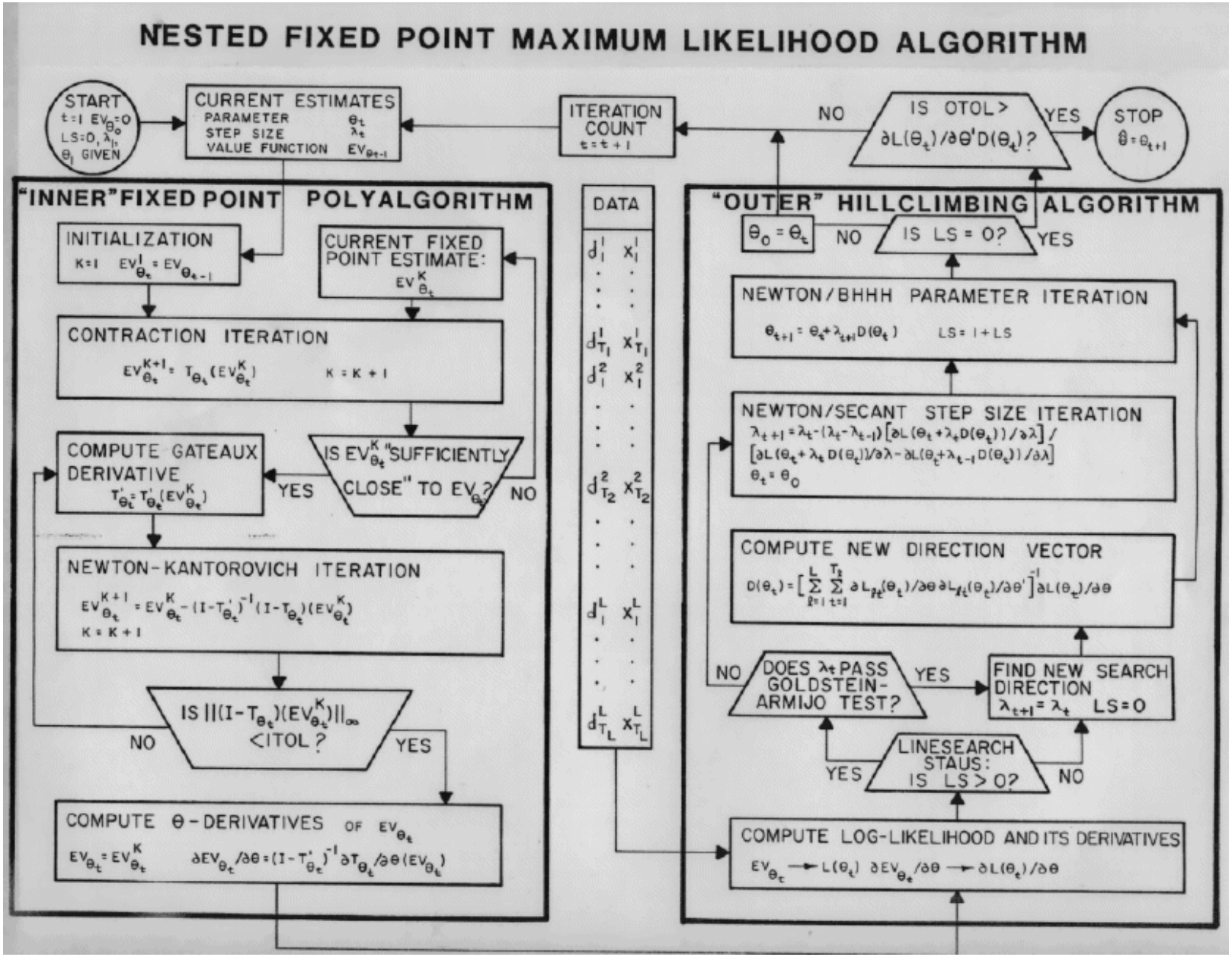
$$\max_{\theta, EV} L(\theta, EV) \text{ subject to } EV = \Gamma_\theta(EV) \tag{3}$$

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 (3) reduces to the unconstrained optimization problem*

$$\max_{\theta} L(\theta, EV_\theta) \tag{4}$$

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:

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

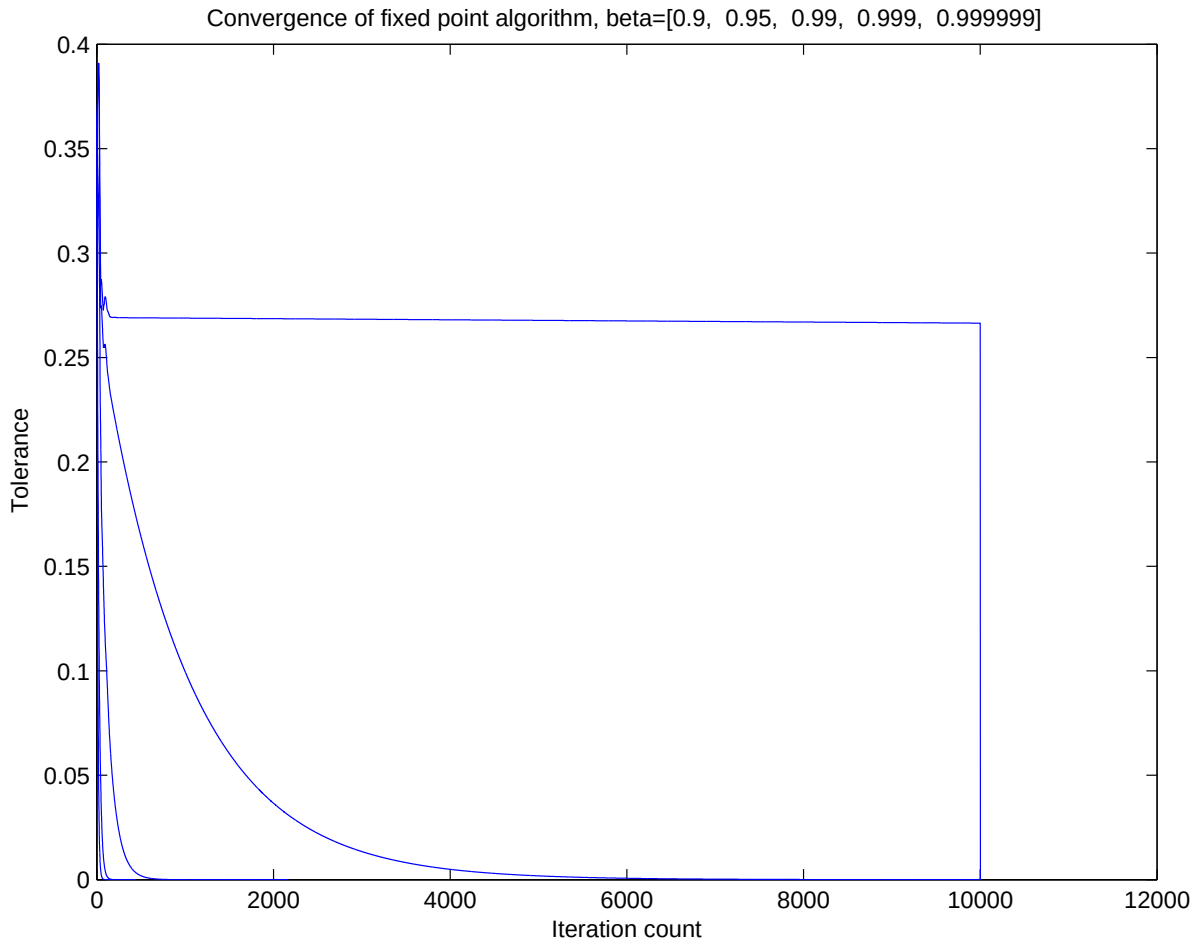
where I is the identity operator on B, and 0 is the zero element of B (i.e. the zero function). The nonlinear operator $I - \Gamma$ has a Fréchet derivative $I - \Gamma'$ which is a bounded linear operator on B with a bounded inverse.

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

Successive Approximations, VERY Slow



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, $\beta = 0.9999$

Quadratic convergence!

```
1
2  Begin contraction iterations
3     j           tol         tol(j)/tol(j-1)
4     1         0.21854635        0.21854635
5     2         0.21852208        0.99988895
6  Elapsed time: 0.00056 (seconds)
7
8  Begin Newton-Kantorovich iterations
9    nwt           tol
10    1       1.03744352e-02
11    2       4.40564315e-04
12    3       8.45941486e-07
13    4       3.63797881e-12
14 Elapsed time: 0.00326 (seconds)
15
16 Convergence achieved!
```

STEP 2: When to switch to Newton-Kantorovich

Observation:

- $tol_k = \|EV_{k+1} - EV_k\| < \beta\|EV_k - EV\|$
- 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 dev=dbellman(pk,  P, mp)
    tmp=P(:,2:mp.n).*repmat(pk(2:mp.n,1)',mp.n,1);
    dev=(mp.beta*[1-(sum(tmp,2)) tmp]);
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 the likelihood

```
% Update u, du and P evaluated in grid points
dc=0.001*mp.grid;
cost=mp.c*0.001*mp.grid;
if numel(theta)>2 % if full MLE
   P = nfxp.statetransition(mp.p, mp.n);
end

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

% Evaluate likelihood function
lp=pk(data.x); % probability of keeping at x

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

% add on log like for mileage process
if numel(theta)>2
   p=[mp.p; 1-sum(mp.p)];
   n_p=numel(p)-1;
   logl=logl + log(p(1+ data.dx1));
end
```

STEP 5: MATLAB implementation of scores

```
% step 1: compute derivative of contraction operator wrt. parameters
dtdmp(:, 1)=P*pk-1; % Derivative wrt RC
dtdmp(:, 2)=-(P*dc).*pk; % Derivative wrt c

% step 2: compute derivative of ev wrt. parameters
devdmp=F\dtdmp;  % F=I-Gamma' is by-product of NK-iteration

% step 3: compute derivative of log-likelihood wrt. parameters
score=bsxfun(@times, (lp- 1 + data.d), ...
[-ones(N,1) dc(data.x,:)]  + (devdmp(ones(N,1),:)-devdmp(data.x,:)) ..
);
```

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

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

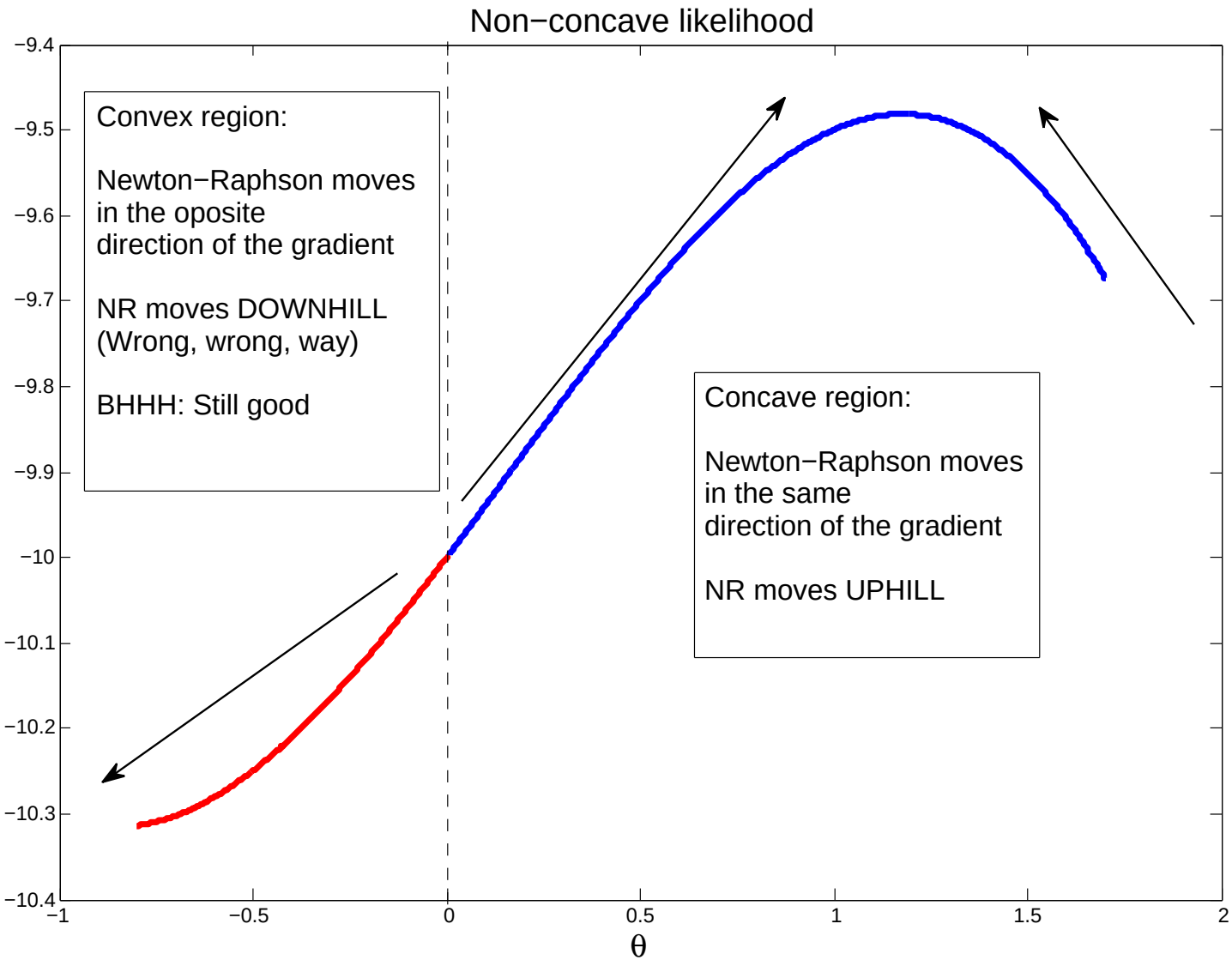
- Why is this valid? Information identity:

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

  (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 order derivatives
  (which are complicated even for this simple model) .

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.

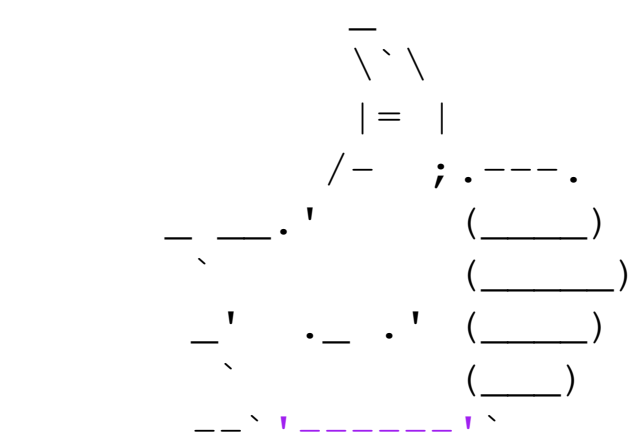
STEP 6: BHHH

"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)

Convergence!

β=0.9999

```
------------------------------------------------------------------
***                    Convergence Achieved
***
------------------------------------------------------------------
```



```
Number of iterations: 9
grad*direc            0.00003
Log-likelihood   -276.74524

    Param.             Estimates          s.e.        t-stat
------------------------------------------------------------------
    RC                   11.1525        0.9167       12.1655
    c                     2.3298        0.3288        7.0856
------------------------------------------------------------------
```

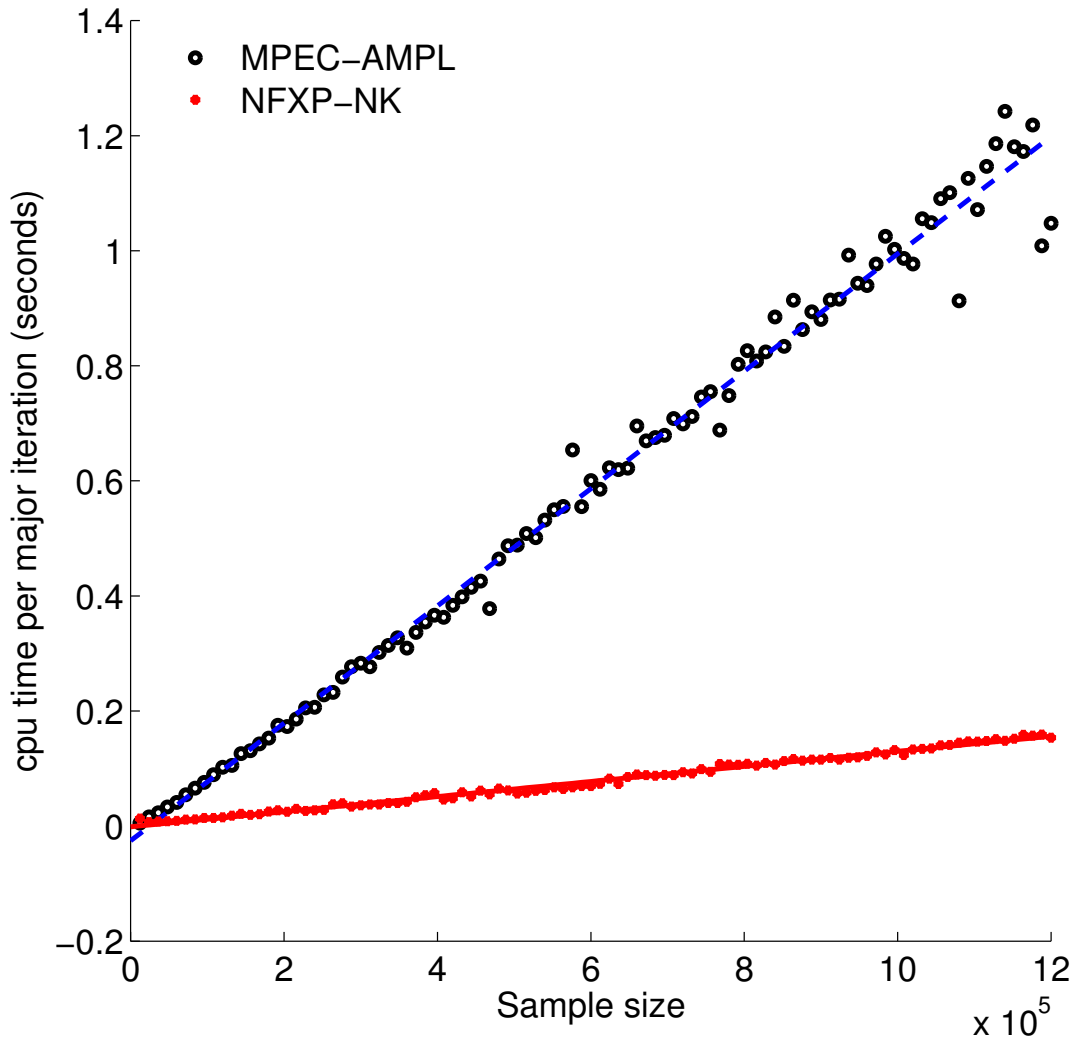
MPEC versus NFXP-NK: sample size 6,000

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

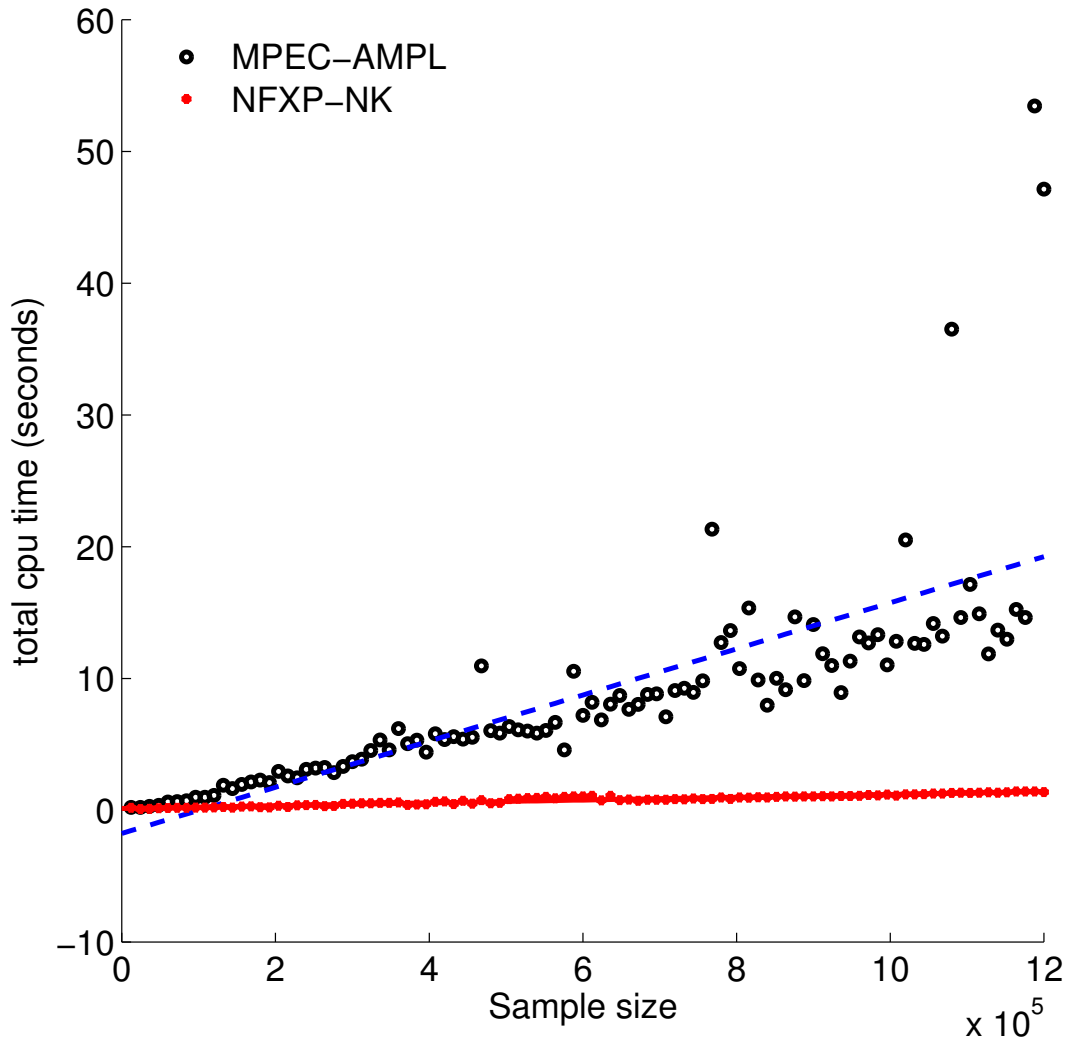
β	Converged (out of 1250)	CPU Time (in sec.)	# of Major Iter.	# of Func. Eval.	# of Bellm. Iter.	# of N-K Iter.
			MPEC-AMPL			
0.975	1247	0.53	9.2	11.7		
0.985	1226	0.76	13.9	32.6		
0.995	1219	0.74	14.2	30.7		
0.999	1249	0.56	9.5	11.1		
0.9995	1250	0.59	9.9	11.2		
0.9999	1250	0.63	11.0	12.7		
			NFXP-NK			
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 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 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.

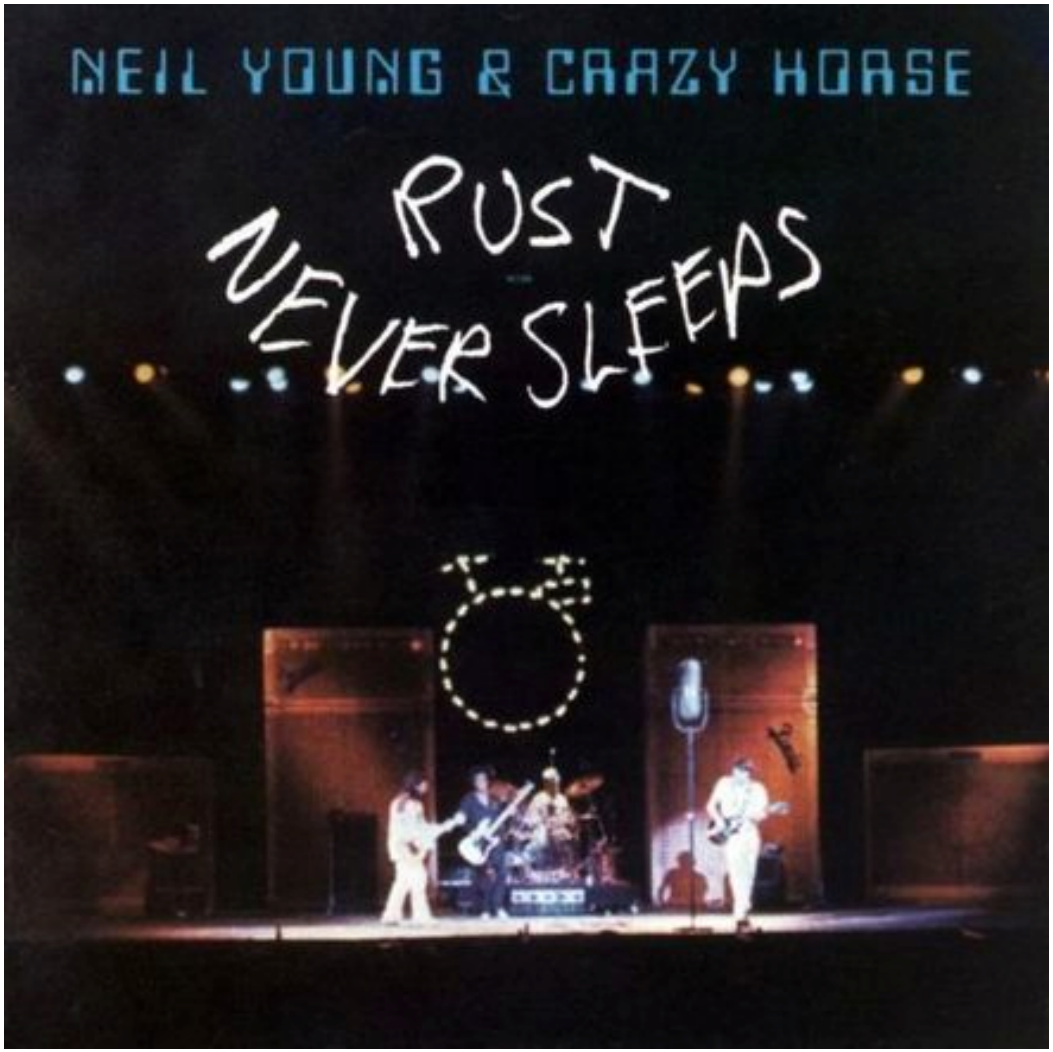
Using the efficient version of NFXP proposed by Rust (1987) we find:

- 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 $\beta \to 1$
- It is non-trivial to compute standard error using MPEC, whereas they are a natural by-product of NFXP.

If the Hessian and the Jacobian fully utilize their special structure:

- MPEC subject to curse of dimensionality in the number of gridpoints (AMPL implementation does not have this problem).

Patient still alive



Another reference


Iskhakov, Lee, Seo, Rust and Schjerning (2015): "Constrained Optimization Approaches to Estimation of Structural Models: Comment"
http://bschjerning.com/papers/nfxp_mpec_comment.pdf

PART II

Maximum Likelihood Estimation of Discrete Markov Decision Models by Sieve Approximations

with Dennis Kristensen (UCL) and Patrick Mogensen (U. Copenhagen)

Approximation

- Most available solution algorithms and estimation procedures make use of numerical approximations in many dimensions:
  1. Value/Policy function
  2. Expectation operator in Bellman equation
  3. Integrals in choice probabilities and likelihood function

- Various approximations are employed such as
  1. Discretization (uniform grids, random grids, low discrepancy grids, etc.)
  2. Parametric approximations (polynomials, splines, wavelets, neural networks, etc.)
  3. Quadrature/Simulation (MCMC, importance sampling, particle filtering, etc.)

Outline part II

1. Estimation and Solution Method
   - Augmentation of model
   - Approximation of value function
   - Approximation of likelihood
2. Theory:
   - Error bounds on value function and MLE
3. Numerical performance
4. Conclusion

The General Problem

Bellman equation

$$V_\theta(z) = \max_{d \in \mathcal{D}(z)} \{u_{\theta_1}(z,d) + \beta \int V_\theta(z') p_{\theta_p}(z'|z,d) dz'\}$$

u_{θ_1} and p_{θ_p}: known up to a set of parameters, θ_1 and θ_p

- **The agent's problem:** Maximize expected sum of current and future discounted utilities
  - d : Discrete control variable, $d \in \mathcal{D}(z) = \{1, 2, .., J\}$.
  - z : Current state, fully observed by agent
  - $z\prime$: Future state; possibly continuous and subject to uncertainty
- **The agents beliefs about z':**
  - Obeys a (controlled) Markov transition probability $p_{\theta_p}(z_{t+1}|z_t, d_t)$
- **Model solution, $V_\theta(z)$**
  - Find the fixed point for the Bellman equation
  - $V_\theta(z)$ can be a very high dimensional function

MLE of Markov Decision Models

- **Econometric problem:** Given observations of n individual agents over T time periods:

$$(d_{i,t}, x_{i,t}),\ t = 1, ..., T \text{ and } i = 1, ..., n,$$

we wish to estimate the underlying Markov decision model.

  - $d_{i,t}$: individual i's discrete choice at time t.
  - $x_{i,t}$: sub-component of individual i's state vector $z_{i,t}$.
  - $z_t = (x_t, \varepsilon_t)$ where ε_t is a set of unobservables.

- **MLE:**

$$\hat{\theta} = \arg\max_{\theta} \ell_n(\theta), \quad \ell_n(\theta) = \sum_{i=1}^{n} \log p(\underline{x}_i, \underline{d}_i; \theta),$$

where $\underline{x}_i = (x_{i,1}, ..., , x_{i,T_i})$, $\underline{d}_i = (d_{i,1}, ..., , d_{i,T_i})$ and $\log p(\underline{x}_i, \underline{d}_i; \theta)$ is the log-likelihood of individual i.

- **Numerical evaluation of model and MLE:** Requires (approximate) computation of value function and likelihood.

Augmentation of Model

To facilitate implementation, we augment the model.

- Let $\eta_t = (\eta_t(1), ..., \eta_t(J))$ be a extreme value shock which is i.i.d. over alternatives and time, and independent of $\{z_t\}$.
- Transition density in augmented model:

$$p_{\theta_p,\lambda}(z_{t+1}, \eta_{t+1}|z_t, \eta_t, d_t) = p_{\theta_p}(z_{t+1}|z_t, d_t) f_\lambda(\eta_{t+1}),$$

where $f_\lambda(\eta) := f(\eta/\lambda)/\lambda$ with $f(\eta)$ being extreme value density and $\lambda > 0$ scale parameter.

- Value function in augmented model:

$$v_{\theta,\lambda}(z_t) = \max_{d_t \in \mathcal{D}} \{u_{\theta_1}(z_t) + \lambda\eta(d_t) + \beta EV_{\theta,\lambda}(z_t, d_t)\},$$

where $EV_{\theta,\lambda}(z_t, d_t)$ is the expected value function,

$$EV_{\theta,\lambda}(z_t, d_t) := \int_{\mathcal{Z}} \int_{\mathbb{R}^J} v_{\theta,\lambda}(z_{t+1}) p_{\theta_p}(z_{t+1}|z_t, d_t) f_\lambda(\eta_{t+1}) d\eta_{t+1} dz_{t+1}.$$

Augmentation of model with extreme value errors

Augmentation of Model

$$v_{\theta,\lambda}(z_t) = \max_{d_t \in \mathcal{D}} \{ u_{\theta_1}(z_t) + \lambda \eta(d_t) + \beta EV_{\theta,\lambda}(z_t, d_t) \},$$

- The addition of η_t to the model works as a smoothing device. It facilitates computation of the (expected) value function and likelihood.
- A similar idea have been used in the estimation of static discrete choice models; see e.g. McFadden (1989) and McFadden and Train (2003).
- One can think of the extreme value density $f_\lambda(\eta)$ as a kernel smoother with $\lambda > 0$ playing the role of a bandwidth.
- We fix $\lambda = \lambda_n$ at a (small) value for a given sample size n. As $\lambda \to 0$ as $\to \infty$, augmented model and MLE is asymptotically equivalent to the original ones.

Sieve Approximation of Value Function

- **Bellman operator:** The augmented model falls within the framework of Rust (1988). Thus, the expected value function solves a fixed point problem:

$$EV_{\theta,\lambda}(z,d) = \Gamma_{\theta,\lambda}(EV_{\theta,\lambda})(z,d), \tag{5}$$

  where $\Gamma_{\theta,\lambda} : \mathcal{V} \mapsto \mathcal{V}$ is the so-called Bellman operator.
- Since $f_\lambda(\eta)$ is an extreme value density $\Gamma_{\theta,\lambda}$ can be written as:

$$\begin{aligned}\Gamma_{\theta,\lambda}(EV)(z,d) \\ = \int_{\mathcal{Z}} \log\left[\sum_{j\in\mathcal{D}} \exp\left[\frac{u_{\theta_1}(z',j)+\beta EV(z',j)}{\lambda}\right]\right] p_{\theta_p}(z'|z,d)dz'.\end{aligned}$$

- The fixed-point problem is an infinite-dimensional problem and so in general numerically infeasible.

Sieve Approximation of Value Function

- Suppose that the expected value function $EV_{\theta,\lambda}(z,d)$ can be approximated by by a set of basis functions,

$$EV_{\theta,\lambda}(z,d) \simeq B_K(z)'\gamma(d), \quad \gamma(d) \in \mathbb{R}^K.$$

- Here, $B_K(z) = (b_1(z), .., b_K(z))$ is a set of K basis functions chosen by the researcher and $\gamma(d)$ is set a coefficients that uniquely characterizes the expected value function
- For example, we can choose $B_K(z)$ as polynomials, such that $B_K(z) = (1, \ z, z^2, .., z^{K-1})$ or Chebyshev polynomials
- As K increases the approximation gets more flexible.

Sieve Approximation of Value Function

Approximate $EV_{\theta,\lambda}$ by combining simulations and sieve methods.

Simulate Bellman operator: With $Z_\theta^{(r)}(z,d) \sim p_{\theta_p}(z'|z,d)$,

$$\hat{\Gamma}_{\theta,\lambda}(EV)(z,d) = \frac{1}{R_1}\sum_{r=1}^{R_1} \log\left[\sum_{j\in\mathcal{D}} \exp\left[\frac{u_{\theta_1}(Z_\theta^{(r)}(z,d),j) + \beta EV(Z_\theta^{(r)}(z,d),j)}{\lambda}\right]\right]$$

Approximate $EV_{\theta,\lambda}$ by $\hat{EV}_{\theta,\lambda}(z,d) = B_K(z)'\hat{\gamma}_{\theta,\lambda}(d)$ where $\hat{\gamma}_{\theta,\lambda}$ solves the approximate fixed-point problem:

$$\hat{\gamma}_{\theta,\lambda} = \arg\min_{\gamma\in\mathbb{R}^{JK}} \sum_{d=1}^{J}\sum_{r=1}^{R_2}[\hat{\Gamma}_{\theta,\lambda}(B'\gamma)(\tilde{Z}^{(r)},d) - B(\tilde{Z}^{(r)})'\gamma(d)]^2,$$

for some (random) grid $\tilde{Z}^{(r)}$, $r = 1,...,R_2$.

Sieve Approximation of Value Function

- The above least-squares problem can be solved iteratively:

$$\begin{aligned}\hat{\gamma}^{[i]}_{\theta,\lambda}(d) &= [\sum_{r=1}^{R_2} B_K(\tilde{Z}^{(r)})B_K(\tilde{Z}^{(r)})']^{-1} \\ &\quad \times \sum_{r=1}^{R_2} B_K(\tilde{Z}^{(r)})\hat{\Gamma}_{\theta,\lambda}(B'\hat{\gamma}^{[i-1]}_{\theta,\lambda})(\tilde{Z}^{(r)}, d)\end{aligned}$$

- This is a standard series regression estimator as used in nonparametric econometrics (Newey, 1997).
- Can be combined with Newton-Kantorivich iterations as mentioned above.

Approximation of Likelihood Function

- Conditional choice probability:

$$P_{\theta,\lambda}(d|x,\varepsilon) = \frac{\exp[\{u_{\theta_1}(x,\varepsilon,d) + \beta EV_{\theta,\lambda}(x,\varepsilon,d)\}/\lambda]}{\sum_{j\in\mathcal{D}} \exp[\{u_{\theta_1}(x,\varepsilon,j) + \beta EV_{\theta,\lambda}(x,\varepsilon,j)\}/\lambda]}.$$

- Thus, the likelihood of observables is given as

$$p_\lambda(\underline{x},\underline{d};\theta) = \int_{\mathcal{E}^T} p_\lambda(\underline{x},\underline{d},\underline{\varepsilon};\theta) d\underline{\varepsilon}_i,$$

where

$$\begin{aligned} & p_\lambda(\underline{x},\underline{d},\underline{\varepsilon};\theta) \\ = & \prod_{t=1}^{T} P_{\theta,\lambda}(d_t|x_t,\varepsilon_t) \times p_{\theta_p}(x_t,\varepsilon_t|x_{t-1},\varepsilon_{t-1},d_{t-1}). \end{aligned}$$

Approximation of Likelihood Function

- Given the sieve approximator $\hat{V}_{\theta,\lambda}$, draw $\underline{\varepsilon}^{(s)} \sim g(\underline{\varepsilon})$ from some density $g(\underline{\varepsilon})$ with support $\mathcal{E}^T$ and compute

$$\hat{p}_\lambda(\underline{x}, \underline{d}; \theta) = \frac{1}{S} \sum_{s=1}^{S} \frac{\hat{p}_\lambda(\underline{x}, \underline{d}, \underline{\varepsilon}^{(s)}; \theta)}{g(\underline{\varepsilon}^{(s)})},$$

  where $\hat{p}_\lambda(\underline{x}, \underline{d}, \underline{\varepsilon}^{(s)}; \theta)$ is evaluated using $\hat{EV}_{\theta,\lambda}$.
- Computation can be sped up using more advanced simulators such as MCMC (Norets, 2009) or particle filtering (Brownlees, Kristensen and Shin, 2011).

Approximation error?

What are the implications for statistical inference when approximating the value functions, Bellman operator, conditional choice probabilities and the likelihood function?

Theory: Value Function Approximation

- Suppose $\exists \gamma_{\theta,\lambda} : ||B_K' \gamma_{\theta,\lambda} - EV_{\theta,\lambda}||_\infty = O(K^{-\alpha})$ for some $\alpha > 0$.
- For example: If $z \mapsto EV_{\theta,\lambda}(z, d)$ is s times differential and $B_K(z)$ is chosen as polynomials, then $\alpha = s/\dim(z)$.
- Also define $\zeta(K) := ||B_K||_\infty$. For example, with polynomials, $\zeta(K) = O(K^{1+2\dim(z)})$.

Theorem (1)

Under regularity conditions,

$$\begin{aligned} ||\hat{EV}_{\theta,\lambda} - EV_{\theta,\lambda}||_\infty &= O_P(\zeta(K)K^{-\alpha}) + O_P(\zeta(K)K^{1/2}/\sqrt{R_1 R_2}) \\ &= \textit{approximation bias} + \textit{simulation noise}, \end{aligned}$$

where $R_1 = \#$simulations and $R_2 = \#$random grid points.

Theory: Value Function Approximation

$$||\hat{EV}_{\theta,\lambda} - EV_{\theta,\lambda}||_\infty = O_P(\zeta(K)K^{-\alpha}) + O_P(\zeta(K)K^{1/2}/\sqrt{R_1 R_2}),$$

where $R_1 = \#$simulations and $R_2 = \#$random grid points.

- For fixed R_1, this rate is identical to the one for nonparametric series regression estimators (Newey, 1997, Theorem 1).
- Bias: $\alpha = s/\dim(z)$. Thus, the smoother $EV_{\theta,\lambda}(z,d)$ is the better is the rate. On the other hand, the larger $\dim(z)$ is the larger K has to be chosen.
- Parametric rate is attainable: Choosing $K = R_2^{1/\{2(1+2\dim(z)-\alpha)\}}$ and $R_1 = K\zeta^2(K)$,

$$||\hat{EV}_{\theta,\lambda} - EV_{\theta,\lambda}||_\infty = O_P(1/\sqrt{R_2}).$$

Theory: Simulated MLE

Theorem (3)

Under regularity conditions, the simulated MLE, $\hat{\theta}_{\text{approx}}$, satisfies:

$$\begin{aligned} & \|\hat{\theta}_{\text{approx}} - \hat{\theta}\| \\ = \;& O_P(\zeta(K)K^{-\alpha}) + \{O_P(\zeta(K)K^{1/2}/\sqrt{R_1 R_2}) + O_P(S^{-1/2})\} \\ = \;& \textit{bias} + \textit{variance}, \end{aligned}$$

where $\hat{\theta}$ is the exact MLE, $R_1 = \#$simulations used for Bellman operator, $R_2 = \#$random grid points, and $S = \#$simulations used to compute likelihood.

- The approximate MLE inherits the error of the value function.

More of Harold Zurcher

Decisions: $d \in \{0, 1\}$ with $d = 1$ if engine replaced, and $d = 0$ otherwise, and x being elapsed millage.

Utiltiy

$$u(x_t, d, \theta_1) + \varepsilon_t(d_t) = \begin{cases} -RC - c(0, \theta_1) + \varepsilon_t(1) & \text{if } d_t = 1 \\ -c(x_t, \theta_1) + \varepsilon_t(0) & \text{if } d_t = 0 \end{cases} \tag{6}$$

where $c(x_t) = c\sqrt{x_t}$ maintenance/operating costs

States:

- x_t follows regenerating random walk
- ε is extreme value.

Sieve Approximation: Chebyshev polynomials with $K = m$ nodes are used to approximate value function.

- Where approximate "exact" solution and MLE with $R = 5,000$ simulations and $m = 50$ (49 degree Chebyshev polynomial).
- Compare this with approximate MLE with smaller R and m.

Numerical Performance - Choice Probabilities

APPROXIMATION ERROR IN CONDITIONAL CHOICE PROBABILITIES
ALTERNATIVE APPROXIMATIONS

Number of nodes, *m*	Based on ML estimates for alternative approximations				
	10	20	50	100	5000
2	6.31	6.31	6.31	6.31	6.31
4	1.26	1.16	1.20	1.20	1.17
6	0.16	0.10	0.09	0.09	0.10
8	0.17	0.13	0.11	0.11	0.13
10	0.07	0.03	0.01	0.01	0.02
50	0.05	0.00	0.03	0.02	0.00
	Based on ML estimates for R=5000 and m=50				
	10	20	50	100	5000
2	3.87	3.87	3.87	3.87	3.87
4	1.19	0.96	0.95	0.96	0.95
6	0.34	0.10	0.16	0.16	0.11
8	0.22	0.04	0.07	0.07	0.03
10	0.23	0.01	0.06	0.07	0.02
50	0.23	0.01	0.05	0.05	0.00

Note: All figures are measured in percentage points. Approximation error is measured as the maximum absolute deviation between the approximated choice probability and the "exact" choice probability. The "exact" solution was based on *R=5000*, and *m=50*. In the top panel, the choice probabilities were based on ML estimates based on using the alternative approximations. In lower

Numerical Performance - Bias in MLE

BIAS IN ML ESTIMATES FOR ALTERNATIVE APPROXIMATIONS
RUST'S ENGINE REPLACEMENT MODEL
COST FUNCTION: $C(x) = c\sqrt{x}$

Number of nodes, m	Engine replacement costs, RC			Cost function parameter, c		
	R = 10	R = 20	R = 5000	c = 10	c = 20	c = 5000
2	-3.59	-3.59	-3.59	3.83	3.86	3.86
	(0.58)	(0.58)	(0.58)	(8.06)	(8.07)	(8.07)
4	0.70	0.64	0.64	0.38	0.75	0.73
	(1.73)	(1.72)	(1.72)	(3.84)	(3.95)	(3.94)
6	0.05	-0.02	-0.02	-0.19	-0.04	-0.05
	(1.60)	(1.57)	(1.57)	(3.86)	(3.90)	(3.90)
8	0.09	0.03	0.03	-0.11	0.05	0.05
	(1.60)	(1.58)	(1.58)	(3.84)	(3.90)	(3.89)
10	0.06	0.00	0.00	-0.16	0.00	-0.01
	(1.59)	(1.57)	(1.57)	(3.83)	(3.88)	(3.87)
50	0.06	0.00	0.00	-0.16	0.01	0.00
	(1.59)	(1.56)	(1.56)	(3.81)	(3.85)	(3.85)
Paramter estimate ("exact" solution)			11.14			16.49

Numerical Performance - Likelihood Function

TABLE 3
LIKELIHOOD RATIO
ALTERNATIVE APPROXIMATIONS AGAINST "EXACT" SOLUTION

Number of nodes, *m*	Based on ML estimates for alternative approximations				
	10	20	50	100	5000
2	-7.10	-7.10	-7.10	-7.10	-7.10
4	1.04	0.99	1.01	1.01	0.99
6	-0.14	-0.18	-0.17	-0.17	-0.18
8	0.09	0.05	0.06	0.06	0.05
10	0.03	-0.01	0.00	0.00	-0.01
50	0.04	0.00	0.01	0.01	0.00
Likelihood ("exact" solution)					-298.57

Note: The table presents the log-likelihood value under alternative levels of approximation, differenced against the "exact" solution. The "exact" solution were based on, *R=5000*, and n=50.

Consequences of discretizing the data

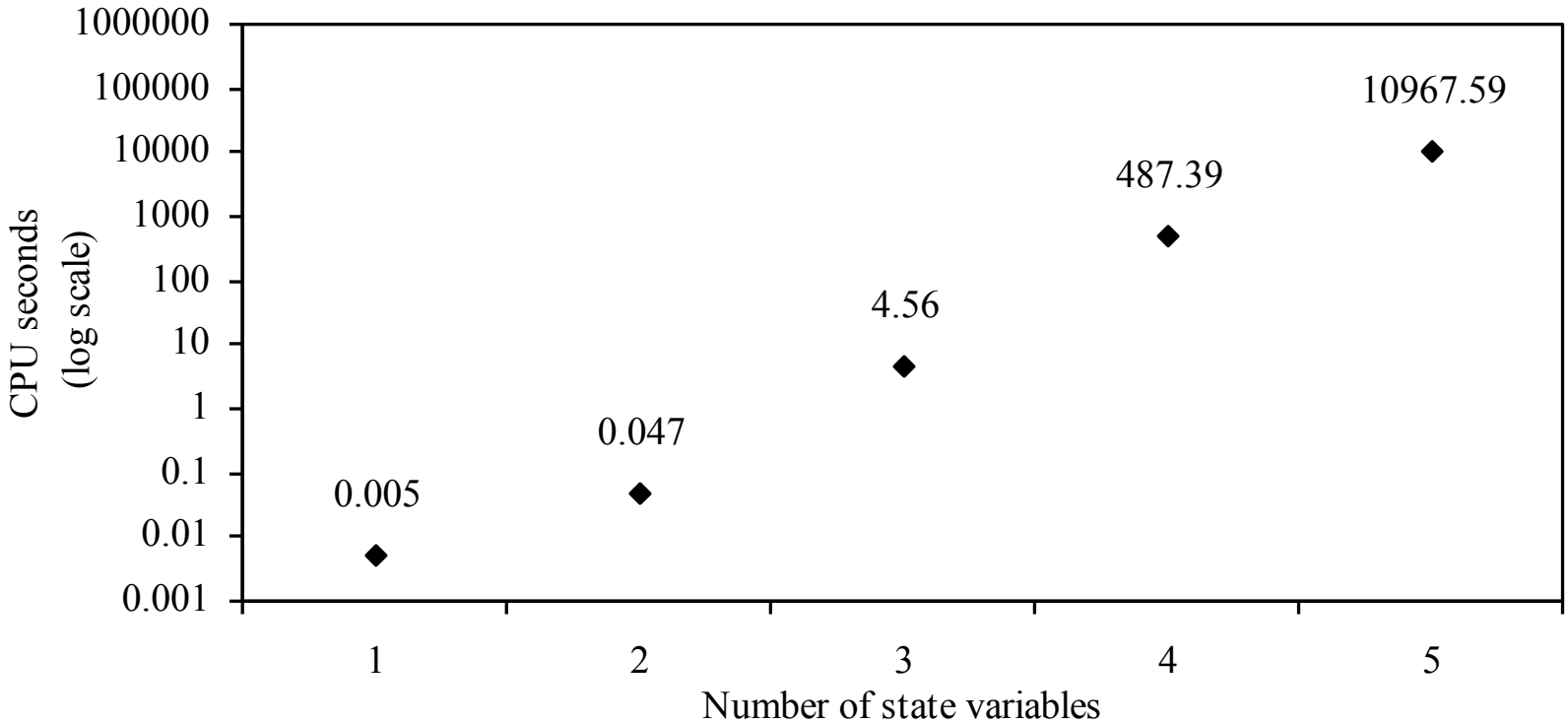
TABLE 4
BIAS AND APPROXIMATION ERROR DUE TO DISCRETIZATION
RUST'S ENGINE REPLACEMENT MODEL
COST FUNCTION: $C(x) = c\sqrt{x}$

Number of grid points	Bias (Standard error)			Mean approx. error (Standard deviation of approx. error)		
	c	RC	LR	$x_t(m)$	$x_t(m)-x_{t-1}(m)$	$dx_t(m)$
2	-10.55	-4.21	-29.85	25.55	-1.85	327.43
	(1.02)	(0.36)		(62.26)	(18.18)	(37.83)
4	-5.26	-1.99	-3.34	5.84	-1.01	162.05
	(2.57)	(1.14)		(32.30)	(15.94)	(18.66)
6	-4.68	-1.67	-4.38	2.89	-0.66	106.93
	(2.33)	(1.03)		(21.34)	(13.81)	(12.28)
8	-3.55	-1.08	-2.27	1.55	-0.53	79.37
	(2.93)	(1.33)		(16.04)	(12.16)	(9.11)
10	-2.46	-0.68	-0.32	0.94	-0.34	62.83
	(2.98)	(1.32)		(13.01)	(11.11)	(7.21)
25	-0.90	-0.11	-0.23	0.20	-0.12	23.14
	(3.57)	(1.54)		(5.26)	(6.75)	(2.80)
50	-0.18	0.06	0.17	0.07	-0.04	9.92
	(3.77)	(1.59)		(2.62)	(4.10)	(1.59)
75	0.10	0.10	0.35	0.02	-0.02	5.60
	(3.87)	(1.61)		(1.74)	(2.64)	(1.35)
90	0.06	0.07	0.29	0.04	-0.02	4.59
	(3.85)	(1.59)		(1.45)	(2.05)	(1.37)
100	0.00	0.00	0.03	0.03	-0.01	4.27
	(3.85)	(1.59)		(0.26)	(0.36)	(0.27)
	Parameter Estimates (Standard error)		Likelihood	Mean of variable (Standard deviation of variable)		
Continuous data	16.39	11.14	-298.56	115.91	3.32	3.32
	(3.83)	(1.57)		(84.77)	(1.42)	(1.42)

Note: Bias is measured as the difference between parameter estimates, based on discretized and continuous data, respectively. The estimated standard errors of the parameter estimates are given in

CPU time

FIGURE 1
CPU TIME USED TO SOLVE MODEL



Note: When the models were solved, I used 100 Halton Draws to calculate integrals and 6 Chebyshev coefficients in each dimension of the state space for the models with up to 4 state variables. For the model with 5 state variables, I used only 5 Cebyshev coefficient in each dimension of the state space. The models were solved using a IMB ThinkPad T41 with a 1.6 GHz Pentium M processor and 2 GB RAM.

Monte Carlo - Unobserved heterogeneity

Rust's model with random coefficients

Experimental design:

- Replacement costs, RC_i
  - *bus specific* and *randomly distributed* in the population of busses
  - normally distributed with mean $\overline{RC}$ and variance σ^2_{RC}.
- Linear cost function $C(x) = cx$
- Parameters: c and $\overline{RC}$, are set roughly equal to the ML estimates from one of the linear specifications of Rust (1987). [1]
- Sample sizes of $N = 100$, $T = 250$
- I draw 5000 Monte Carlo samples, and for each of them, I obtain partial ML estimates for models estimated with and without unobserved heterogeneity.

Random vs Fixed Coefficients

TABLE 5
MONTE CARLO EXPERIMENT
FIXED AND RANDOM COEFFICIENTS

Monte Carlo Distribution of ML and MSL estimates

		Fixed Coefficients		Random Coefficients		
Statistic	σ_{RC}^{dgp}	RC	c	μ_{RC}	σ_{RC}	c
Mean Bias	0	0.05	0.01	0.19	0.41	0.03
	1	-0.52	-0.07	0.05	-0.02	0.01
	2	-1.96	-0.28	0.10	0.08	0.01
Mean Absolute E	0	0.35	0.06	0.43	0.41	0.07
	1	0.57	0.08	0.43	0.29	0.06
	2	1.96	0.28	0.46	0.30	0.07
Monte Carlo	0	0.44	0.074	0.51	0.38	0.082
std. dev	1	0.42	0.066	0.54	0.37	0.080
	2	0.51	0.075	0.57	0.38	0.084
Mean std. Error	0	0.45	0.073	0.46	0.13	0.071
	1	0.41	0.068	0.52	0.08	0.079
	2	0.28	0.053	0.56	0.06	0.080

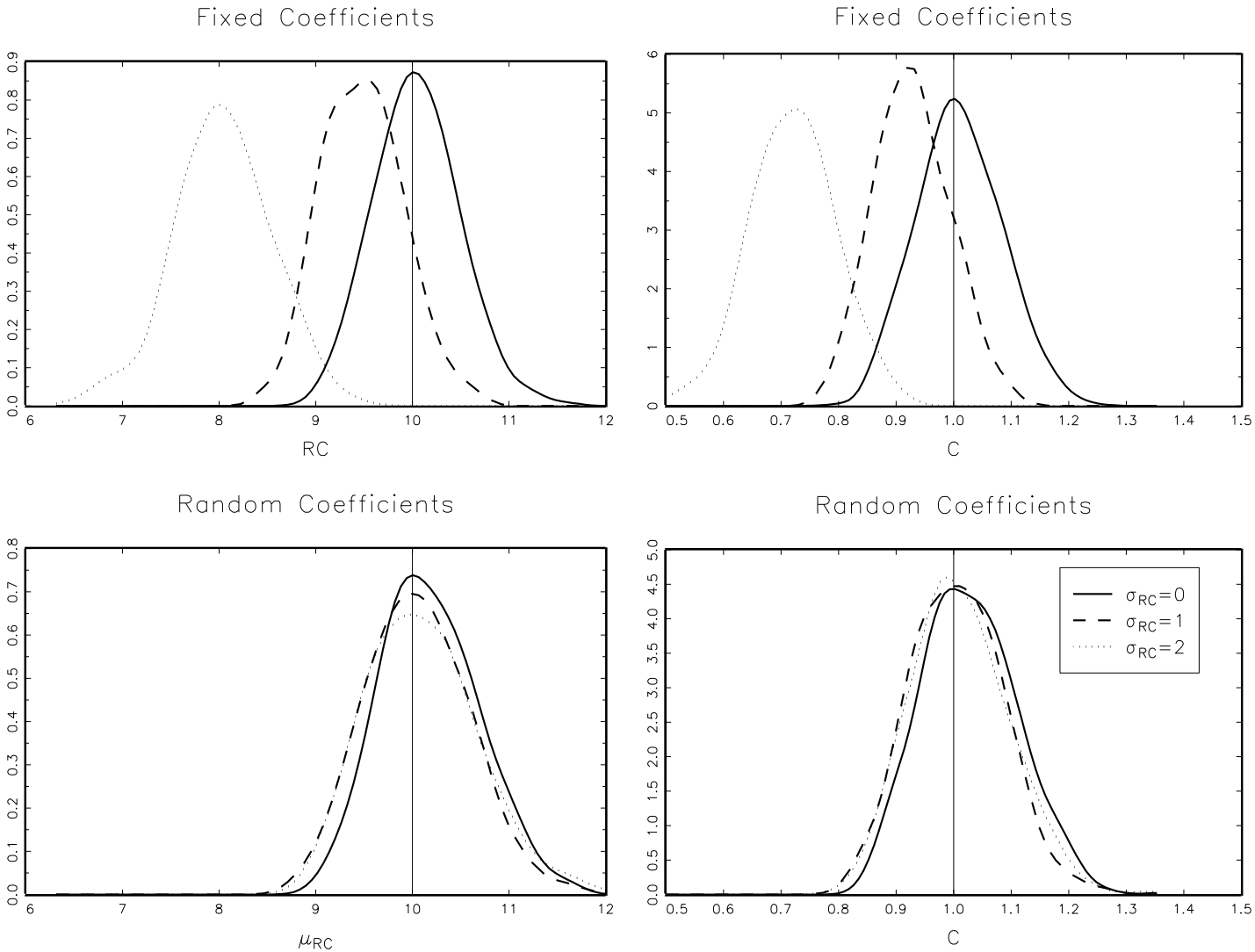
Bias in estimated coefficients ratios

	σ_{RC}^{dgp}	RC/c	μ_{RC}/c
Mean Bias	0	-0.3%	-0.7%
	1	2.4%	0.2%
	2	12.1%	0.2%

Note: The Monte Carlo experiment is based on 1000 Monte Carlo samples of sample size *N=100*,

Random vs Fixed Coefficients

FIXED AND RANDOM COEFFICIENTS



Note: See the note in Table 5 for a description of the experimental design.

Main Results

1. We derive additional biases and variances in approx. MLE due to approx. solution and simulations.
2. These provide guidelines for how approximate solutions and #simulations should be implemented for good performance of approx. MLE.
3. They also allow us to adjust standard errors of approx. MLE to take into account approximation bias and simulation variance.
4. ... And potentially bias-adjust the approx. MLE.
5. As a specific solver, we consider a sieve approximator (aka linear approximator) of the model solution. We analyze its properties which in turn allow us to show how this affects the MLE.
6. Simulations illustrate the theoretical results in finite samples/simulations.