Lecture 2: Structural estimation of discrete decision problems (NFXP and MPEC)

2021 Econometric Society Summer School in Dynamic Structural Econometrics

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

University of Bonn
16-22 August, 2021

Road Map for rest of today

Lecture 2: Constrained versus unconstrained optimization approaches

- **PART I**: The Nested Fixed Point Algorithm (NFXP)
- **PART II**: Mathematical Programming with Equilibrium Constraints (MPEC)
- Leading example: Rust's Engine replacement model
- Matlab implementation (also available in python)

Lecture 3-4: CCP estimation based on the Hotz-Miller inversion (Miller)

- Conditional Independence and the Inversion Theorem
- Identification in Discrete Choice Models
- CCP Estimators

Structural Estimation in Microeconomics

Some 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).
- Bajari, Benkard and Levin (2007): Two step-minimum distance (equilibrium inequalities).
- Arcidiacono Miller (2002): CCP with unobserved heterogeneity (EM Algorithm).
- Norets (2009): Bayesian Estimation (allows for serial correlation in ϵ)
- Su and Judd (2012): MLE using constrained optimization (MPEC)
- and MUCH more
- Any estimator method or solution algorithm of DDC models must confront *Harold Zurcher*

Structural Estimation in Microeconomics

Some 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).
- Bajari, Benkard and Levin (2007): Two step-minimum distance (equilibrium inequalities).
- Arcidiacono Miller (2002): CCP with unobserved heterogeneity (EM Algorithm).
- Norets (2009): Bayesian Estimation (allows for serial correlation in ϵ)
- Su and Judd (2012): MLE using constrained optimization (MPEC)
- and MUCH more
- Any estimator method or solution algorithm of DDC models must confront *Harold Zurcher*

PART I

The Nested Fixed Point Algorithm (NFXP)

Rust (ECTA, 1987):

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

Harold Alois Zuercher June 16, 1926 - June 21, 2020 (age 94)

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

Policy experiments:

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

Who cares about Harold Zurcher?

- Occupational Choice (Keane and Wolpin, JPE 1997)
- Retirement (Rust and Phelan, ECMA 1997)
- Brand choice and advertising (Erdem and Keane, MaScience 1996)
- Choice of college major (Arcidiacono, JoE 2004)
- Individual migration decisions (Kennan and Walker, ECMA 2011)
- High school attendance and work decisions (Eckstein and Wolpin, ECMA 1999)
- Sales and dynamics of consumer inventory behavior (Hendel and Nevo, ECMA 2006)
- Advertising, learning, and consumer choice in experience good markets (Ackerberg, IER 2003)
- Route choice models (Fosgerau et al, Transp. Res. B)
- Fertility and labor supply decisions (Francesconi, JoLE 2002)
- Car ownership, type choice and use (Gillingham et al, WP)
- Residential and Work-location choice (Buchinsky et al, ECMA 2015)
- ...and many more (2309 cites, August 2021)

Formulating, solving and estimating a dynamic model

Components of the dynamic model

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

Solution is given by:

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

Structural Estimation

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

Zurcher's Bus Engine Replacement Problem

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

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

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

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

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

General Behavioral Framework

The decision problem

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

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

where

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

Recursive form of the maximization problem

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

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

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

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

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

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

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

Rust's Assumptions

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

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

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

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

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

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

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

Rust's Assumptions simplifies DP problem

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

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

DP problem under AS and CI

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

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

So that

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

where

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

Bellman equation in expected value function space

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

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

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

where

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

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

Bellman equation in integrated value function space

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

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

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

where

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

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

Compute the expectation of maximum under EV

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

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

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

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

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

HUGE benefits

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

The DP problem under AS, CI and EV

Putting all this together

- Conditional Choice Probabilities (CCPs) are given by

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

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

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

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

Mileage is continuous. How to deal with continuous state?

Rust discretized the range of travelled miles into $n = 175$ bins, indexed with i: $\hat{X} = \{\hat{x}_1, ..., \hat{x}_n\}$ with $\hat{x}_1 = 0$

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

$$p(x'|\hat{x}_k, d, \theta_2) = \begin{cases} Pr\{x' = \hat{x}_{k+j}|\theta_3\} = \theta_{3j} \text{ if } d = 0 \\ Pr\{x' = \hat{x}_{1+j}|\theta_3\} = \theta_{3j} \text{ if } d = 1 \end{cases}$$

- 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

The choice specific expected value function can be found as fixed point on the Bellman operator

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

where

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

$\Pi(d)$ is a $n \times n$ state transition matrix conditional on decision d

Transition matrix for mileage, $d = 0$

If not replacing ($d = 0$)

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

Transition matrix for mileage, $d = 1$

If replacing $(d = 1)$

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

Likelihood Function

Likelihood

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

$$\ell^f\left(x_1, \ldots x_T, d_1, \ldots d_r | x_0, d_0, \theta\right) = \prod_{t=1}^{T} P\left(d_t | x_t, \theta\right) p\left(x_t | x_{t-1}, d_{t-1}, \theta_3\right)$$

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, $p\left(x_t | x_{t-1}, d_{t-1}, \theta_3\right)$?

- Can be estimated in a first step without solving DP problem (non-parametrically or parametrically)
...or jointly with DP problem if $p\left(x_t | x_{t-1}, d_{t-1}, \theta_3\right)$ is fully specified.

Structural Estimation

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

Log likelihood function

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

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

where

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

and

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

The Nested Fixed Point Algorithm

Since the contraction mapping Γ 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):

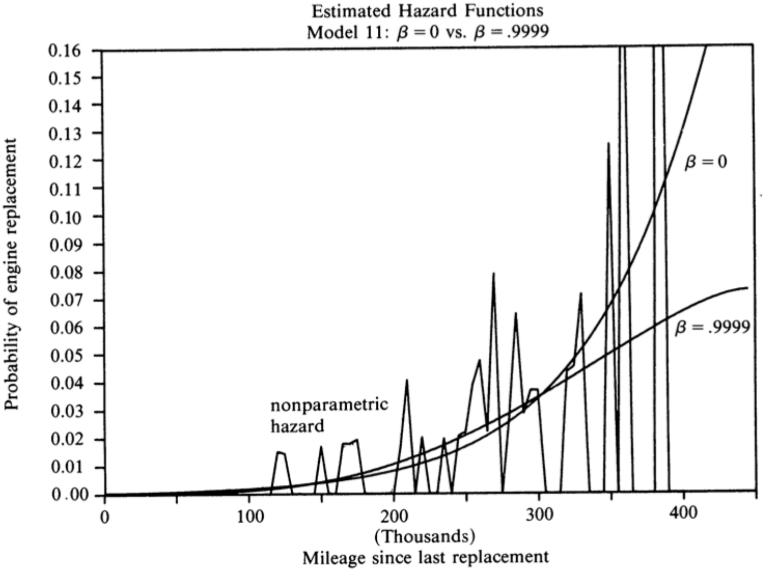
The implicit function EV_θ defined by $EV_\theta = \Gamma(EV_\theta)$ is solved by:

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

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]

	Bus Group		
Cost Function	1, 2, 3	4	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

[a] First entry in each box is (partial) log likelihood value ℓ^2 in equation (5.2)) at $\beta = .9999$. Second entry is partial

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		

MATLAB implementation:

Estimating parameters for bus types 1,2,3,4 (model 19)

Output from `run_busdata.m`:

bertelschjerning — MATLAB_maci64 -nodesktop ▸ matlab_helper — 94×20

```
>> run_busdata
Structural Estimation using busdata from Rust(1987)
Beta            =    0.99990
n               =  175.00000
Sample size     = 8156.00000


    Param.                    Estimates         s.e.         t-stat
-----------------------------------------------------------------------------------------
    RC                           9.7977        1.2146         8.0665
    c                            1.3511        0.3246         4.1627
    p                 (1)        0.1070        0.0034        31.2088
    p                 (2)        0.5152        0.0055        93.0612
    p                 (3)        0.3622        0.0053        68.0450
    p                 (4)        0.0143        0.0013        10.8946
    p                 (5)        0.0009        0.0003         2.6469
-----------------------------------------------------------------------------------------
log-likelihood     = -8607.89037
runtime (seconds) =     0.20874
>>
```

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		

Equilibrium bus mileage and demand for enigines

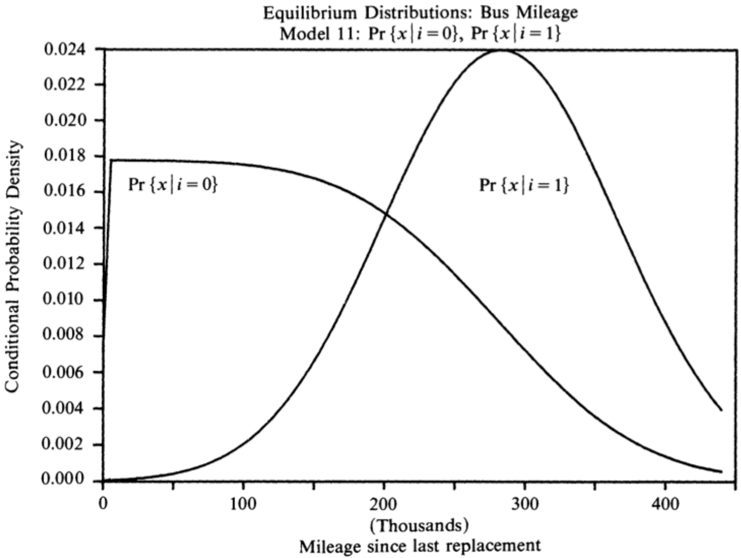
- Let π be the long run stationary (or equilibrium) distribution of the controlled process $\{i_t, x_t\}$
- π is then given by the unique solution to the functional equation

$$\pi(x, i) = \int_y \int_j P(i|x, \theta) p(x|y, j, \theta_3) \pi(dy, dj)$$

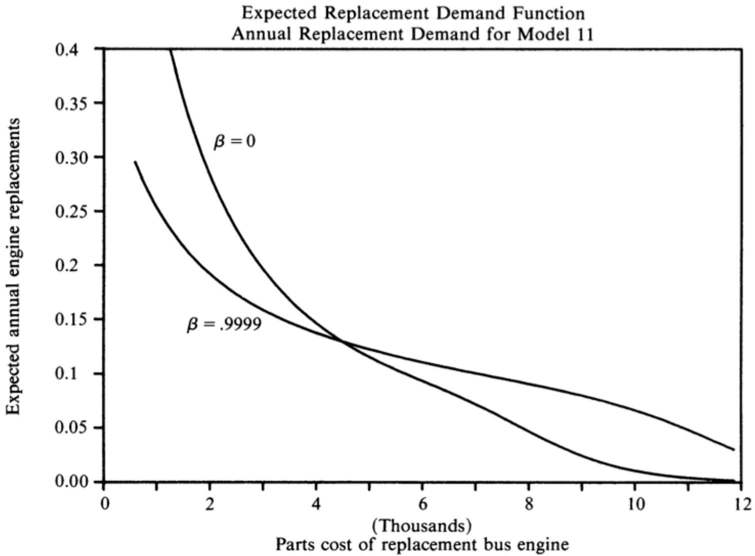
- π is the ergodic distribution of the controlled state transition matrix
- Carly the equilibrium distribution of π is an implicit function of the structural parameters θ, which we emphasize by the notation π_θ
- Given π_θ, we can also obtain the following simple formula for annual equilibrium demand for engines as a function of RC

$$d(RC) = 12M \int_0^\infty \pi_\theta(dx, 1)$$

Equilibrium 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

Constrained and Unconstrained Optimization Approaches to Structural Estimation (MPEC vs. NFXP)

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:

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

- State variables process x_t (mileage since last replacement)

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

- 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

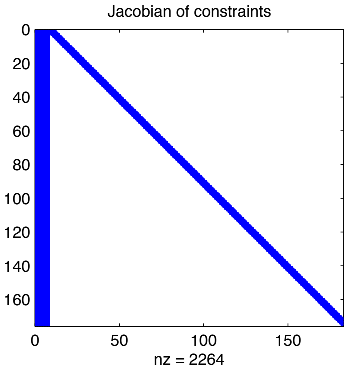
MPEC/MATLAB:

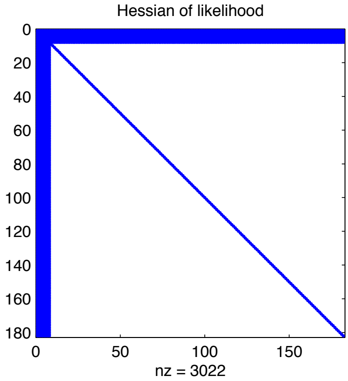
- User need to supply Jacobians, Hessian, and Sparsity Patterns
- Su and Judd do not supply analytical derivatives.
- ktrlink provides link between MATLAB and KNITRO solvers.

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

The choice specific expected value function can be found as fixed point on the Bellman operator

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

where

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

$\Pi(d)$ is a $n \times n$ state transition matrix conditional on decision d

Transition matrix for mileage is sparse

Transition matrix conditional on keeping engine

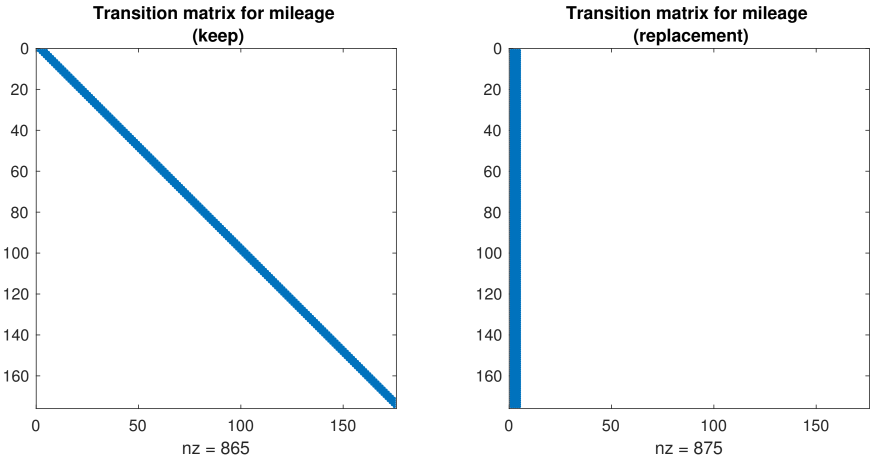
$$
\Pi(d = \text{keep})_{nxn} = \begin{pmatrix}
\pi_0 & \pi_1 & \pi_2 & 0 & \cdot & \cdot & \cdot & 0 \\
0 & \pi_0 & \pi_1 & \pi_2 & 0 & \cdot & \cdot & 0 \\
0 & 0 & \pi_0 & \pi_1 & \pi_2 & 0 & \cdot & 0 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \\
0 & & & & \pi_0 & \pi_1 & \pi_2 & 0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

Transition matrix for mileage is sparse

Transition matrix conditional on replacing engine

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

Transition matrix is sparse



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

References

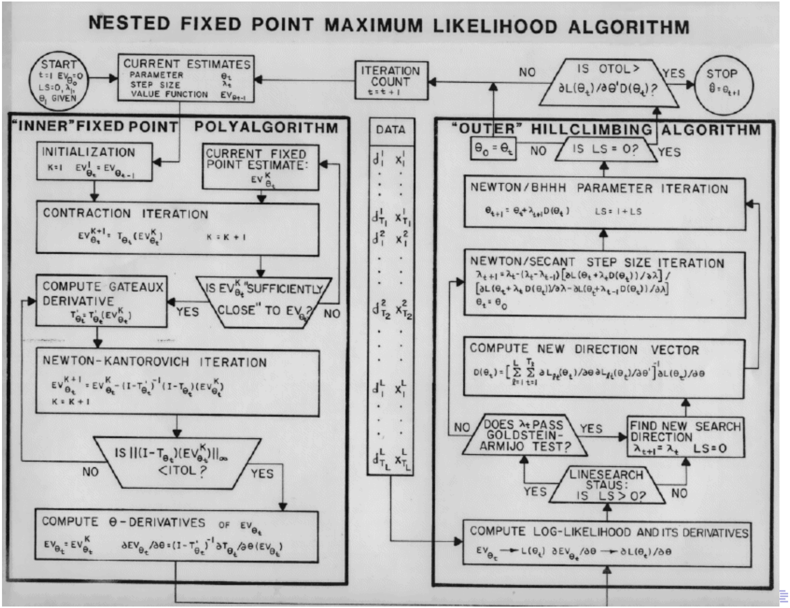
- Rust (1987): "Optimal Replacement of GMC Bus Engines: An Empirical Model of Harold Zurcher" *Econometrica* 55-5, pp 999-1033.
- Rust (2000): "Nested Fixed Point Algorithm Documentation Manual: Version 6" `https://editorialexpress.com/jrust/nfxp.html`
- Iskhakov, F. , J. Rust, B. Schjerning, L. Jinhyuk,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

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 $F = [I - \Gamma](EV_\theta) = 0$ using Newtons method
  $||EV_{k+1} - EV|| \leq A||EV_k - EV||^2$
  quadratic convergence around fixed point, EV

STEP 2: Newton-Kantorovich Iterations

Convert the problem of finding a fixed point $EV_\theta = \Gamma(EV_\theta)$ into the problem of finding a zero of the nonlinear operator $F_\theta(EV_\theta)$

$$F_\theta(EV_\theta) = (I - \Gamma_\theta)(EV_\theta) = 0$$

where I is the identity operator on B, and 0 is the zero element of B (i.e. the zero function).

Newton-Kantorovich iteration:

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

The nonlinear operator $F_\theta = I - \Gamma_\theta$ has a Fréchet derivative $I - \Gamma'_\theta$ 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

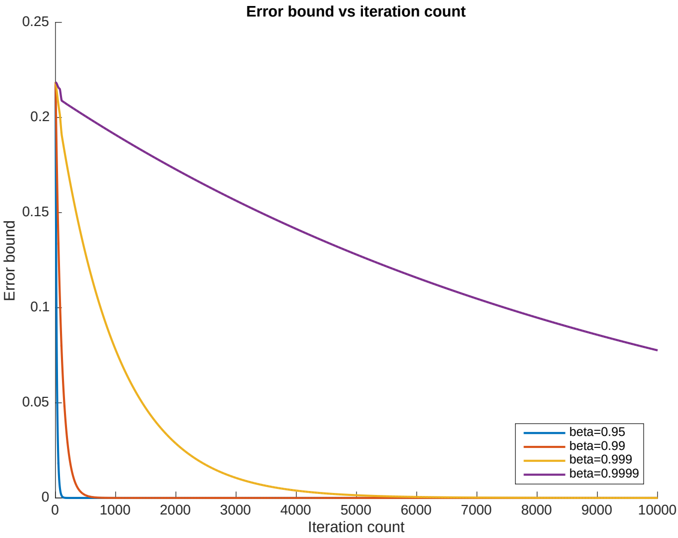
```
Begin contraction iterations
   j             tol         tol(j)/tol(j-1)
   1          0.24310300        0.24310300
   2          0.24307590        0.99988851
   3          0.24304810        0.99988564
   :              :                 :
 9998         0.08185935        0.99990000
 9999         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!
```

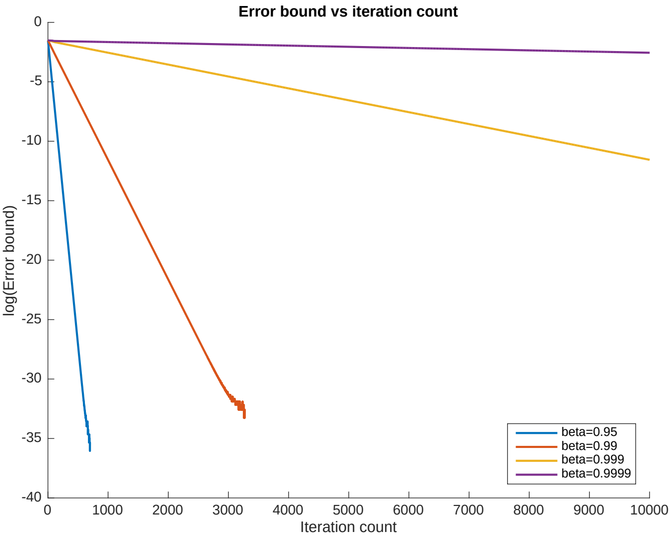
STEP 2: Newton-Kantorovich Iterations

Successive Approximations, VERY Slow



STEP 2: Newton-Kantorovich Iterations

Successive Approximations, Linear convergence



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

Quadratic convergence!

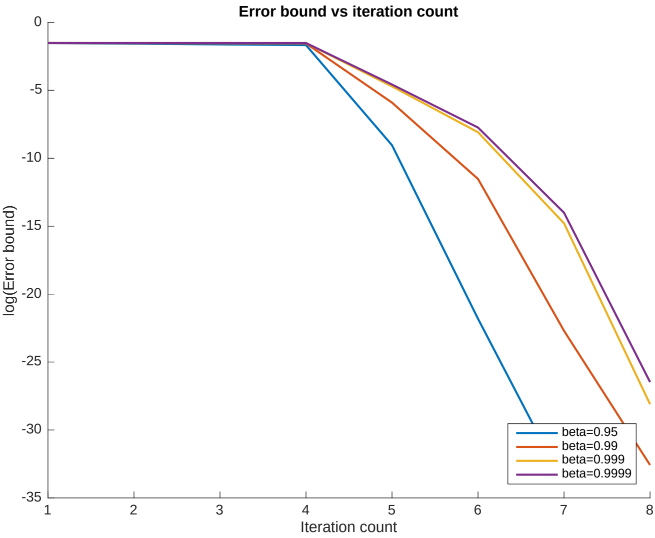
```
Begin contraction iterations
   j           tol        tol(j)/tol(j-1)
   1        0.21854635       0.21854635
   2        0.21852208       0.99988895
Elapsed time: 0.00056 (seconds)

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 4: Analytical Fréchet derivative of Bellman operator

Fréchet derivative

- For NK iteration we need Γ'

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

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

```
function [ev1, pk, dbellman_dev]=bellman_ev(ev, mp, P)
    cost=0.001*mp.c*mp.grid;                % Cost function
    vK=-cost            + mp.beta*ev;       % Value off keep
    vR=-cost(1)-mp.RC + mp.beta*ev(1);      % Value of replacing

    % Need to recenter logsum by subtracting max(vK, vR)
    maxV=max(vK, vR);
    V=(maxV + log(exp(vK-maxV)  +  exp(vR-maxV)));
    ev1=P{1}*V;

    % If requested, also compute choice probability
    if nargout>1
        pk=1./(1+exp((vR-vK)));
    end
    if nargout>2 % compute Frechet derivative
        dbellman_dev=mp.beta*bsxfun(@times, P{1}, pk');
        % Add additional term for derivative wrt Ev(1)
        % since Ev(1) enter logsum for all states
        dbellman_dev(:,1)=dbellman_dev(:,1)+mp.beta*P{1}*(1-pk);
    end
end % end of ZURCHER.bellman_ev
```

Bellman operator can also be written in terms of the smoothed value function

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 $\epsilon's$.

Under our assumptions so far, the smoothed value function, V_σ is a fixed point on the mapping

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

where $V_\sigma = [V_\sigma(1), .., V_\sigma(n)]$ and $u(d) = [u(1,d), .., u(n,d)]$

Easy to implement to implement Fréchet derivative.

STEP 1-4: MATLAB implementation based on smoothed value function

```
function [V1, pk, dBellman_dV]=bellman_integrated(V0, mp, P)
    cost=0.001*mp.c*mp.grid;                             % Cost function
    vK=-cost              + mp.beta*P{1}*V0;             % Value off keep
    vR=-mp.RC-cost(1)     + mp.beta*P{2}*V0;             % Value of replacing
    maxV=max(vK, vR);
    V1=(maxV + log(exp(vK-maxV)  +  exp(vR-maxV)));

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

    if nargout>2 % compute Frechet derivative
        dBellman_dV=mp.beta*(P{1}.*pk + P{2}.*(1-pk));
    end
end % end of ZURCHER.bellman_integrated
```

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

```
cost=0.001*mp.c*mp.grid;
dc=0.001*mp.grid;

% step 1: compute derivative of contraction operator wrt. parameters
dbellman_dmp=zeros(mp.n,2);
dbellman_dmp(:, 1)=(1-pk)*(-1); % Derivative wrt. RC
dbellman_dmp(:, 2)=pk.*(-dc);   % Derivative wrt. c

% step 2: compute derivative of ev wrt. parameters
devdmp=F\dbellman_dmp;

% step 3: compute derivative of log-likelihood wrt. parameters
score=bsxfun(@times, (data.d-pxR), ...
    [-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

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

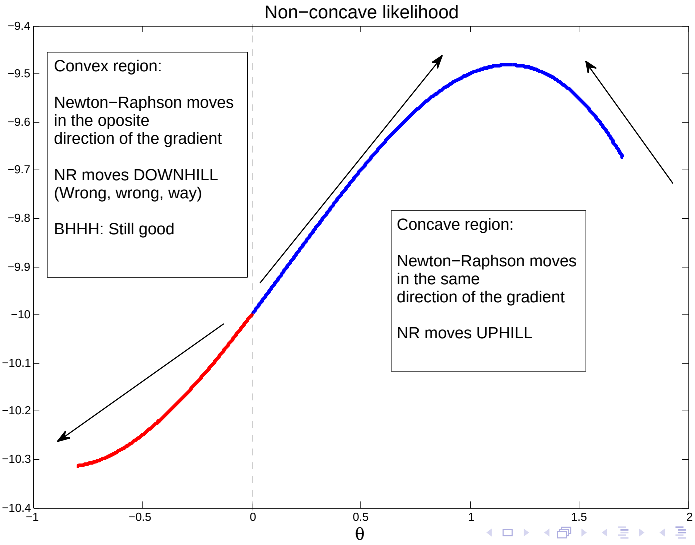
- Why is this valid? Information identity:

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

  (only valid for MLE and CMLE)

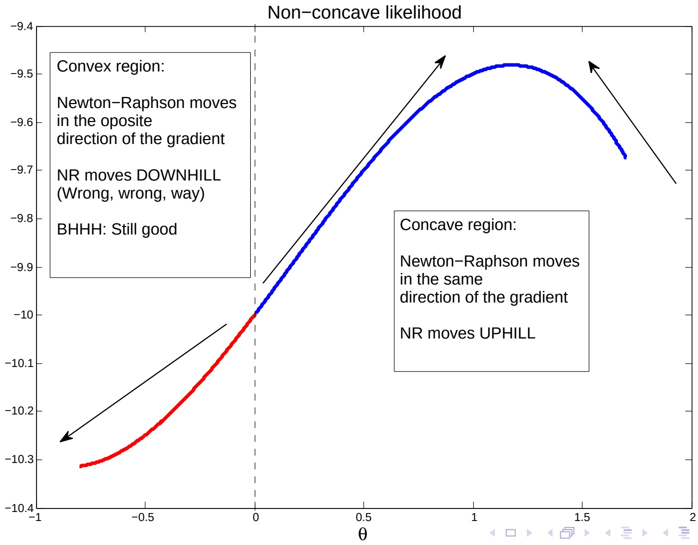
STEP 6: BHHH

Some times linesearch may not help Newtons Method



STEP 6: BHHH

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

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

Use BHHH!

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
---------------------------------------------------------------------
```

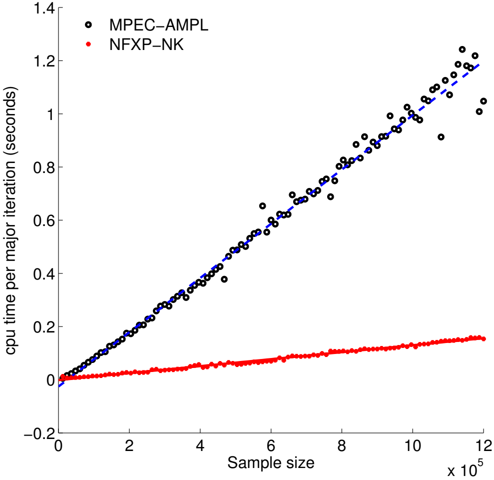
MPEC versus NFXP-NK: sample size 6,000

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

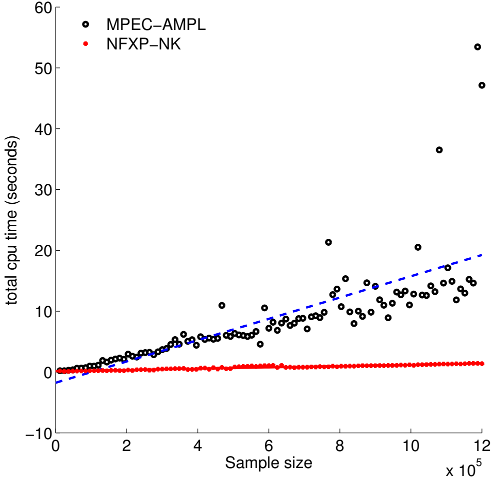
β	Converged (out of 1250)	CPU Time (in sec.)	# of Major Iter.	# of Func. Eval.	# of Bellm. Iter.	# of N-K Iter.
			MPEC-AMPL			
0.975	1247	0.53	9.2	11.7		
0.985	1226	0.76	13.9	32.6		
0.995	1219	0.74	14.2	30.7		
0.999	1249	0.56	9.5	11.1		
0.9995	1250	0.59	9.9	11.2		
0.9999	1250	0.63	11.0	12.7		
			NFXP-NK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

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.
- NFXP does not slow down as $\beta \to 1$

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.

Inference

- NFXP: Trivial to compute standard errors by inverting the Hessian from the unstrained likelihood (which is a by-product of NFXP).
- MPEC: Standard errors can be computed inverting the *bordered Hessian* Reich and Judd (2019): Develop simple and efficient approach to compute confidence intervals.

MPEC does not seem appropriate when estimating life cycle models