Constrained Optimization Approaches to Estimation of Structural Models: Comment

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

January 2015

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

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 and CONOPT

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.

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:

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

- State variables process x_t (mileage since last replacement)

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

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

Zurcher's Bus Engine Replacement Problem

- Zurcher's problem Maximize expected sum of current and future discounted utilities, summarized by the *Bellman equation*:

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

- Under (CI) and (XV) we can integrate out the unobserved state variables

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

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

Structural Estimation

Econometric problem: Given observations of mileage and replacement decisions for $i = 1, .., n$ busses observed over T_i time periods:

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

we wish to estimate parameters $\theta = \{\theta_u, \theta_x\}$ using MLE

Likelihood

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

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

where $P(d_{i,t}|x_{i,t}, \theta)$ is the choice probability given the observable state variable, $x_{i,t}$.

Choice Probabilities and Expected Value functions

- Under the extreme value (XV) assumption choice probabilities are multinomial logistic

$$P(d|x,\theta) = \frac{\exp\{u(x,d,\theta_u) + \beta EV_\theta(x,d)\}}{\sum_{j\in D(y)}\{u(x,j,\theta_1) + \beta EV_\theta(x,j)\}} \tag{3}$$

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

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

- Γ_θ is a *contraction mapping* with unique fixed point EV_θ, i.e.

$$\|\Gamma(EV) - \Gamma(W)\| \leq \beta\|EV - W\|$$

Rust (Econometrica, 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		

Death to NFXP?

Su and Judd (Econometrica, 2012)

TABLE II

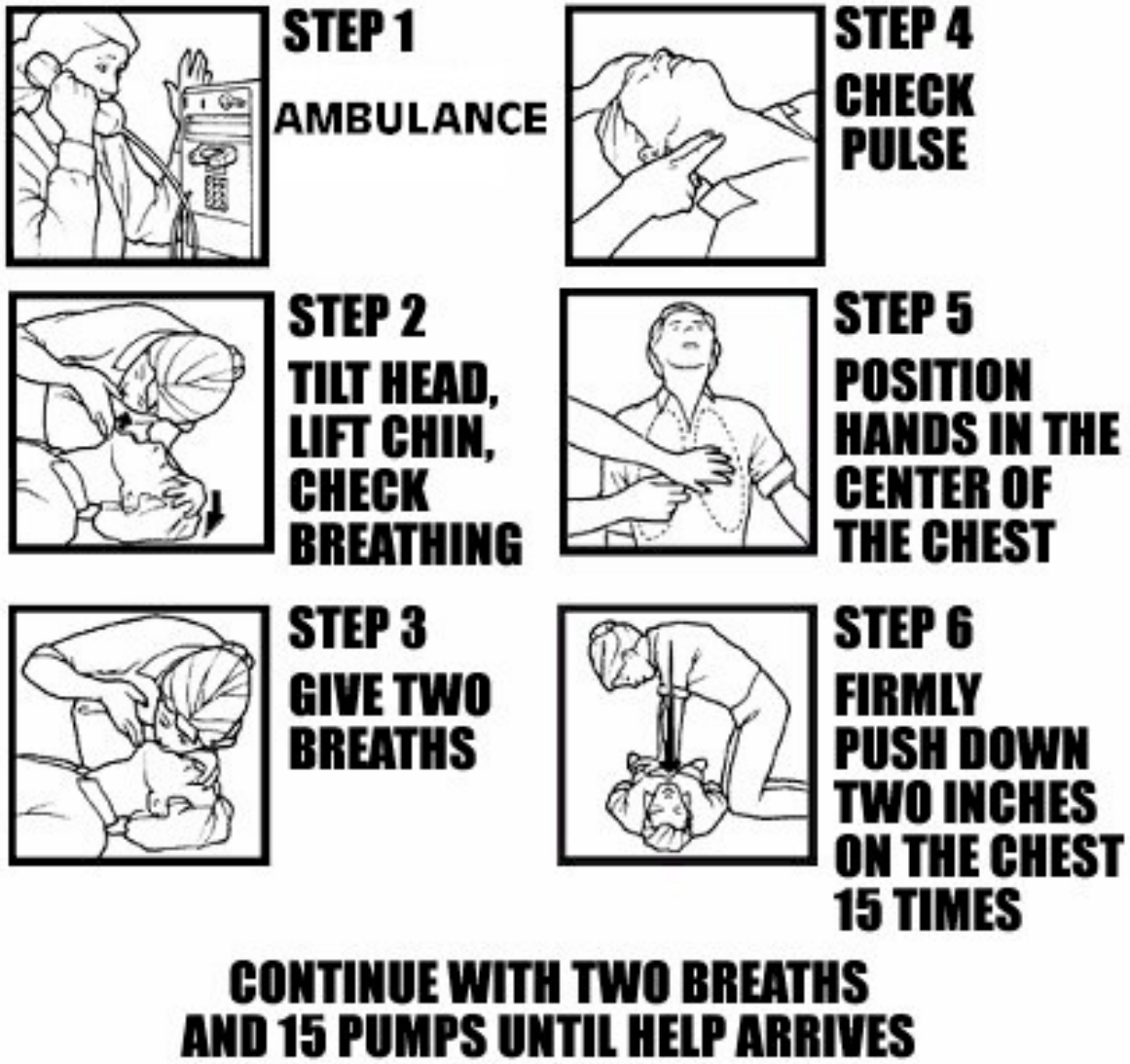
NUMERICAL PERFORMANCE OF NFXP AND MPEC IN THE MONTE CARLO EXPERIMENTS[a]

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

How to do CPR


STEP 1
AMBULANCE
STEP 2
TILT HEAD, LIFT CHIN, CHECK BREATHING
STEP 3
GIVE TWO BREATHS
STEP 4
CHECK PULSE
STEP 5
POSITION HANDS IN THE CENTER OF THE CHEST
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:

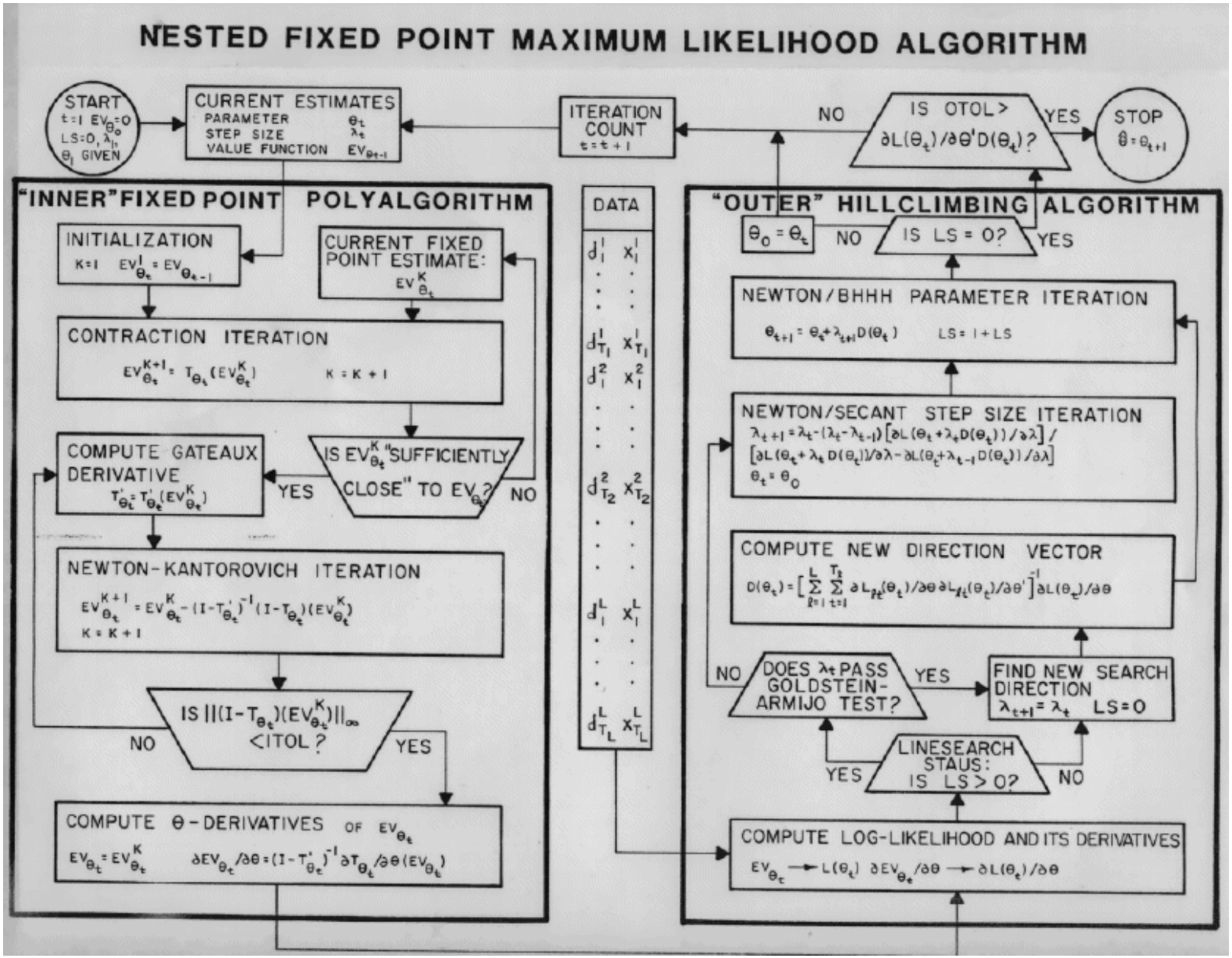
$$\max_{\theta, EV} L(\theta, EV) \text{ subject to } EV = \Gamma_\theta(EV) \tag{4}$$

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 (4) reduces to the unconstrained optimization problem

$$\max_{\theta} L(\theta, EV_\theta) \tag{5}$$

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:

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

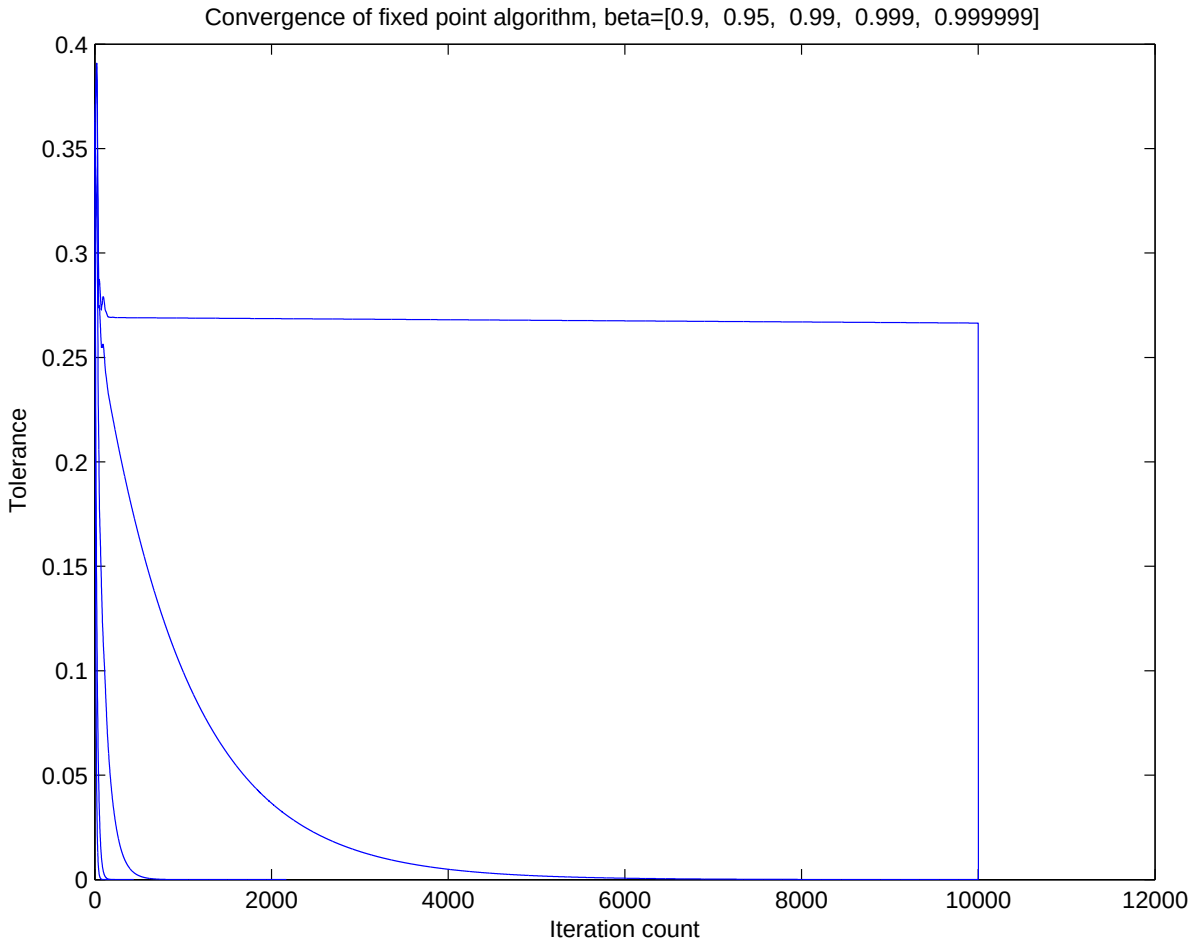
where I is the identity operator on B, and 0 is the zero element of B (i.e. the zero function). The nonlinear operator $I - \Gamma$ has a Fréchet derivative $I - \Gamma'$ which is a bounded linear operator on B with a bounded inverse.

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!

```
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: 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

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

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

STEP 4: 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 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 6: BHHH

- Recall Newton-Raphson

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

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

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

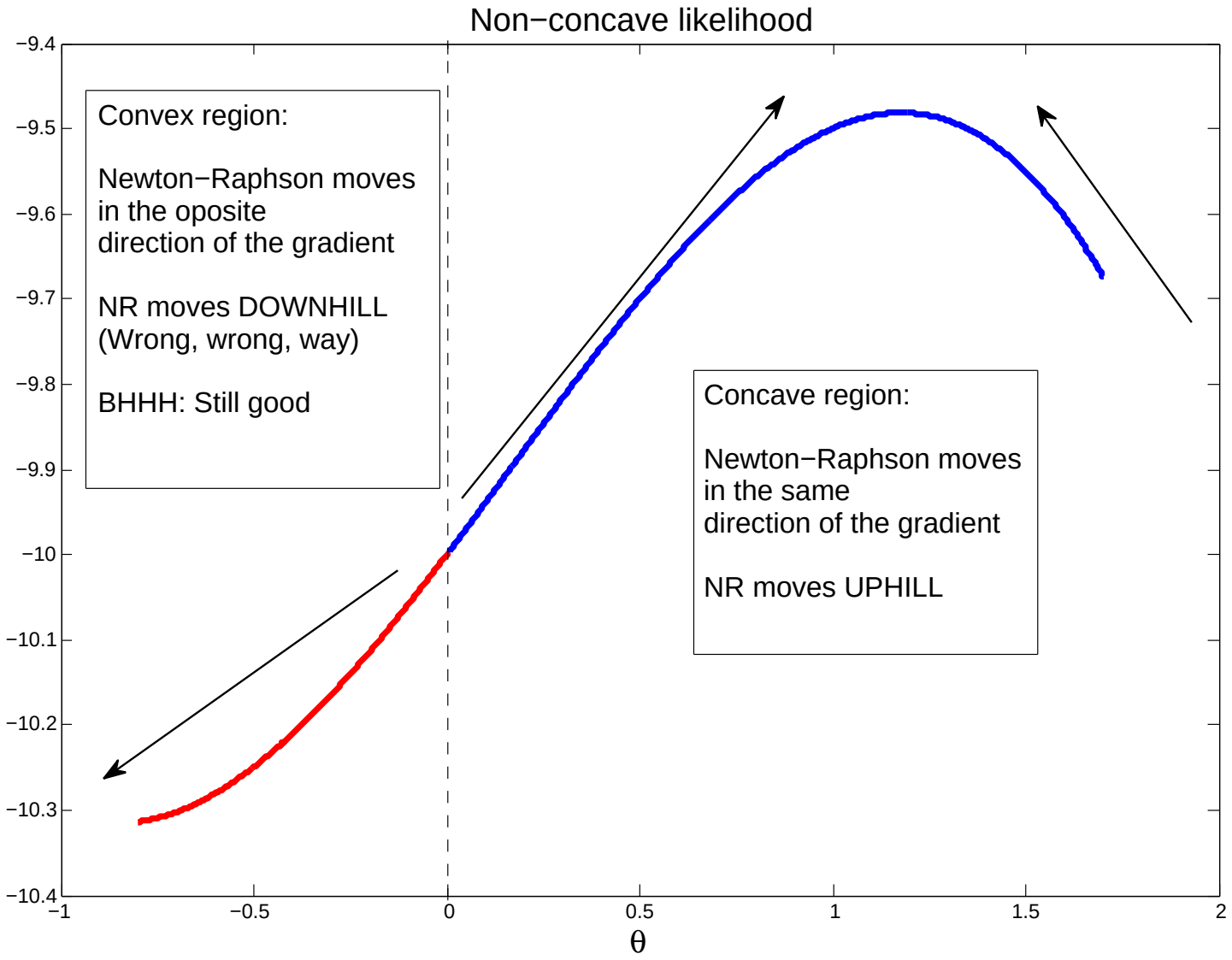
- Why is this valid? Information identity:

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

(only valid for MLE and CMLE)

STEP 6: BHHH

Some times linesearch may not help Newtons Method



STEP 6: BHHH

Advantages

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

Disadvantages

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

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

STEP 6: BHHH

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

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
-----------------------------------------------------------------

Time to convergence is   0 min and 0.07 seconds
```

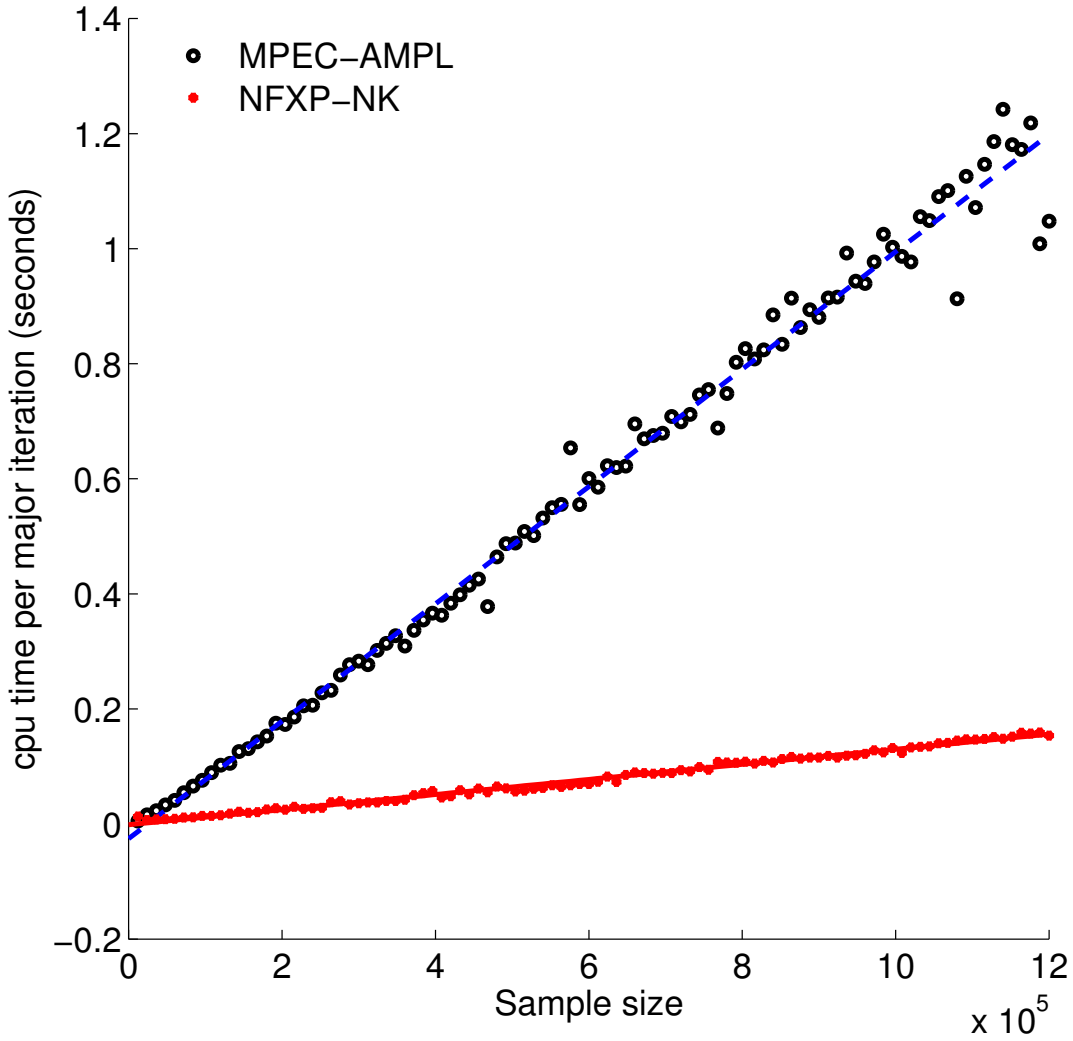
MPEC versus NFXP-NK: sample size 6,000

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

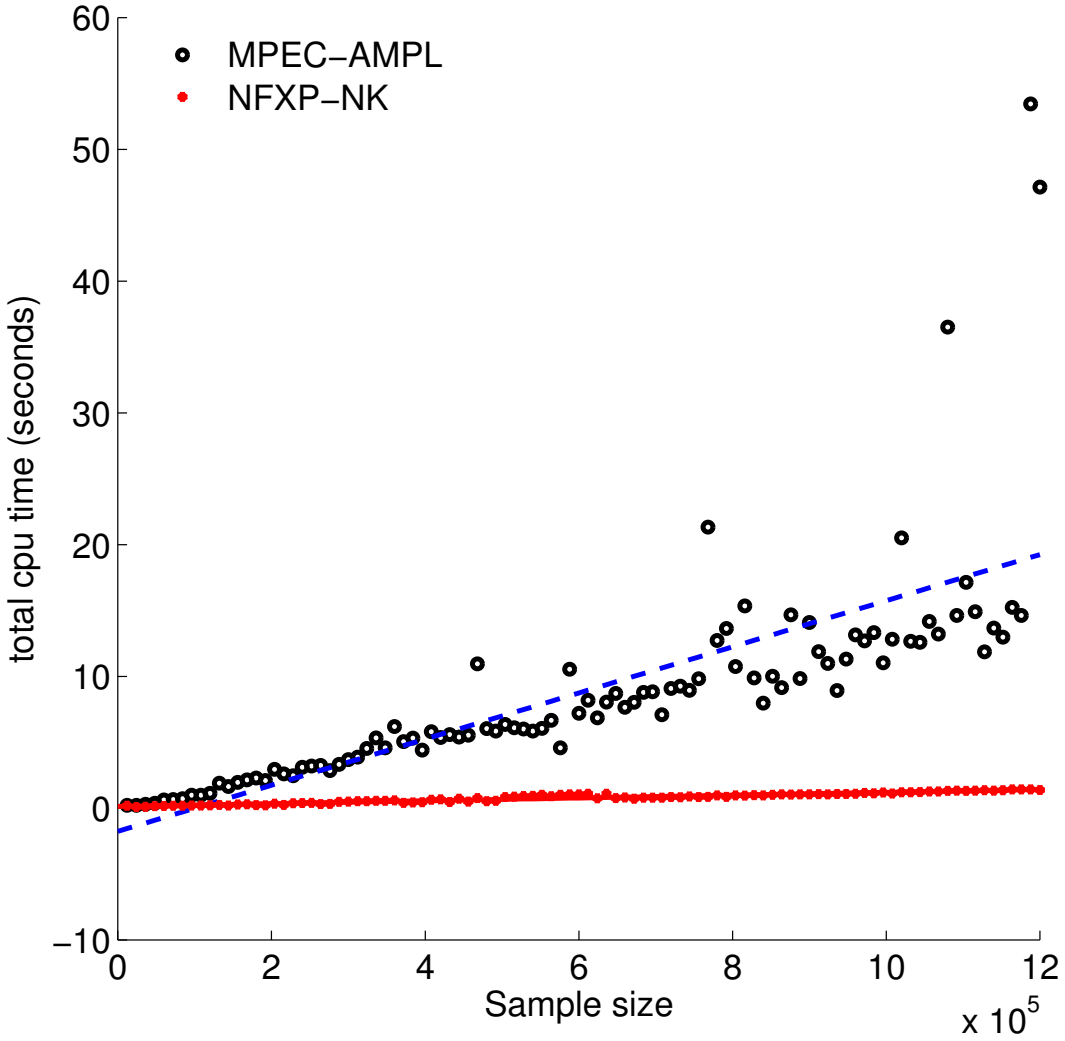
β	Converged (out of 1250)	CPU Time (in sec.)	# of Major Iter.	# of Func. Eval.	# of Bellm. Iter.	# of N-K Iter.
			MPEC-AMPL			
0.975	1247	0.53	9.2	11.7		
0.985	1226	0.76	13.9	32.6		
0.995	1219	0.74	14.2	30.7		
0.999	1249	0.56	9.5	11.1		
0.9995	1250	0.59	9.9	11.2		
0.9999	1250	0.63	11.0	12.7		
			NFXP-NK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

Patient still alive