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.
Bertil Schjerning, University of Copenhagen

January 2015
CONstrained optimIzatIOn approaChes to estImatIOn of structural modells

by Che-Lin Su and Kenneth L. Judd

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.
Monte Carlo study demonstrates the uses of parametric bootstrap to compute
standard errors on structural parameters.

5. CONCLUSION

In this paper, we have proposed a new constrained optimization approach,
MPEC, for estimating structural econometrics models. We have illustrated
that the MPEC approach can be applied directly to maximum-likelihood es-
timation of single-agent dynamic discrete-choice models. Our approach can be
easily implemented using existing standard constrained optimization software.
Monte Carlo results confirmed that MPEC is significantly faster than NFXP,
particularly when the discount factor in the dynamic-programming model is
close to 1.

As shown by Dubé, Fox, and Su (2012), MPEC can also be applied to esti-
mate random-coefficients logit demand models. We believe that our approach
will be useful for estimating structural models in various contexts and applica-
tions. For future research, we plan to investigate the applicability of the MPEC
approach to estimate dynamic discrete-choice games studied in Aguirregabiria
and Mira (2007), Bajari, Benkard, and Levin (2007), Pakes, Ostrovsky, and
Berry (2007), Pesendorfer and Schmidt-Dengler (2008), and Arcidiacono and
Miller (2011).
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. \(\theta\)
- Quasi-Newton algorithm: Usually BHHH, BFGS or a combination.
- Each evaluation of \(L(\theta, EV_\theta)\) requires solution of \(EV_\theta\)

**Inner loop (fixed point algorithm):**
The implicit function \(EV_\theta\) 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:**
  \[
  u(x_t, d, \theta_u) + \varepsilon_t(d_t) = \begin{cases} 
  -RC - c(0, \theta_u) + \varepsilon_t(1) & \text{if } d_t = 1 \\
  -c(x_t, \theta_u) + \varepsilon_t(0) & \text{if } d_t = 0 
  \end{cases}
  \]  
  (1)

- **State variables process** \(x_t\) (mileage since last replacement)
  \[
  p(x_{t+1}|x_t, d_t, \theta_x) = \begin{cases} 
  g(x_{t+1} - 0, \theta_x) & \text{if } d_t = 1 \\
  g(x_{t+1} - x_t, \theta_x) & \text{if } d_t = 0 
  \end{cases}
  \]  
  (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*:

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

- **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 $\ell^f$ has the particular simple form

$$\ell^f_i(\theta) = \sum_{t=2}^{T_i} \log(P(d_{i,t} | x_{i,t}, \theta)) + \sum_{t=2}^{T_i} \log(p(x_{i,t} | x_{i,t-1}, d_{i,t-1}, \theta_x))$$

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

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

(3)

- The expected value function is given by the unique fixed point to the contraction mapping \(\Gamma_\theta\), defined by

\[
EV_\theta(x, d) = \Gamma_\theta(EV_\theta)(x, d)
\]

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

- \(\Gamma_\theta\) is a **contraction mapping** with unique fixed point \(EV_\theta\), i.e.

\[
\|\Gamma(EV) - \Gamma(W)\| \leq \beta \|EV - W\|
\]
## TABLE X

**Structural Estimates for Cost Function** $c(x, \theta_1) = .001\theta_{11} x$

**Fixed Point Dimension** = 175

(Standard errors in parentheses)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Discount Factor</th>
<th>Estimates Log-Likelihood</th>
<th>Data Sample</th>
<th>Heterogeneity Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\beta = .9999$</td>
<td>$RC$</td>
<td>11.7257 (2.597)</td>
<td>10.896 (1.581)</td>
<td>9.7687 (1.226)</td>
</tr>
<tr>
<td></td>
<td>$\theta_{11}$</td>
<td>2.4569 (.9122)</td>
<td>1.1732 (.327)</td>
<td>1.3428 (.315)</td>
</tr>
<tr>
<td></td>
<td>$\theta_{30}$</td>
<td>.0937 (.0047)</td>
<td>.1191 (.0050)</td>
<td>.1071 (.0034)</td>
</tr>
<tr>
<td></td>
<td>$\theta_{31}$</td>
<td>.4475 (.0080)</td>
<td>.5762 (.0075)</td>
<td>.5152 (.0055)</td>
</tr>
<tr>
<td></td>
<td>$\theta_{32}$</td>
<td>.4459 (.0080)</td>
<td>.2868 (.0069)</td>
<td>.3621 (.0053)</td>
</tr>
<tr>
<td></td>
<td>$\theta_{33}$</td>
<td>.0127 (.0018)</td>
<td>.0158 (.0019)</td>
<td>.0143 (.0013)</td>
</tr>
<tr>
<td></td>
<td>$LL$</td>
<td>-3993.991</td>
<td>-4495.135</td>
<td>-8607.889</td>
</tr>
<tr>
<td>$\beta = 0$</td>
<td>$RC$</td>
<td>8.2969 (1.0477)</td>
<td>7.6423 (.7204)</td>
<td>7.3113 (.5073)</td>
</tr>
<tr>
<td></td>
<td>$\theta_{11}$</td>
<td>56.1656 (13.4205)</td>
<td>36.6692 (7.0675)</td>
<td>36.0175 (5.5145)</td>
</tr>
<tr>
<td></td>
<td>$\theta_{30}$</td>
<td>.0937 (.0047)</td>
<td>.1191 (.0050)</td>
<td>.1070 (.0034)</td>
</tr>
<tr>
<td></td>
<td>$\theta_{31}$</td>
<td>.4475 (.0080)</td>
<td>.5762 (.0075)</td>
<td>.5152 (.0055)</td>
</tr>
<tr>
<td></td>
<td>$\theta_{32}$</td>
<td>.4459 (.0080)</td>
<td>.2868 (.0069)</td>
<td>.3622 (.0053)</td>
</tr>
<tr>
<td></td>
<td>$\theta_{33}$</td>
<td>.0127 (.0018)</td>
<td>.0158 (.0019)</td>
<td>.0143 (.0013)</td>
</tr>
<tr>
<td></td>
<td>$LL$</td>
<td>-3996.353</td>
<td>-4496.997</td>
<td>-8614.238</td>
</tr>
</tbody>
</table>

**Myopia tests:**

<table>
<thead>
<tr>
<th>LR Statistic $(df = 1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.724</td>
</tr>
<tr>
<td>3.724</td>
</tr>
<tr>
<td>12.698</td>
</tr>
</tbody>
</table>

**Marginal Significance Level**

<table>
<thead>
<tr>
<th>$\beta = 0$ vs. $\beta = .9999$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0297</td>
</tr>
<tr>
<td>0.0536</td>
</tr>
<tr>
<td>0.00037</td>
</tr>
</tbody>
</table>
Death to NFXP?
Su and Judd (Econometrica, 2012)

### TABLE II

**Numerical Performance of NFXP and MPEC in the Monte Carlo Experiments**

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

*aFor each $\beta$, 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 standard errors on structural parameters.

In this paper, we have proposed a new constrained optimization approach, MPEC, for estimating structural econometrics models. We have illustrated that the MPEC approach can be applied directly to maximum-likelihood estimation of single-agent dynamic discrete-choice models. Our approach can be easily implemented using existing standard constrained optimization software. Monte Carlo results confirmed that MPEC is significantly faster than NFXP, particularly when the discount factor in the dynamic-programming model is close to 1.

As shown by Dubé, Fox, and Su (2012), MPEC can also be applied to estimate random-coefficients logit demand models. We believe that our approach will be useful for estimating structural models in various contexts and applications. For future research, we plan to investigate the applicability of the MPEC approach to estimate dynamic discrete-choice games studied in Aguirregabiria and Mira (2007), Bajari, Benkard, and Levin (2007), Pakes, Ostrovsky, and Berry (2007), Pesendorfer and Schmidt-Dengler (2008), and Arcidiacono and Miller (2011).
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

Nested Fixed Point Algorithm


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

\[
\max_{\theta, EV} L(\theta, EV) \text{ subject to } EV = \Gamma(\theta(EV))
\]  \quad (4)

Since the contraction mapping \( \Gamma \) always has a unique fixed point, the constraint \( EV = \Gamma(\theta(EV)) \) implies that the fixed point \( EV_\theta \) is an implicit function of \( \theta \). Thus, the constrained optimization problem (4) reduces to the unconstrained optimization problem

\[
\max_{\theta} L(\theta, EV_\theta)
\]  \quad (5)

where \( EV_\theta \) is the implicit function defined by \( EV_\theta = \Gamma(EV_\theta) \).
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 → slow when \( \beta \) close to 1

- **Newton-Kantorovich:**
  Solve \([I - \Gamma](EV_\theta) = 0\) using Newton’s 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

Convergence of fixed point algorithm, beta=[0.9, 0.95, 0.99, 0.999, 0.999999]
### STEP 2: Newton-Kantorovich Iterations, $\beta = 0.9999$

**Successive Approximations, VERY Slow**

Begin contraction iterations

<table>
<thead>
<tr>
<th>$j$</th>
<th>$\text{tol}$</th>
<th>$\text{tol}(j)/\text{tol}(j-1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.24310300</td>
<td>0.24310300</td>
</tr>
<tr>
<td>2</td>
<td>0.24307590</td>
<td>0.99988851</td>
</tr>
<tr>
<td>3</td>
<td>0.24304810</td>
<td>0.99988564</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>9998</td>
<td>0.08185935</td>
<td>0.99990000</td>
</tr>
<tr>
<td>9999</td>
<td>0.08185116</td>
<td>0.99990000</td>
</tr>
<tr>
<td>10000</td>
<td>0.08184298</td>
<td>0.99990000</td>
</tr>
</tbody>
</table>

Elapsed time: 1.44752 (seconds)

Begin Newton-Kantorovich iterations

<table>
<thead>
<tr>
<th>nwt</th>
<th>tol</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9.09494702e-13</td>
</tr>
</tbody>
</table>

Elapsed time: 1.44843 (seconds)

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

Quadratic convergence!

Begin contraction iterations
\[
\begin{array}{ccc}
  j & tol & tol(j)/tol(j-1) \\
  1 & 0.21854635 & 0.21854635 \\
  2 & 0.21852208 & 0.99988895 \\
\end{array}
\]
Elapsed time: 0.00056 (seconds)

Begin Newton-Kantorovich iterations
\[
\begin{array}{ccc}
  nwt & tol \\
  1 & 1.03744352e-02 \\
  2 & 4.40564315e-04 \\
  3 & 8.45941486e-07 \\
  4 & 3.63797881e-12 \\
\end{array}
\]
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 $\beta$ close to 1
- Relative tolerance $tol_{k+1}/tol_k$ approach $\beta$

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:
  \[
  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
  \\
  \\
  Newton will immediately “strip away” the irrelevant constant $k$
  \\
  Switch to Newton whenever $tol_1/tol_0$ is sufficiently close to $\beta$
STEP 3: Recenter to ensure numerical stability

Logit formulas must be reentered.

\[ P_i = \frac{\exp(V_i)}{\sum_{j \in D(y)} \exp(V_j)} = \frac{\exp(V_i - V_0)}{\sum_{j \in D(y)} \exp(V_j - V_0)} \]

and “log-sum” must be recenteret too

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

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 $\Gamma'$

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

- In terms of its finite-dimensional approximation, $\Gamma'_\theta$ 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 $EV_\theta$

- $\Gamma'_\theta$ is simply $\beta$ 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. \( \theta \)
- 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

\[ \theta^{g+1} = \theta^g - \lambda \left( \sum_i H_i(\theta^g) \right)^{-1} \sum_i s_i(\theta^g) \]

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

\[ \theta^{g+1} = \theta^g + \lambda \left( \sum_i s_i s_i' \right)^{-1} \sum_i s_i \]

- Why is this valid? Information identity:

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

(only valid for MLE and CMLE)
**STEP 6: BHHH**

Some times linesearch may not help Newtons Method

Non-concave likelihood

Convex region:
- Newton–Raphson moves in the opposite direction of the gradient
- NR moves DOWNHILL
  (Wrong, wrong, way)

Concave region:
- Newton–Raphson moves in the same direction of the gradient
- NR moves UPHILL

BHHH: Still good

$\theta$
STEP 6: BHHH

Advantages
- $\sum_is_is_i'$ is always positive definite
  - i.e. it always moves uphill for $\lambda$ 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.
“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!

$\beta=0.9999$

*** Convergence Achieved ***

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

<table>
<thead>
<tr>
<th>Param.</th>
<th>Estimates</th>
<th>s.e.</th>
<th>t-stat</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC</td>
<td>11.1525</td>
<td>0.9167</td>
<td>12.1655</td>
</tr>
<tr>
<td>c</td>
<td>2.3298</td>
<td>0.3288</td>
<td>7.0856</td>
</tr>
</tbody>
</table>

Time to convergence is 0 min and 0.07 seconds
MPEC versus NFXP-NK: sample size 6,000

<table>
<thead>
<tr>
<th>β</th>
<th>Converged (out of 1250)</th>
<th>CPU Time (in sec.)</th>
<th># of Major Iter.</th>
<th># of Func. Eval.</th>
<th># of Bellm. Iter.</th>
<th># of N-K Iter.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.975</td>
<td>1247</td>
<td>1.677</td>
<td>60.9</td>
<td>69.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.985</td>
<td>1249</td>
<td>1.648</td>
<td>62.9</td>
<td>70.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.995</td>
<td>1249</td>
<td>1.783</td>
<td>67.4</td>
<td>74.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.999</td>
<td>1249</td>
<td>1.849</td>
<td>72.2</td>
<td>78.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.9995</td>
<td>1250</td>
<td>1.967</td>
<td>74.8</td>
<td>81.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.9999</td>
<td>1248</td>
<td>2.117</td>
<td>79.7</td>
<td>87.5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### MPEC-Matlab

<table>
<thead>
<tr>
<th>β</th>
<th>Converged (out of 1250)</th>
<th>CPU Time (in sec.)</th>
<th># of Major Iter.</th>
<th># of Func. Eval.</th>
<th># of Bellm. Iter.</th>
<th># of N-K Iter.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.975</td>
<td>1246</td>
<td>0.054</td>
<td>9.3</td>
<td>12.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.985</td>
<td>1217</td>
<td>0.078</td>
<td>16.1</td>
<td>44.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.995</td>
<td>1206</td>
<td>0.080</td>
<td>17.4</td>
<td>49.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.999</td>
<td>1248</td>
<td>0.055</td>
<td>9.9</td>
<td>12.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.9995</td>
<td>1250</td>
<td>0.056</td>
<td>9.9</td>
<td>11.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.9999</td>
<td>1249</td>
<td>0.060</td>
<td>11.1</td>
<td>13.1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### MPEC-AMPL

<table>
<thead>
<tr>
<th>β</th>
<th>Converged (out of 1250)</th>
<th>CPU Time (in sec.)</th>
<th># of Major Iter.</th>
<th># of Func. Eval.</th>
<th># of Bellm. Iter.</th>
<th># of N-K Iter.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.975</td>
<td>1250</td>
<td>0.068</td>
<td>11.4</td>
<td>13.9</td>
<td>155.7</td>
<td>51.3</td>
</tr>
<tr>
<td>0.985</td>
<td>1250</td>
<td>0.066</td>
<td>10.5</td>
<td>12.9</td>
<td>146.7</td>
<td>50.9</td>
</tr>
<tr>
<td>0.995</td>
<td>1250</td>
<td>0.069</td>
<td>9.9</td>
<td>12.6</td>
<td>145.5</td>
<td>55.1</td>
</tr>
<tr>
<td>0.999</td>
<td>1250</td>
<td>0.069</td>
<td>9.4</td>
<td>12.5</td>
<td>141.9</td>
<td>57.1</td>
</tr>
<tr>
<td>0.9995</td>
<td>1250</td>
<td>0.078</td>
<td>9.4</td>
<td>12.5</td>
<td>142.6</td>
<td>57.5</td>
</tr>
<tr>
<td>0.9999</td>
<td>1250</td>
<td>0.070</td>
<td>9.4</td>
<td>12.6</td>
<td>142.4</td>
<td>57.7</td>
</tr>
</tbody>
</table>
### MPEC versus NFXP-NK: sample size 60,000

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

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