Nonlinear Optimization/MPEC Approach

Chris Conlon
thanks to Che-Lin Su

Grad IO III

January 21, 2012

Unconstrained Optimization

Basic idea in many estimation problems is to use Newton-type methods to solve FOCs of equilibrium or estimating equations

$$\min f(x) : x \in \mathbb{R}^n$$

- $f : R^n \to R, c : R^n \to R^m$ smooth (typically $\mathcal{C}^2$)
- $x \in \mathbb{R}^n$ finite dimensional (may be large)

Want to find a local minimizer

$$\nabla f(x^*) = 0$$

Optimization Algorithms generate a sequence $x^{(k)}$ such that the gradient test

$$||\nabla f(x^{(k)})|| \leq \tau$$

is satisfied for some tolerance $\tau = 1e - 6$ or so. Warning!

General NLP Problem

A Nonlinear Programming (NLP) problem is defined by:

$$\begin{cases} \min_x f(x) \quad \text{objective} \\ \text{subject to} \quad c(x) = 0 \quad \text{constraints} \\ x \geq 0 \quad \text{variables} \end{cases}$$

Typical assumptions

- $f : R^n \to R, c : R^n \to R^m$ smooth (typically $\mathcal{C}^2$)
- $x \in R^n$ finite dimensional (perhaps large)
- more general $l \leq c(x) \leq u$ is possible

Optimality Conditions for NLP

Constraint Qualifications (CQ)

Linearization of $c(x) = 0$ characterize all feasible permutations, x^* local minimizer & CQ holds $\exists$ multipliers λ^*, γ^*:

$$\begin{aligned} \nabla f(x^*) - \nabla c(x^*)\lambda^* - \gamma^* &= 0 \\ c(x^*) = &= 0 \\ X^*\lambda^* &= 0 \\ x^* \geq 0, \gamma^* &\geq 0 \end{aligned}$$

Where $X^* = diag(x^*)$, thus $X^*\lambda^* = 0 \Leftrightarrow x_i^*\gamma_i^* = 0$

NLP Solvers

$F(w) = 0$ where $w = (x, \lambda, \gamma)$ with $x \geq 0, z \geq 0$. Optimization Algorithms generate a sequence $w^{(k)}$ such that the gradient test

$$||\nabla f(w^{(k)})|| \leq \tau$$

is satisfied for some tolerance $\tau = 1e - 6$ or so. (Same warning).

Optimization

"Folk Theory" of Optimization in Economics

- Unconstrained Optimization is easier than Constrained Optimization
- More parameters are harder
- Quasi-Newton Methods are unreliable

Consequences of Folk Theory

- Rewrite all problems as unconstrained optimization
- Use fixed points and multi-step procedures to reduce parameter space
- Use Nelder-Mead/Simplex methods for optimization

Optimization

Thanks to recent advances in optimization:

More Accurate Description of Optimizaiton

1. *Shape* of the problem is what matters – convexity is really important
2. Constrained Problems are not much more difficult
3. More parameters can make the problem easier (or harder)

Consequences of State of the Art Optimization

- Tested stable Newton-routines are very reliable.
- Good Solvers handle 10,000+ parameters
- Computational burden are Jacobian and Hessian (and storage)

Recent Advances in Optimization Literature

Large Scale Algorithms

- Much focus has been on very large convex optimization problems – these have gotten really good.
- Most of these rely on first and second derivatives and quadratic approximations.
- Ways to do derivatives: analytic, numeric, symbolic and automatic (new!)
- Easy to solve 10,000+ parameter constrained problems often in less than 20 major iterations.
- Lots of industrial strength software packages.
- When in doubt express your problem as a convex one.
- Algorithm is polynomial $\approx O(k^3)$

Convexity

An optimization problem is convex if

$$\min_{x} f(\mathbf{x}) \quad s.t. \quad h(\mathbf{x}) \leq 0 \quad A\mathbf{x} = 0$$

- $f(\mathbf{x}), h(\mathbf{x})$ are convex (PSD second derivative matrix)
- Equality Constraint is affine

Some helpful identities about convexity

- Compositions and sums of convex functions are convex.
- Norms $||$ are convex, max is convex, log is convex
- $\log(\sum_{i=1}^{n} \exp(x_i))$ is convex.
- Fixed Points can introduce non-convexities.
- Globally convex problems have a unique optimum

Nested Logit Model

FIML Nested Logit Model is Non-Convex

$$\min_{\theta} \sum_j q_j \ln P_j(\theta) \quad \text{s.t.} \quad P_j(\theta) = \frac{e^{x_j \beta/\lambda} (\sum_{k \in g_l} e^{x_j \beta/\lambda})^{\lambda-1}}{\sum_{\forall l'} (\sum_{k \in g_l'} e^{x_j \beta/\lambda})^{\lambda}}$$

This is a pain to show but the problem is with the cross term $\frac{\partial^2 P_j}{\partial \beta \partial \lambda}$ because $\exp[x_j \beta/\lambda]$ is not convex.

A Simple Substitution Saves the Day: let $\gamma = \beta/\lambda$

$$\min_{\theta} \sum_j q_j \ln P_j(\theta) \quad \text{s.t.} \quad P_j(\theta) = \frac{e^{x_j \gamma} (\sum_{k \in g_l} e^{x_j \gamma})^{\lambda-1}}{\sum_{\forall l'} (\sum_{k \in g_l'} e^{x_j \gamma})^{\lambda}}$$

This is much better behaved and easier to optimize.

Nested Logit Model (Conlon Mortimer 2009)

	Original[1]	**Substitution**[2]	**No Derivatives**[3]
Parameters	49	49	49
Nonlinear λ	5	5	5
Neg LL	2.279448	2.279448	2.27972
Iterations	197	146	352
Time	59.0 s	10.7 s	192s

Discuss Simplex, Sparsity.

Extremum Estimators

Often faced with extremum estimator problems in econometrics (ML, GMM, MD, etc.) that look like:

$$\hat{\theta} = \arg\max_{\theta} Q_n(\theta), \quad \theta \in \Theta \tag{1}$$

Many economic problems contain constraints, such as: market clearing (supply equals demand), consumer's consume their entire budget set, or firm's first order conditions are satisfied. A natural way to represent these problems is as constrained optimization.

Constrained Problems

MPEC

$$\hat{\theta} = \arg\max_{\theta, P} Q_n(\theta, P), \quad \text{s .t.} \quad \Psi(P, \theta) = 0, \quad \theta \in \Theta$$

Fixed Point / Implicit Solution

In much of the literature the tradition has been to express the solutions $\Psi(P, \theta) = 0$ implicitly as $P(\theta)$:

$$\hat{\theta} = \arg\max_{\theta} Q_n(\theta, P(\theta)), \quad \theta \in \Theta$$

Constrained Problems

MPEC

$$\hat{\theta} = \arg\max_{\theta, P} Q_n(\theta, P), \quad \text{s .t.} \quad \Psi(P, \theta) = 0, \quad \theta \in \Theta$$

Fixed Point / Implicit Solution

In much of the literature the tradition has been to express the solutions $\Psi(P, \theta) = 0$ implicitly as $P(\theta)$:

$$\hat{\theta} = \arg\max_{\theta} Q_n(\theta, P(\theta)), \quad \theta \in \Theta$$

Rust Problem

- Bus repairman sees mileage x_t at time t since last overhaul
- Repairman chooses between overhaul and normal maintenance

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

- Repairman solves DP:

$$V_\theta(x_t) = \sum_{f_t, f_{t+1}, \ldots} E \left\{ \sum_{j=t}^{\infty} \beta^{j-t} [u(x_j, f_j, \theta) + \varepsilon_j(f_j)] | x_t \right\}$$

- Econometrician
  - Observes mileage x_t and decision d_t but not cost.
  - Assumes extreme value distribution for $\varepsilon_t(d_t)$
- Structural parameters to be estimated $\theta = (\theta^c, RC, \theta^p)$.
  - Coefficients of cost function $c(x, \theta^c) = \theta_1^c x + \theta_2^c x^2$
  - Overhaul cost RC
  - Transition probabilities in mileages $p(x_{t+1} | x_t, d_t, \theta^p)$

Rust Problem

- Data: time series $(x_t, d_t)_{t=1}^T$
- Likelihood function

$$
\begin{aligned}
\mathcal{L}(\theta) &= \prod_{t=2}^{T} P(d_t|x_t, \theta^c, RC) p(x_t|x_{t-1}, d_{t-1}, \theta^p) \\
\text{with } P(d|x, \theta^c, RC) &= \frac{\exp[u(x, d, \theta^c, RC) + \beta EV_\theta(x, d)]}{\sum_{d' \in \{0,1\}} \exp[u(x, d', \theta^c, RC) + \beta EV_\theta(x', d)]} \\
EV_\theta(x, d) &= T_\theta(EV_\theta)(x, d)
\end{aligned}
$$

$$
\equiv \int_{x'=0}^{\infty} \log \left[\sum_{d' \in \{0,1\}} \exp[u(x, d', \theta^c, RC) + \beta EV_\theta(x', d)] \right] p(dx'|x, d, \theta^p)
$$

Rust Problem

- Outer Loop: Solve Likelihood

$$\max_{\theta \geq 0} \mathcal{L}(\theta) = \prod_{t=2}^{T} P(d_t | x_t, \theta^c, RC) p(x_t | x_{t-1}, d_{t-1}, \theta^p)$$

- Convergence test: $\|\nabla_\theta \mathcal{L}(\theta)\| \leq \epsilon_{out}$
- Inner Loop: Compute expected value function EV_θ for a given θ
- EV_θ is the implicit expected value function defined by the Bellman equation or the fixed point function

$$EV_\theta = T_\theta(EV_\theta)$$

- Convergence test: $\|EV_\theta^{(k+1)} - EV_\theta^{(k)}\| \leq \epsilon_{in}$
- Start with contraction iterations and polish with Newton Steps

NFXP Concerns

- Inner-loop error propagates into outer-loop function and derivatives
- NFXP needs to solve inner-loop exactly each stage of parameter search
  - to accurately compute the search direction for the outer loop
  - to accurately evaluate derivatives for the outer loop
  - for outer loop to converge!
- Stopping rules: choosing inner-loop and outer-loop tolerance
  - inner loop can be slow: contraction mapping is linearly convergent
  - tempting to loosen inner loop tolerance ϵ_{in} (such as $1e-6$ or larger!).
  - Outer loop may not converge with loose inner loop tolerance.
    - check solver output message
    - tempting to loosen outer loop tolerance ϵ_{out} to promote convergence ($1e-3$ or larger!).

Stopping Rules

- $\mathcal{L}(EV(\theta, \epsilon_{in}), \theta)$ the programmed outer loop objective function
- L: the Lipschitz constant (like modulus) of inner-loop contraction mapping
- Analytic derivatives $\nabla_\theta \mathcal{L}(EV(\theta, \epsilon_{in}), \theta)$ is provided: $\epsilon_{out} = O(\frac{L}{1-L}\epsilon_{in})$
- Finite-difference derivatives are used: $\epsilon_{out} = O(\sqrt{\frac{L}{1-L}\epsilon_{in}})$

Stopping Rules

- Form the augmented likelihood function for data $X = (x_t, d_t)_{t=1}^{T}$

$$\begin{aligned} \mathcal{L}(EV, \theta; X) &= \prod_{t=2}^{T} P(d_t | x_t, \theta^c, RC) p(x_t | x_{t-1}, d_{t-1}, \theta^p) \\ \text{with } P(d|x, \theta^c, RC) &= \frac{\exp[u(x, d, \theta^c, RC) + \beta EV(x, d)}{\sum_{d' \in \{0,1\}} \exp[u(x, d', \theta^c, RC) + \beta EV(x', d)} \end{aligned}$$

- Rationality and Bellman equation imposes a relationship between θ and EV

$$EV = T(EV, \theta)$$

- Solve constrained optimization problem

$$\max_{(\theta, EV)} \mathcal{L}(EV, \theta; X)$$
$$\text{subject to } EV = T(EV, \theta)$$

Results

β	Imple.	Parameters						MSE
		RC	θ_{11}	θ_{31}	θ_{32}	θ_{33}	θ_{34}	
	true	11.726	2.457	0.0937	0.4475	0.4459	0.0127	
0.975	MPEC1	12.212	2.607	0.0943	0.4473	0.4454	0.0127	3.111
		(1.613)	(0.500)	(0.0036)	(0.0057)	(0.0060)	(0.0015)	–
	MPEC2	12.212	2.607	0.0943	0.4473	0.4454	0.0127	3.111
		(1.613)	(0.500)	(0.0036)	(0.0057)	(0.0060)	(0.0015)	–
	NFXP	12.213	2.606	0.0943	0.4473	0.4445	0.0127	3.123
		(1.617)	(0.500)	(0.0036)	(0.0057)	(0.0060)	(0.0015)	–
0.980	MPEC1	12.134	2.578	0.0943	0.4473	0.4455	0.0127	2.857
		(1.570)	(0.458)	(0.0037)	(0.0057)	(0.0060)	(0.0015)	–
	MPEC2	12.134	2.578	0.0943	0.4473	0.4455	0.0127	2.857
		(1.570)	(0.458)	(0.0037)	(0.0057)	(0.0060)	(0.0015)	–
	NFXP	12.139	2.579	0.0943	0.4473	0.4455	0.0127	2.866
		(1.571)	(0.459)	(0.0037)	(0.0057)	(0.0060)	(0.0015)	–

Results

β	Imple.	Parameters						MSE
		RC	θ_{11}	θ_{31}	θ_{32}	θ_{33}	θ_{34}	
	true	11.726	2.457	0.0937	0.4475	0.4459	0.0127	
0.985	MPEC1	12.013	2.541	0.0943	0.4473	0.4455	0.0127	2.140
		(1.371)	(0.413)	(0.0037)	(0.0057)	(0.0060)	(0.0015)	–
	MPEC2	12.013	2.541	0.0943	0.4473	0.4455	0.0127	2.140
		(1.371)	(0.413)	(0.0037)	(0.0057)	(0.0060)	(0.0015)	–
	NFXP	12.021	2.544	0.0943	0.4473	0.4455	0.0127	2.136
		(1.368)	(0.411)	(0.0037)	(0.0057)	(0.0060)	(0.0015)	–
0.990	MPEC1	11.830	2.486	0.0943	0.4473	0.4455	0.0127	1.880
		(1.305)	(0.407)	(0.0036)	(0.0057)	(0.0060)	(0.0015)	–
	MPEC2	11.830	2.486	0.0943	0.4473	0.4455	0.0127	1.880
		(1.305)	(0.407)	(0.0036)	(0.0057)	(0.0060)	(0.0015)	–
	NFXP	11.830	2.486	0.0943	0.4473	0.4455	0.0127	1.880
		(1.305)	(0.407)	(0.0036)	(0.0057)	(0.0060)	(0.0015)	–

Results

β	Imple.	Parameters						MSE
		RC	θ_{11}	θ_{31}	θ_{32}	θ_{33}	θ_{34}	
	true	11.726	2.457	0.0937	0.4475	0.4459	0.0127	
0.995	MPEC1	11.819	2.492	0.0942	0.4473	0.4455	0.0127	1.892
		(1.308)	(0.414)	(0.0036)	(0.0057)	(0.0060)	(0.0015)	–
	MPEC2	11.819	2.492	0.0942	0.4473	0.4455	0.0127	1.892
		(1.308)	(0.414)	(0.0036)	(0.0057)	(0.0060)	(0.0015)	–
	NFXP	11.819	2.492	0.0942	0.4473	0.4455	0.0127	1.892
		(1.308)	(0.414)	(0.0036)	(0.0057)	(0.0060)	(0.0015)	–

Results

β	Imple.	Runs Conv.	CPU Time (in sec.)	# of Major Iter.	# of Func. Eval.	# of Contrac. Mapping Iter.
0.975	MPEC1	1240	0.13	12.8	17.6	–
	MPEC2	1247	7.9	53.0	62.0	–
	NFXP	998	24.6	55.9	189.4	$1.348e+5$
0.980	MPEC1	1236	0.15	14.5	21.8	–
	MPEC2	1241	8.1	57.4	70.6	–
	NFXP	1000	27.9	55.0	183.8	$1.625e+5$
0.985	MPEC1	1235	0.13	13.2	19.7	–
	MPEC2	1250	7.5	55.0	62.3	–
	NFXP	952	42.2	61.7	227.3	$2.658e+5$
0.990	MPEC1	1161	0.19	18.3	42.2	–
	MPEC2	1248	7.5	56.5	65.8	–
	NFXP	935	70.1	66.9	253.8	$4.524e+5$
0.995	MPEC1	965	0.14	13.4	21.3	–
	MPEC2	1246	7.9	59.6	70.7	–
	NFXP	950	111.6	58.8	214.7	$7.485e+5$

Results

```
KNITRO 5.2.0: alg=1
opttol=1.0e-6
feastol=1.0e-6

Problem Characteristics
-----------------------
Objective goal:  Maximize
Number of variables:                    207
    bounded below:                        6
    bounded above:                      201
    bounded below and above:              0
    fixed:                                0
    free:                                 0
Number of constraints:                  202
    linear equalities:                    1
    nonlinear equalities:               201
    linear inequalities:                  0
    nonlinear inequalities:               0
    range:                                0
Number of nonzeros in Jacobian:        2785
Number of nonzeros in Hessian:         1620
```

Results

```
Final Statistics
-----------------
Final objective value               =  -2.35221126396447e+03
Final feasibility error (abs / rel) =   1.33e-15 / 1.33e-15
Final optimality error  (abs / rel) =   1.00e-08 / 6.71e-10
# of iterations                     =         12
# of CG iterations                  =          0
# of function evaluations           =         13
# of gradient evaluations           =         13
# of Hessian evaluations            =         12
Total program time (secs)           =       0.10326 (     0.097 CPU time)
Time spent in evaluations (secs)    =       0.05323

=================================================================

KNITRO 5.2.0: Locally optimal solution.
objective -2352.211264; feasibility error 1.33e-15
12 major iterations; 13 function evaluations
```

BLP Demand Example

BLP 1995

The estimator solves the following mathematical program:

$$
\begin{aligned}
\min_{\theta_2} \quad & g(\xi(\theta_2))' W g(\xi(\theta_2)) \quad \text{s.t.} \\
g(\xi(\theta_2)) &= \frac{1}{N} \sum_{\forall j,t} \xi_{jt}(\theta_2)' z_{jt} \\
\xi_{jt}(\theta_2) &= \delta_j(\theta_2) - x_{jt}\beta - \alpha p_{jt} \\
s_{jt}(\delta(\theta_2), \theta_2) &= \int \frac{\exp[\delta_j(\theta_2) + \mu_{ij}]}{1 + \sum_k \exp[\delta_j(\theta_2) + \mu_{ik}]} f(\mu|\theta_2) \\
\log(S_{jt}) &= \log(s_{jt}(\delta(\theta_2), \theta_2)) \quad \forall j, t
\end{aligned}
$$

BLP Algorithm

The estimation algorithm is generally as follows:

1. Guess a value of nonlinear parameters θ_2
2. Compute $s_{jt}(\delta, \theta_2)$ via integration
3. Iterate on $\delta_{jt}^{h+1} = \delta_{jt}^{h} + \log(S_{jt}) - \log(s_{jt}(\delta^h, \theta_2))$ to find the δ that satisfies the share equation
4. IV Regression δ on observable X and instruments Z to get residual ξ.
5. Use ξ to construct $g(\xi(\theta_2))$.
6. Possibly construct other errors/instruments from supply side.
7. Construct GMM Objective

The idea is that $\delta(\theta_2)$ is an implicit function of the nonlinear parameters θ_2. And for each guess we find that implicit solution for reduce the parameter space of the problem.

Dube Fox Su 2009

BLP-MPEC

The estimator solves the following mathematical program:

$$\begin{aligned} \min_{\sigma,\alpha,\beta,\xi} \quad & g(\xi)' W g(\xi) \quad \text{s.t.} \\ g(\xi) &= \frac{1}{N} \sum_{\forall j,t} \xi_{jt}' z_{jt} \\ s_{jt}(\sigma,\alpha,\beta,\xi) &= \sum_i w_i \frac{\exp[x_{jt}\beta + \xi_{jt} - \alpha p_{jt} + \sum_l \nu_{il} x_{jt}^l \sigma_l]}{1 + \sum_k \exp[x_{kt}\beta + \xi_{kt} - \alpha p_{kt} + \sum_l \nu_{il} x_{kt}^l \sigma_l]} \\ \log(S_{jt}) &= \log s_{jt}(\sigma,\alpha,\beta,\xi) \quad \forall j,t \end{aligned}$$

- Expand the parameter space of the nonlinear search to include α, β, ξ
- Don't have to solve for ξ except at the end.
- No implicit functions of θ_2
- Sparsity!