High Performance Quadrature Rules

How Numerical Integration Affects a Popular Model of Product Differentiation

Benjamin S. Skrainka (UCL)
Kenneth L. Judd (Hoover)

October 25, 2010

The Big Picture

Accurate and efficient numerical approximation of multi-dimensional integrals is crucial for modern economic research:

- Unobserved heterogeneity
- Uncertainty
- Incomplete information (expectations)
- Approximation of densities, functions, etc. using basis functions: e.g., sieve estimation

Need to approximate integrals quickly and accurately!

Research Objectives

Our paper shows that polynomial-based quadrature rules are vastly superior to **M**onte **C**arlo methods for multi-dimensional numerical integration:

- Polynomial-based methods are both more accurate and more efficient
- Using MC methods incorrectly compromises numerical results and may mask identification problems
- MC causes many problems in **B**erry, **L**evinsohn, & **P**akes (1995, 2004)-style models, including incorrect point estimates, multiple local optima, and non-convergence Berry's mapping

Results currently based on five simulated data sets.

A Bit of Literature

Some integration literature:

- Stroud (1971)
- Genz (1993)
- Cools (1997, 2002, 2003)
- Judd (1998)
- Heiss & Winschel (2008)

Some discrete choice literature:

- Berry, Levinsohn, & Pakes (1995, 2004); Nevo (2000a, 2000b, 2001)
- McFadden & Train (2000)
- Train (2009)

Just a Bit More Literature

There is a growing, but young literature on identification:

- Chiou & Walker (2007); Walker (2002); Walker, Ben-Akiva, & Bolduc (2004)
- Berry & Haile (2009)

Numerical literature focuses on optimization:

- Dubé, Fox, & Su (2009); Su & Judd (2009)

Roadmap

The plan for this talk is:

1. Review quadrature methods for multi-dimensional numerical integration
2. Example: mixed logit
3. BLP model
4. Results
5. Future Research
6. Conclusions

Numerical Integration

Integration Basics

Want to approximate some (multidimensional) integral

$$I[f] := \int_{\Omega} f(x)\, w(x)\, dx,\ \Omega \subset \mathbb{R}^d,\ w(x) \geq 0\, \forall x \in \Omega$$

as

$$Q[f] := \sum_{j=1}^{R} w_j f(y_j),\ y_j \in \Omega$$

- The crucial issue is how to choose the nodes and weights, (w_j, y_j)
- Ideally, a rule should have $\lim_{R\to\infty} Q^R[f] = I[f]$

Example: Mixed Logit

Conditional shares with linear utility & Type I Extreme value:

$$s_{ij}\left(\alpha_i\right) = \frac{\exp\left[-\alpha_i \log p_j + z_j^T \beta\right]}{\sum_k \exp\left[-\alpha_i \log p_k + z_k^T \beta\right]}$$

Computed market shares are then:

$$\begin{aligned} s_j &= \int_{-\infty}^{\infty} s_{ij}\left(\alpha_i\right) \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}\left[\alpha_i - \alpha\right]^2\right) d\alpha_i \\ &= \frac{1}{\sqrt{\pi}} \int_{-\infty}^{\infty} s_{ij}\left(\sqrt{2}\sigma u\right) \exp\left(-u^2\right) du \\ &\approx \frac{1}{\sqrt{\pi}} \sum_k w_k s_{ij}\left(\sqrt{2}\sigma u_k\right). \end{aligned}$$

Note: mixed logit $\leftrightarrow$ random coefficients

Monte Carlo Methods

There are several MC tools – all based on number theory:

- **p**seudo-**MC**:
  - Draw R nodes, y_i, from weight function $w(x)$
  - Set weights to $1/R$
  - Converges as $R^{-1/2}$
  - Warning: must increase draws 100x for each extra decimal point of accuracy!
- **q**uasi-**MC**:
  - Use number theory to choose nodes which have better properties, e.g. equidistribution, low discrepancy
  - Set weights to $1/R$
  - Converges as $R^{-d/2}$
- For high dimensions, MC is the only option....

Warning!

Beware of 'random numbers':

- 'Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin.' – von Neumann
- Computer generates *pseudo*-random numbers, which are not truly random!
- Make sure your proof works with $Q[f]$ and not just for $I[f]$!
- Different quasi-MC algorithms work better for different kinds of problems so it pays to experiment

Polynomial-based Rules

Polynomial based rules are usually much more efficient:

- *Exact* for all polynomial integrands $f(x)$ less than or equal to some degree, i.e. $I[f] = Q^R[f]$
- Example: in one dimension, a Gaussian rule with R nodes is exact for all polynomials of degree less than or equal to $2R-1$
- $\lim_{R\to\infty} Q^R[f] = I[f] \;\; \forall f \in C^1$

  $\Rightarrow$ very accurate for any smooth function which is well approximated by a polynomial

  $\Rightarrow$ requires many fewer nodes than MC rules
- Easier to use than most economists think: just call the right library function to generate nodes and weights instead of `randn()`

One-dimensional Polynomial-based Methods

Domain and weight function determine which nodes are clever:

- In one dimension, use Gaussian rule
- Choose rule based on domain and weight function:

Rule	$w(x)$	Ω
Gauss-Hermite	$\exp\left(-x^2\right)$	$\mathbb{R}$
Gauss-Chebyshev	$\left(1-x^2\right)^{-1/2}$	$[-1,1]$
Gauss-Legendre	1	$[-1,1]$
Gauss-Laguerre	$\exp(-x)$	$[0,\infty]$

- The best method for one dimension unless your function is ugly

Example: Gauss-Hermite

For five nodes we have:

y_k	w_k
2.020182870456086e+00	1.995324205904591e-02
9.585724646138185e-01	3.936193231522410e-01
0	9.453087204829417e-01
-9.585724646138185e-01	3.936193231522410e-01
-2.020182870456086e+00	1.995324205904591e-02

- These nodes and weights will exactly integrate any polynomial of degree $2 \times 5 - 1 = 9$ or less!
- The moral: use a Gaussian rule if your function is well approximated by a polynomial.
- Computed using `gauher()` from Press et al. (1992)

Multidimensional Polynomial-based Methods

Gaussian Product rules:

- Form tensor product of all combinations of one dimensional nodes and weights
- I.e. all points on a lattice

Sparse Grids:

- Choose nodes on lattice more symmetrically
- Uses more nodes and weights than a monomial rule

Monomial Rules:

- Uses absolute minimum of nodes
- Look up data in a table e.g. Stroud (1971)
- Monomial: $x^{\mathbf{p}} \equiv \prod_j x_j^{p_j}$
  - where $\mathbf{p} = (p_1, p_2, \ldots, p_J)$
  - Degree is $\sum_j p_j$
- Exact for all monomials less than or equal to some degree

Sparse Grids

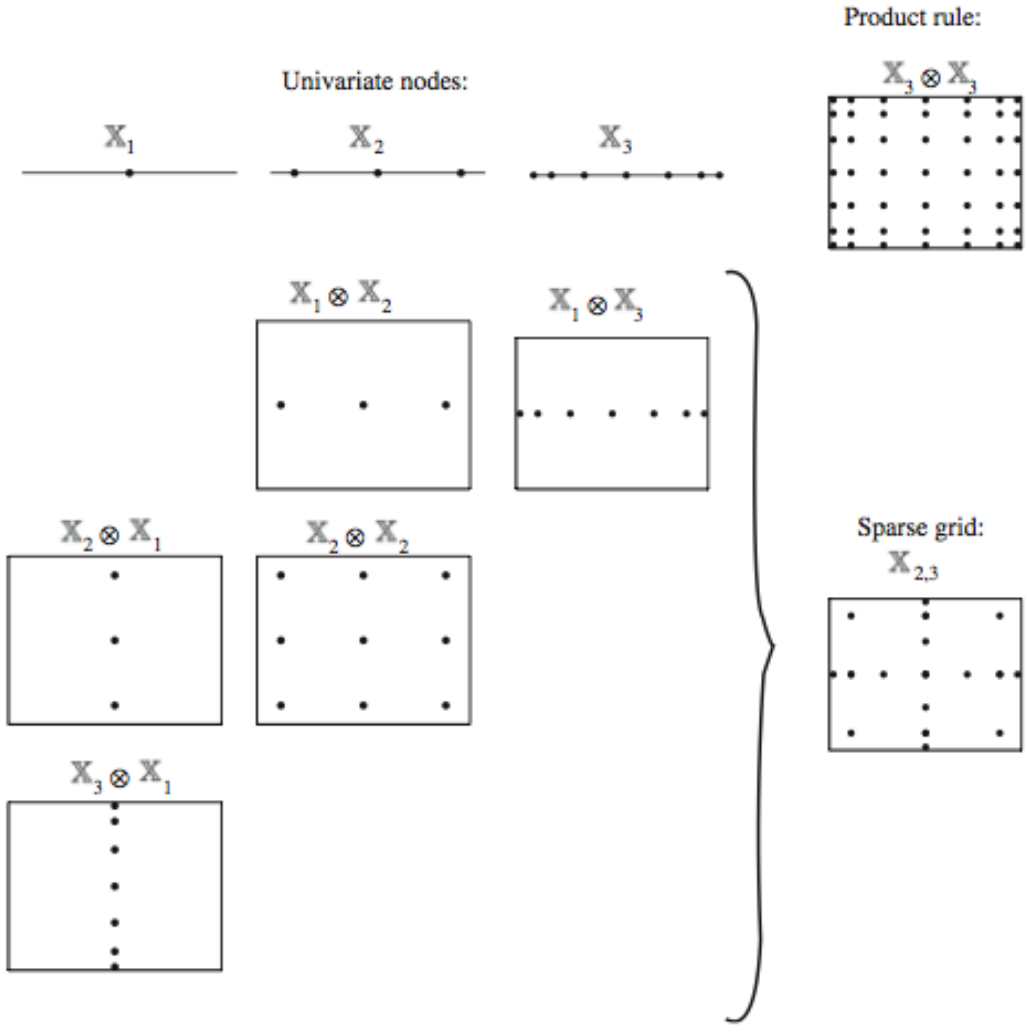


Fig. 1. Construction of the sparse grid in two dimensions.

Figure from Heiss & Winschel (2008)

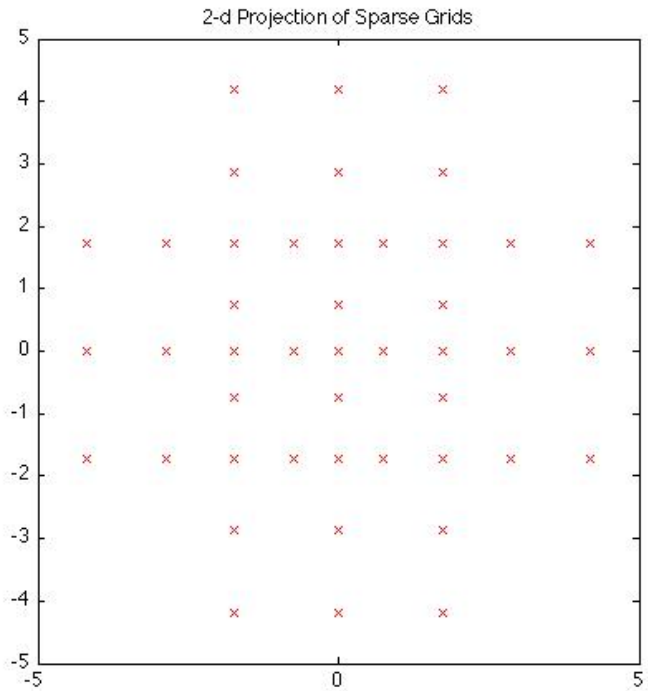
SGI Scaling

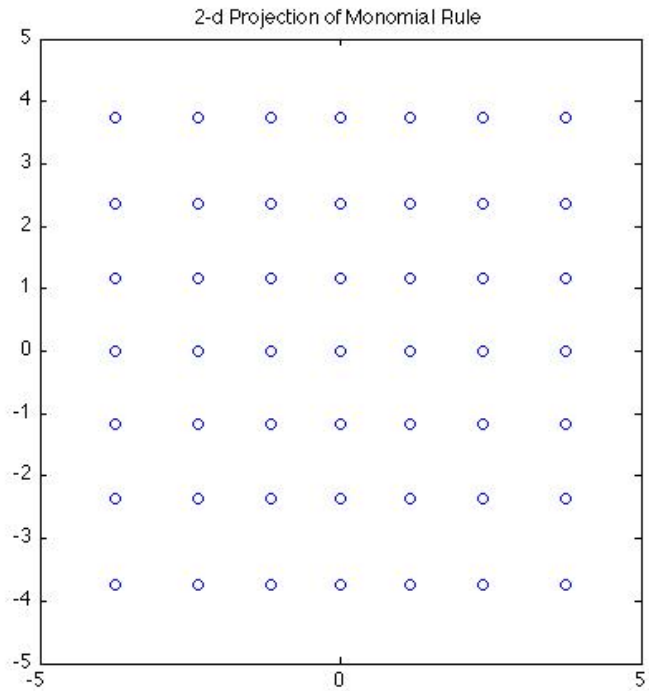
Table 1
Number of function evaluations

Dimensions	Product rule	Smolyak rule	
	Gaussian	Gaussian	KP
Level $k = 2$, polynomial exactness $= 3$			
$D = 1$	2	2	3
$D = 5$	32	11	11
$D = 10$	1,024	21	21
$D = 20$	1,048,576	41	41
Level $k = 3$, polynomial exactness $= 5$			
$D = 1$	3	3	3
$D = 5$	243	61	51
$D = 10$	59,049	221	201
$D = 20$	3,486,784,401	841	801
Level $k = 4$, polynomial exactness $= 7$			
$D = 1$	4	4	7
$D = 5$	1,024	241	151
$D = 10$	1,048,576	1,581	1,201
$D = 20$	1,099,511,627,776	11,561	10,001
Level $k = 5$, polynomial exactness $= 9$			
$D = 1$	5	5	7
$D = 5$	3,125	781	391
$D = 10$	9,765,625	8,761	5,281
$D = 20$	95,367,431,640,625	120,321	90,561

Table from Heiss & Winschel (2008)

SGI vs. Monomial Rule





Polynomial-based vs. MC Rules

The advantages of polynomial-based quadrature rules:

- Integrates any monomial below a certain degree exactly
- Integrates anything which can be approximated with monomials well
- More efficient
- More accurate

The advantages of MC rules:

- Easier to implement – but not by that much
- Only option for very high dimensions
- Handle weird domains
- May be better for some irregularities (jumps, kinks)

But, neither method handles highly irregular functions – e.g. lack of smoothness, extreme curvature

The BLP Model

BLP is a mixed logit with unobserved product-market heterogeneity for studying product differentiation:

- Individuals' random coefficients capture horizontal differences in taste
- Product-market shock, ξ_{jt}, captures vertical differences in quality
- Traditional estimation strategy uses nested-fixed point (Rust, 1987):
  - Outer loop uses $\hat{\xi}_{jt}^{n}$ to compute parameter estimates, $\hat{\theta}^{n}$, using GMM
  - Inner loop uses current parameter estimates $\hat{\theta}^{n}$ to estimate $\hat{\xi}_{jt}^{n+1}$ by equating observed (S_{jt}) and calculated market share integrals (s_{jt}): $S_{jt} = s_{jt}\left(\delta\left(\xi;\theta_1\right);\theta_2\right)$
  - Market share integrals computed via pMC (inaccurate)
  - (Best practice is to use MPEC (Su & Judd, 2009; Dubé, Fox, & Su, 2000))

The Outer Loop

The outer loop estimates the parameters via GMM:

$$\hat{\theta} = \arg\max_{\theta} \left(Z'\xi\right)' W \left(Z'\xi\right)$$

- Z instruments (e.g. Hausman, Pakes, or cost shifters)
- ξ unobserved product-market heterogeneity
- W Weighting matrix

The Inner Loop

The inner loop inverts the market share equations

$$S_{jt} = s_{jt}\left(\delta\left(\xi, p, x; \theta_1\right); \theta_2\right)$$

to obtain the mean utility δ_{jt} and, thus, ξ_{jt} which is needed for the outer loop

- Usual procedure is to use Berry's mapping, aka 'The Contraction Mapping'
- Shown to converge only for the case of exact integrals (Berry, 1994)

$$\exp\left(\delta_{jt}^{n+1}\right) = \exp\left(\delta_{jt}^{n}\right) \times S_{jt} / s_{jt}\left(\delta_{jt}^{n}; \theta_2\right)$$

Remark: The mapping is written this way for performance reasons – log is more expensive than exp

Remark: For simple logit, $\delta_{jt} = \log\left(s_{jt}\right) - \log\left(s_{0t}\right)$

Specification

Utility:

$$\begin{aligned} u_{ijt} &= V_{ijt} + \epsilon_{ijt} \\ V_{ijt} &= \alpha_i \left(y_i - p_{jt} \right) + x_{jt}' \beta_i + \xi_{jt} \end{aligned}$$

Distributions:

$$\begin{aligned} \epsilon_{ijt} &\sim \text{Type I Extreme Value} \\ \begin{pmatrix} \alpha_i \\ \beta_i \end{pmatrix} &= \begin{pmatrix} \bar{\alpha} \\ \bar{\beta} \end{pmatrix} + \Sigma \nu_i \\ \nu_i &\sim \text{N}(0, 1) \end{aligned}$$

By convention, $\theta = (\theta_1, \theta_2)$ where $\theta_1 = \left(\bar{\alpha}, \bar{\beta} \right)'$ and $\theta_2 = \text{vec}\left(\Sigma \right)$

Repackaging

It is customary to repackage the utility into constant and stochastic parts:

- Mean utility: $\delta_{jt}\left(\xi_{jt};\theta_1\right) = x_{jt}\bar{\beta} - \bar{\alpha}p_{jt} + \xi_{jt}$
- Preference shock: $\mu_{ijt}\left(\nu_i;\Sigma\right) = \left[-p_{jt}\ x_{jt}\right]\left(\Sigma\nu_i\right)$

Goal of estimation is to recover δ_{jt} to obtain ξ_{jt} for GMM outer loop. Once you have estimated the mean utility, e.g. with MLE, you can even regress $\hat{\delta}_{jt}$ on the covariates to estimate $\left(\bar{\alpha},\bar{\beta}\right)'$...

Calculated Share Integrals

A central part of BLP is the computation of the market share integrals:

- Conditional market shares:

$$s_{jt}\left(\delta\left(\xi;\theta_1\right)|\theta_2\right) = \frac{\exp\left[\delta_{jt}+\mu_{ijt}\left(\nu\right)\right]}{\sum\exp\left[\delta_{kt}+\mu_{ikt}\left(\nu\right)\right]}$$

- Unconditional market shares:

$$s_{jt}\left(\delta\left(\xi;\theta_1\right);\theta_2\right) = \int_{\mathbb{R}^{K+1}} \frac{\exp\left[\delta_{jt}+\mu_{ijt}\left(\nu\right)\right]}{\sum\exp\left[\delta_{kt}+\mu_{ikt}\left(\nu\right)\right]}\phi\left(\nu\right)d\nu$$

Assumptions

The model is depends on structural assumptions:

- Specification of indirect utility
- Distributional assumptions
- Identification
- No income effects
- Static
- Purchase at most one unit of good
- Share of outside good

Results

We investigate the performance of BLP using MC data:

- Currently five simulated data sets
- Simulate typical BLP data setup with five random coefficients:
  - Endogenous price & instruments
  - Diagonal Σ ('industry standard' assumption – facilitates identification)
- Code based on Dubé, Fox, & Su (2009)
- All pseudo-random numbers created with MATLAB^{TM}'s `rand()` and `randn()`
- Sparse grids generated using code from `www.sparse-grids.de`
- Gaussian nodes & weights from Press et al. (1992)
- Monomial rule 11-1 from Stroud (1971)

Computed Market Shares

We computed the 5-dimensional, market share integrals, s_{jt}:

- Compared different quadrature rules:
  - pMC
  - qMC (Niederreiter sequences)
  - GH product rule with $3^5, 5^5, 7^5, 9^5$ nodes
  - Sparse grids: exact for degree 11 monomials with 993 nodes.
  - Stroud monomial rule 11-1: exact for all degree 11 monomials with just 983 nodes!
- Evaluated at MPEC parameter estimates $\hat{\theta}^{MPEC}$ and at nine points drawn from $\mathrm{N}\left(\hat{\theta}^{MPEC}, 1/4 \left\|\hat{\theta}^{MPEC}\right\|\right)$
- $N = 100$ simulations for pMC and qMC
  - $R = \{100, 1000, 10000\}$ draws for pMC
  - $R = \{100, 1000, 10000\}$ with $10,000$ burn in for a Niederreiter qMC rule

Share Integral Approximation

Recall the original share integral:

$$s_{jt}\left(\delta\left(\xi;\theta_1\right);\theta_2\right) = \int_{\mathbb{R}^{K+1}} \frac{\exp\left[\delta_{jt}+\mu_{ijt}\left(\nu\right)\right]}{\sum\exp\left[\delta_{kt}+\mu_{ikt}\left(\nu\right)\right]}\phi\left(\nu\right)d\nu$$

We approximate it as:

$$s_{jt}\left(\delta\left(\xi;\theta_1\right);\theta_2\right) = \sum_{m\in\mathcal{N}} w_m \frac{\exp\left[\delta_{jt}+\mu_{ijt}\left(y_m\right)\right]}{\sum\exp\left[\delta_{kt}+\mu_{ikt}\left(y_m\right)\right]}$$

Remark: The kernel $\phi\left(\cdot\right)$ disappears because either we take pseudo-random draws from $\mathrm{N}\left(0,1\right)$, transform qMC numbers via $n=\Phi^{-1}\left(u\right)$, or use a polynomial rule with $w\left(x\right)=\exp\left(-x^2\right)$

Remark: A Cholesky decomposition of the variance is necessary to transform the nodes

Results

We now discuss the following results:

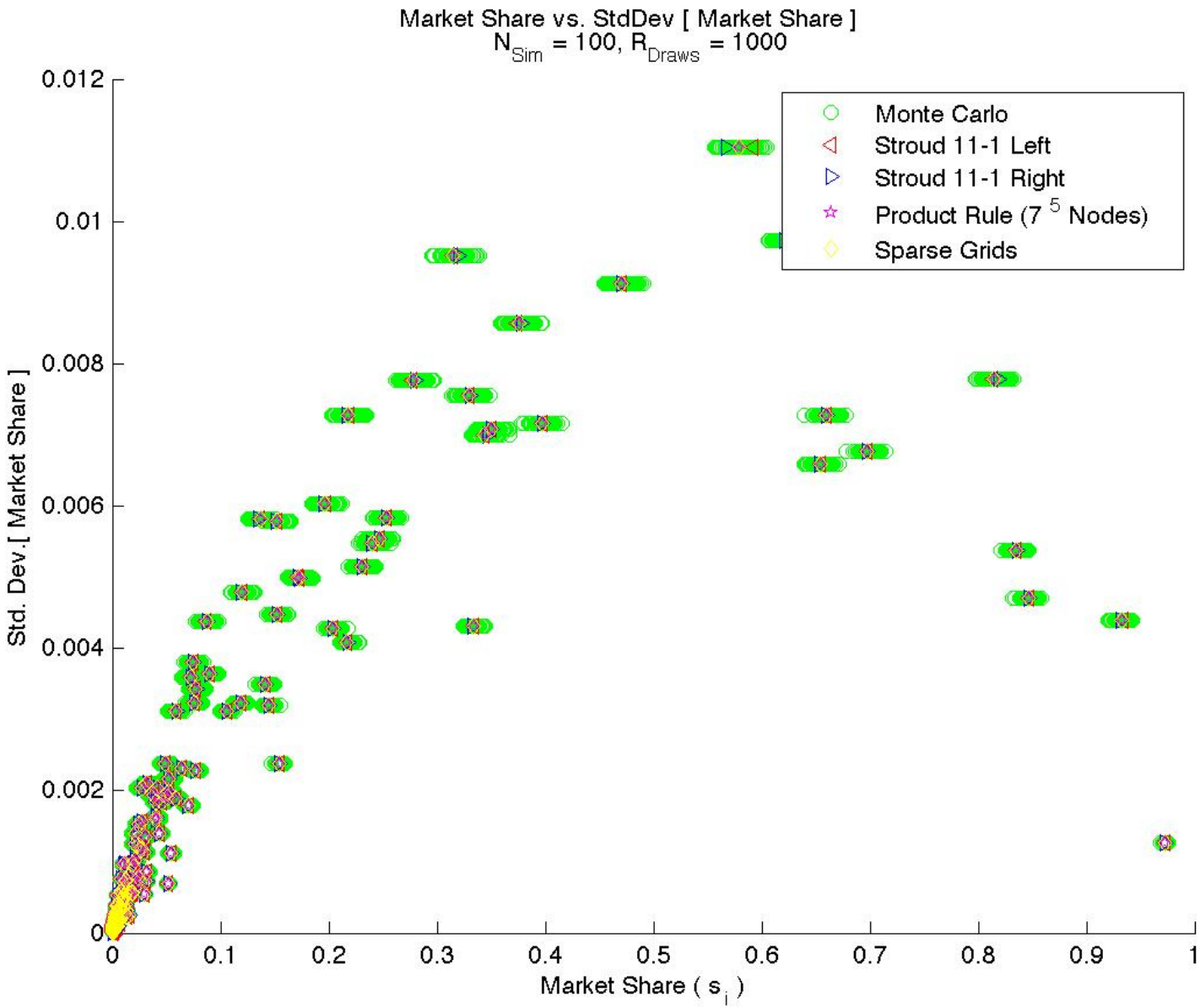
- Market share integrals
- Optimization
- Point estimates
- Berry's mapping

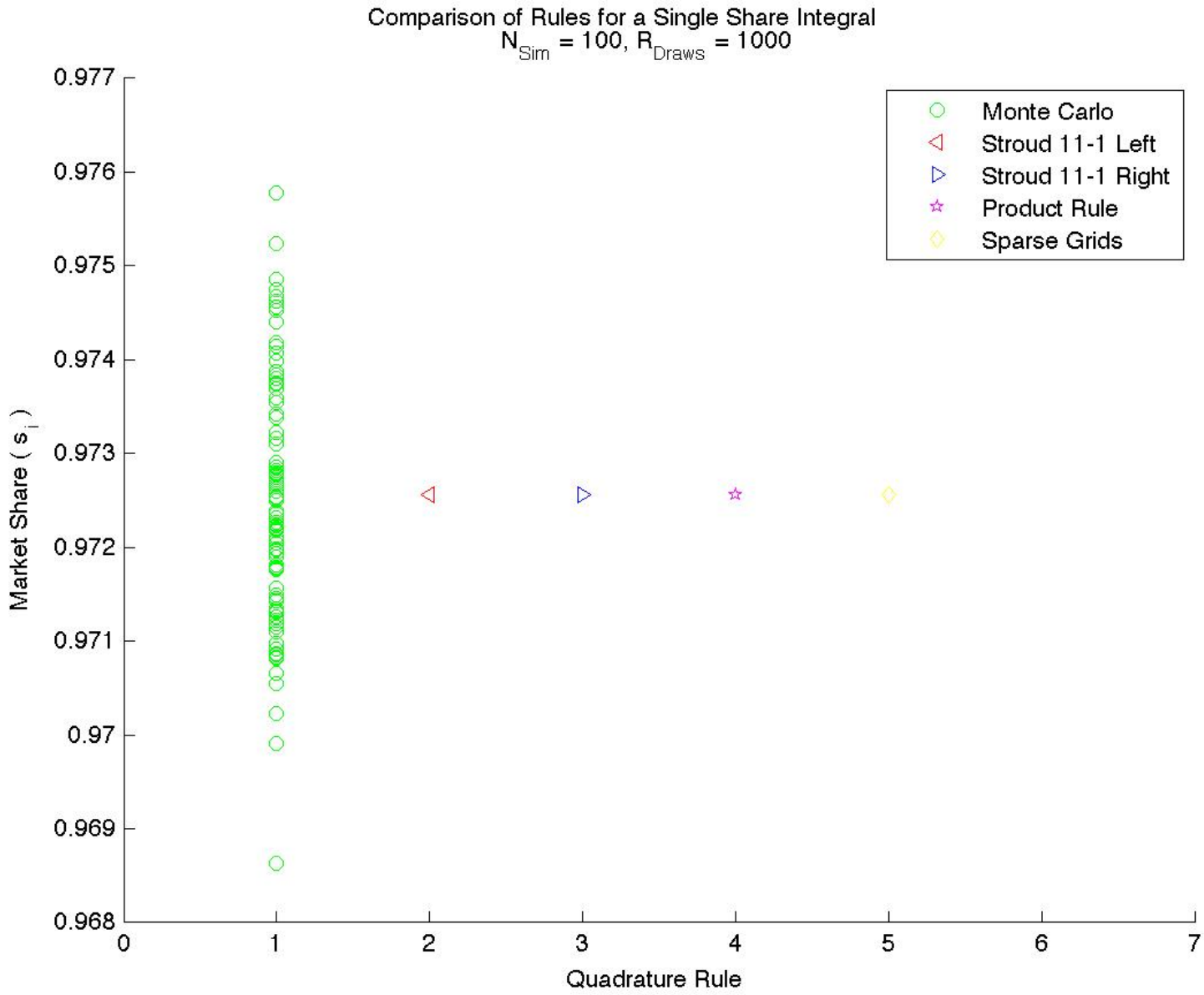
Results: Market Shares

Polynomial rules clearly superior to pMC:

- Polynomials approximate smooth functions like the logit well
- All polynomial-based rules clustered in center of MC cloud, usually at exactly the same point
- Polynomial-based rules close to mean of pMC simulations, as expected, because pMC is unbiased.
- Monomial rule and sparse grids much more efficient in terms of points than GH product rule or MC.
- qMC has lower variance than pMC

Key issue, however, is to approximate the gradient of the GMM objective function accurately!


Market Share vs. StdDev [Market Share]
N_Sim = 100, R_Draws = 1000
Monte Carlo
Stroud 11-1 Left
Stroud 11-1 Right
Product Rule (7 5 Nodes)
Sparse Grids
Std. Dev.[Market Share]
Market Share (s_i)
0.012
0.01
0.008
0.006
0.004
0.002
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Comparison of Rules for a Single Share Integral
N_Sim = 100, R_Draws = 1000
Market Share (s_i)
Quadrature Rule
0.977
0.976
0.975
0.974
0.973
0.972
0.971
0.97
0.969
0.968
0
1
2
3
4
5
6
7
Monte Carlo
Stroud 11-1 Left
Stroud 11-1 Right
Product Rule
Sparse Grids

Market Shares in Numbers

Rule	N_{nodes}	**Max Abs Error**
pMC	100	2.02858e-02
	1,000	7.88069e-03
	10,000	6.77537e-04
qMC	100	1.14975e-02
	1,000	3.55180e-03
	10,000	5.98465e-04
Product Rule	$3^5 = 243$	5.37642e-04
	$4^5 = 1,024$	3.90495e-05
	$5^5 = 3,125$	6.91544e-06
	$7^5 = 16,807$	0
Stroud 11-1	983	1.03309e-04
	983	1.60621e-04
Sparse	993	4.53294e-05

* Errors relative to GH product rule with 7^5 nodes.

Optimization Overview

We compute point estimates:

- Five starting values for a variety of quadrature rules and draws
- Five MC data sets
- Used both KNITRO and SNOPT without significant change in results
- 'non-convergence' means:
  - Exceeded iteration limit of 100
  - Solver aborts with an error message, e.g. the problem is unbounded, the solver cannot make progress, or all constraints cannot be satisfied
  - Numerical problems evaluating constraints indicative of nearly singular Hessian
- With MPEC, quadrature rules only affect result via constraint that observed market shares equal calculated market shares
- MPEC and NXP are equivalent (Dubé, Fox, & Su, 2009)

Solver Convergence

We find that the solver often fails to converge for better approximations of the integral:

- Sparse grids and GH product rules always find the same optimum when they converge:
  - Same point estimates for all starting values
  - Same value of the objective function for all starting values
  - When the best pMC solution is used as a starting value, solver converges to SGI optimum or fails to converge
- pMC rarely finds the same optimum:
  - Different starting values produce different point estimates and values of GMM objective function
  - Monte Carlo approximation creates spurious local optima
- Monomial rules never converge
- qMC is faster than pMC but not significantly more accurate
- For some Monte Carlo datasets, none of the rules converge

Results: Optimization with pMC ($R = 1,000$)

Dataset	EXIT	INFORM	f_k	CPU Time
5	0	1	203.50784	668.71
5	0	1	213.97591	503.92
5	0	1	203.50784	626.74
5	0	1	208.76144	489.06
5	0	1	208.76144	696.81

Table: Point Estimates: pMC with First 5 good starts and $R = 1,000$ draws

Results: Optimization with pMC ($R = 10,000$)

Dataset	EXIT	INFORM	f_k	CPU Time
5	0	1	260.57447	5450.45
5	0	1	279.95232	6514.08
5	0	1	299.22156	5555.86
5	0	1	299.22156	5444.99
5	0	1	279.95232	4403.82

Table: MPEC Results: pMC with $R = 10,000$ draws

Results: Optimization with SGI

Dataset	EXIT	INFORM	f_k	CPU Time
5	0	1	293.89029	494.45
5	0	1	293.89029	571.11
5	0	1	293.89029	481.82
5	0	1	293.89029	556.80
5	0	1	487.40742	6535.28

Table: Point Estimates: SGI with first 5 good starts and 993 nodes (exact for degree ≤ 11)

Results: Optimization with GH Product Rule

Dataset	EXIT	INFORM	f_k	CPU Time
5	0	1	288.69920	6334.33
5	0	1	288.69920	7553.43
5	0	1	288.69920	7164.02
5	0	1	288.69920	8156.16
5	0	1	288.69920	5521.13

Table: Point Estimates: Gauss-Hermite with first 5 good starts and 7^5 nodes

Results: Optimization with Monomial Rule

Dataset	EXIT	INFORM	f_k	CPU Time
5	0	1	277.89463	441.45
5	0	1	278.03790	441.77
5	0	1	277.89463	548.75
5	0	1	277.89463	1134.53
5	0	1	278.03790	656.13

Table: Point Estimates: Monomial with First 5 Good Starts.

Results: Point Estimates with pMC ($R = 1,000$)

Data	INFORM	θ_{11}	θ_{12}	θ_{13}	θ_{14}	θ_{15}
5	1	3.8847	2.941	1.717	0.7584	-5.028
		(0.7081)	(0.2525)	(0.2558)	(0.1694)	(0.5138)
5	1	0.1228	2.608	1.614	0.2418	-2.575
		(0.4817)	(0.1932)	(0.2031)	(0.1285)	(0.1604)
5	1	3.8847	2.941	1.717	0.7584	-5.028
		(0.7081)	(0.2525)	(0.2558)	(0.1694)	(0.5138)
5	1	1.9287	2.693	1.550	0.7341	-3.911
		(0.6686)	(0.2155)	(0.2226)	(0.1501)	(0.3918)
5	1	1.9287	2.693	1.550	0.7341	-3.911
		(0.6686)	(0.2155)	(0.2226)	(0.1501)	(0.3918)

Table: Point Estimates: pMC with First 5 good starts and $R = 1,000$ draws

Results: Point Estimates with pMC ($R = 1,000$)

Data	INFORM	θ_{21}	θ_{22}	θ_{23}	θ_{24}	θ_{25}
5	1	1.432	1.2450	0.8917	1.2194	1.2161
		(1.019)	(0.3340)	(0.1194)	(0.1908)	(0.1532)
5	1	1.567	0.9745	0.7653	0.5999	0.4921
		(0.6346)	(0.1237)	(0.1238)	(0.1533)	(0.04807)
5	1	1.432	1.2450	0.8917	1.2194	1.2161
		(1.019)	(0.3340)	(0.1194)	(0.1908)	(0.1532)
5	1	2.316	1.4281	0.7147	1.0183	0.8986
		(0.5851)	(0.1760)	(0.1347)	(0.1880)	(0.1172)
5	1	2.316	1.4281	0.7147	1.0183	0.8986
		(0.5851)	(0.1760)	(0.1347)	(0.1880)	(0.1172)

Table: Point Estimates: pMC with First 5 good starts and $R = 1,000$ draws

Results: Point Estimates with pMC ($R = 10,000$)

Data	INFORM	θ_{11}	θ_{12}	θ_{13}	θ_{14}	θ_{15}
5	1	-0.2568	2.340	1.196	0.3208	-2.217
		(0.4224)	(0.1931)	(0.1949)	(0.1170)	(0.1624)
5	1	0.3066	2.506	1.259	0.4342	-2.707
		(0.5113)	(0.1932)	(0.1962)	(0.1176)	(0.1620)
5	1	-0.7372	2.306	1.184	0.3491	-2.073
		(0.4224)	(0.1931)	(0.1949)	(0.1170)	(0.1624)
5	1	-0.7372	2.306	1.184	0.3491	-2.073
		(0.4224)	(0.1931)	(0.1949)	(0.1170)	(0.1624)
5	1	0.3066	2.506	1.259	0.4342	-2.707
		(0.5113)	(0.1932)	(0.1962)	(0.1176)	(0.1620)

Table: Point Estimates: pMC with $R = 10,000$ draws

Results: Point Estimates with pMC ($R = 10,000$)

Data	INFORM	θ_{21}	θ_{22}	θ_{23}	θ_{24}	θ_{25}
5	1	0.6103	1.1014	0.2332	0.5633	0.3884
		(2.189)	(0.09419)	(0.2608)	(0.1058)	(0.04790)
5	1	1.3931	1.1934	0.3408	0.5283	0.5531
		(0.6929)	(0.08841)	(0.1647)	(0.1012)	(0.04609)
5	1	0.7923	0.9923	0.4481	0.7718	0.3472
		(2.189)	(0.09419)	(0.2608)	(0.1058)	(0.04790)
5	1	0.7923	0.9923	0.4481	0.7718	0.3472
		(2.189)	(0.09419)	(0.2608)	(0.1058)	(0.04790)
5	1	1.3931	1.1934	0.3408	0.5283	0.5531
		(0.6929)	(0.08841)	(0.1647)	(0.1012)	(0.04609)

Table: Point Estimates: pMC with $R = 10,000$ draws

Results: Point Estimates with SGI

Data	INFORM	θ_{11}	θ_{12}	θ_{13}	θ_{14}	θ_{15}
5	1	-0.5386	2.263	1.151	0.372	-2.075
		(0.6593)	(0.2231)	(0.1989)	(0.1182)	(0.3571)
5	1	-0.5386	2.263	1.151	0.372	-2.075
		(0.6593)	(0.2231)	(0.1989)	(0.1182)	(0.3571)
5	1	-0.5386	2.263	1.151	0.372	-2.075
		(0.6593)	(0.2231)	(0.1989)	(0.1182)	(0.3571)
5	1	-0.5386	2.263	1.151	0.372	-2.075
		(0.6593)	(0.2231)	(0.1989)	(0.1182)	(0.3571)
5	1	-1.5668	2.804	1.390	0.161	-3.108
		(0.9577)	(0.3847)	(0.3638)	(0.2231)	(0.4480)

Table: Point Estimates: SGI with first 5 good starts and 993 nodes (exact for degree ≤ 11)

Results: Point Estimates with SGI

Data	INFORM	θ_{21}	θ_{22}	θ_{23}	θ_{24}	θ_{25}
5	1	2.704e-06	1.0143	0.3028	0.7257	0.3472
		(1.360E+05)	(0.1080)	(0.2113)	(0.07381)	(0.08151)
5	1	3.862e-09	1.0143	0.3028	0.7257	0.3472
		(9.524E+07)	(0.1080)	(0.2113)	(0.07381)	(0.08151)
5	1	1.490e-06	1.0143	0.3028	0.7257	0.3472
		(2.468E+05)	(0.1080)	(0.2113)	(0.07381)	(0.08151)
5	1	4.880e-06	1.0143	0.3028	0.7257	0.3472
		(7.538E+04)	(0.1080)	(0.2113)	(0.07381)	(0.08151)
5	1	4.535e+00	0.9027	1.1143	1.2092	0.6888
		(0.6692)	(0.2141)	(0.1681)	(0.1588)	(0.1092)

Table: Point Estimates: SGI with first 5 good starts and 993 nodes (exact for degree ≤ 11)

Results: Point Estimates with GH Product Rule

Data	INFORM	θ_{11}	θ_{12}	θ_{13}	θ_{14}	θ_{15}
5	1	-0.551	2.243	1.134	0.3711	-2.066
		(0.5336)	(0.2683)	(0.2078)	(0.1225)	(0.3929)
5	1	-0.551	2.243	1.134	0.3711	-2.066
		(0.5336)	(0.2683)	(0.2078)	(0.1225)	(0.3929)
5	1	-0.551	2.243	1.134	0.3712	-2.066
		(0.5336)	(0.2683)	(0.2078)	(0.1225)	(0.3929)
5	1	-0.551	2.243	1.134	0.3712	-2.066
		(0.5336)	(0.2683)	(0.2078)	(0.1225)	(0.3929)
5	1	-0.551	2.243	1.134	0.3712	-2.066
		(0.5336)	(0.2683)	(0.2078)	(0.1225)	(0.3929)

Table: Point Estimates: Gauss-Hermite with first 5 good starts and 7^5 nodes

Results: Point Estimates with GH Product Rule

Data	INFORM	θ_{21}	θ_{22}	θ_{23}	θ_{24}	θ_{25}
5	1	1.250e-07	1.055	1.578e-06	0.7183	0.3442
		(1.594E+06)	(0.08564)	(2.598E+04)	(0.04176)	(0.1017)
5	1	1.639e-07	1.055	1.072e-06	0.7183	0.3442
		(1.216E+06)	(0.08564)	(3.825E+04)	(0.04176)	(0.1017)
5	1	1.819e-06	1.055	5.292e-07	0.7183	0.3442
		(1.095E+05)	(0.08564)	(7.746E+04)	(0.04176)	(0.1017)
5	1	1.852e-07	1.055	7.546e-07	0.7183	0.3442
		(1.076E+06)	(0.08564)	(5.432E+04)	(0.04176)	(0.1017)
5	1	3.086e-06	1.055	2.230e-06	0.7183	0.3442
		(6.455E+04)	(0.08564)	(1.838E+04)	(0.04176)	(0.1017)

Table: Point Estimates: Gauss-Hermite with first 5 good starts and 7^5 nodes

Results: Point Estimates with Monomial Rule

Data	INFORM	θ_{11}	θ_{12}	θ_{13}	θ_{14}	θ_{15}
5	1	-0.466	2.258	1.155	0.3580	-2.115
		(0.5857)	(0.2434)	(0.1912)	(0.1214)	(0.5060)
5	1	-0.492	2.317	1.214	0.3340	-2.123
		(0.6321)	(0.2652)	(0.1960)	(0.1183)	(0.5158)
5	1	-0.466	2.258	1.155	0.3580	-2.115
		(0.5857)	(0.2434)	(0.1912)	(0.1214)	(0.5060)
5	1	-0.466	2.258	1.155	0.3580	-2.115
		(0.5857)	(0.2434)	(0.1912)	(0.1214)	(0.5060)
5	1	-0.492	2.317	1.214	0.3340	-2.123
		(0.6321)	(0.2652)	(0.1960)	(0.1183)	(0.5158)

Table: Point Estimates: Monomial with First 5 Good Starts.

Results: Point Estimates with Monomial Rule

Data	INFORM	θ_{21}	θ_{22}	θ_{23}	θ_{24}	θ_{25}
5	1	1.550e-06	1.0433	0.2330	0.7726	0.3592
		(1.733E+05)	(0.1083)	(0.1558)	(0.04675)	(0.1425)
5	1	1.174e-07	0.9852	0.5095	0.7826	0.3627
		(2.140E+06)	(0.1021)	(0.08569)	(0.04595)	(0.1455)
5	1	3.531e-07	1.0433	0.2331	0.7726	0.3592
		(7.606E+05)	(0.1083)	(0.1557)	(0.04675)	(0.1425)
5	1	1.848e-06	1.0433	0.2331	0.7726	0.3592
		(1.453E+05)	(0.1083)	(0.1557)	(0.04675)	(0.1425)
5	1	1.785e-06	0.9852	0.5095	0.7826	0.3627
		(1.408E+05)	(0.1021)	(0.08569)	(0.04595)	(0.1455)

Table: Point Estimates: Monomial with First 5 Good Starts.

Numerical Identification

- *Numerical Identification* occurs when the solver converges to (the same) optimum, i.e. it finds the global optimum:
- Lack of convergence means:
  - A numerical problem (scaling, poor choice of variables)
  - Problems with numerical accuracy
  - Lack of variation in the data
  - The model is weakly- or un-identified
- That SNOPT 7 is often superior to KNITRO supports the weak identification story because SNOPT 7 was recently upgraded to handle rank deficient systems better
- Walker (2002) shows that taking too few draws will mask an identification problem in mixed logit models

Importance Sampling

Importance sampling will not rescue pMC:

- Importance sampling is really just a non-linear change of variables
- Consequently, it will help any numerical method
- The fundamental problem with pMC is using an inaccurate method to approximate the integral

Results: Berry's Mapping

Berry's (1994) 'contraction' mapping drives estimation procedure:

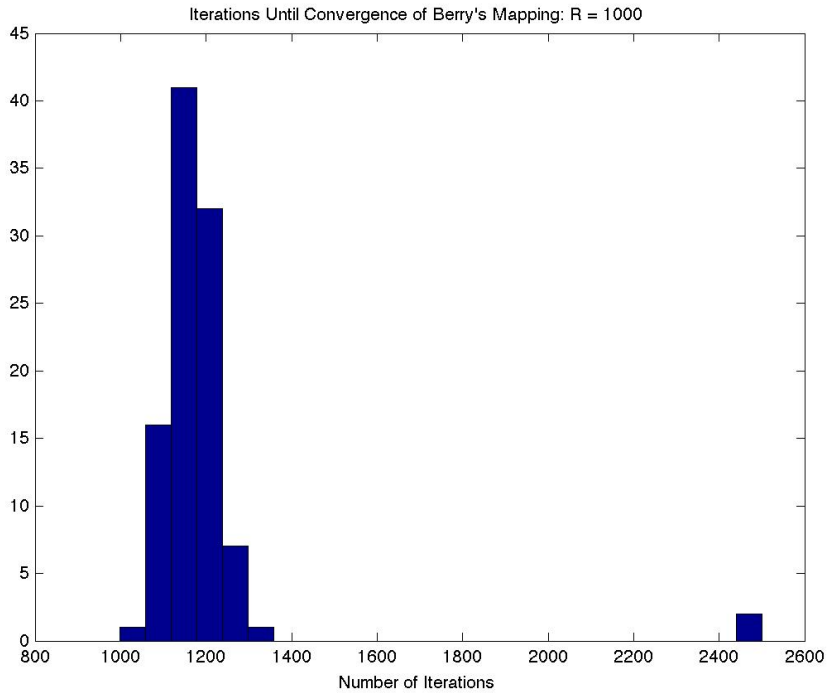
- Used to invert calculated market shares $S_{jt} = s_{jt}\left(\delta\left(\xi;\theta_1\right);\theta_2\right)$ to obtain ξ_{jt}
- A contraction in theory (Berry, 1994): does Berry's theorem hold for MC approximation?
- BLP use importance sampling but few other authors employ variance reduction or qMC
  - I met someone at the scanner data conference using Halton draws, but he was only taking 30 draws....
  - Conlon (2009) uses sparse grids – the only non-pMC BLP paper I know of
- Expensive to compute even with Nevo's transform
- Stopping criterion: $\left\|\exp\left[\delta^{n+1}\right] - \exp\left[\delta^{n}\right]\right\| < \epsilon_{Inner}$

Convergence of Berry's Mapping

We find:

- Berry's mapping converges slowly at best, typically requiring $\sim 1,000$ iterations (not Gaussian!)
- Approximation of integral affects results:
  - Convergence for pMC and qMC requires more iterations as number of draws, R, increases
  - Slower convergence as T and J increase
- Fails sufficiency conditions in BLP's contraction mapping theorem $\sim 10\%$ of the time
- With some Monte Carlo datasets, the mapping never converges ($N_{iter} > 2500$)

Iterations until Convergence



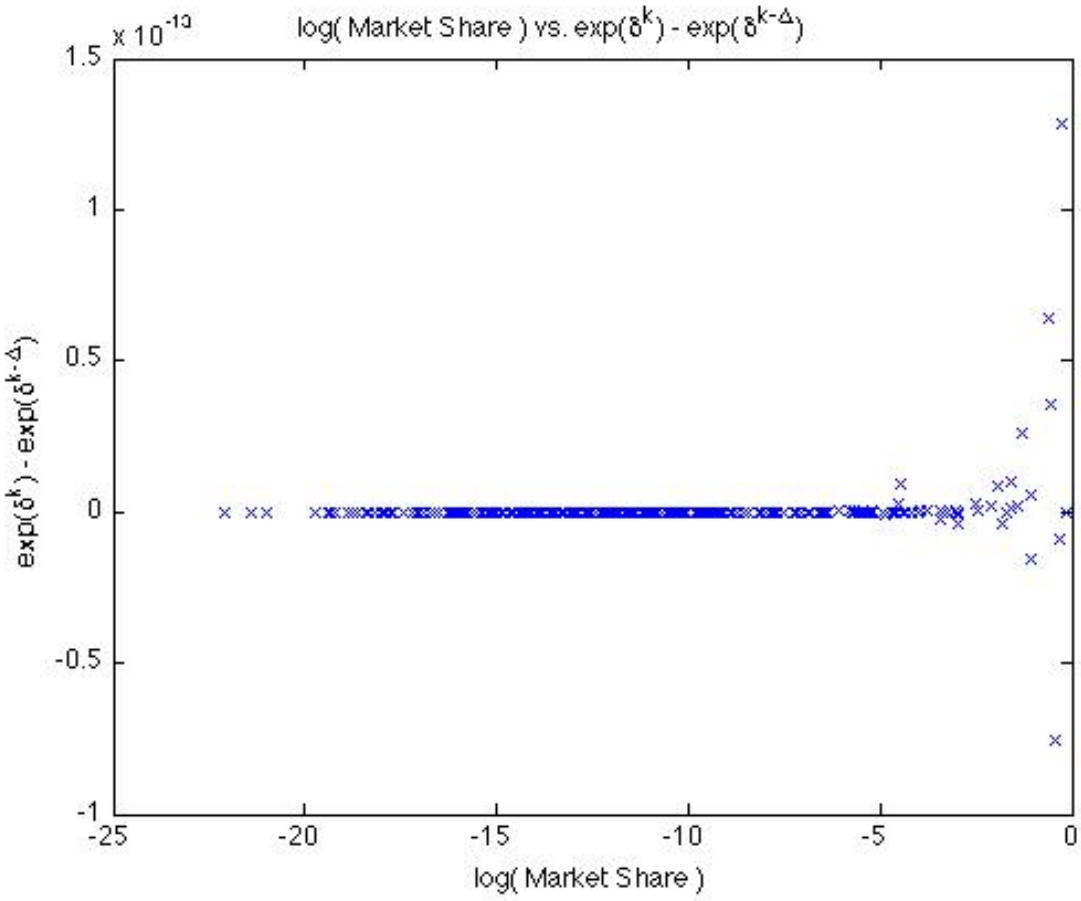
Properties of the Mapping

We also investigate the properties of Berry's mapping:

- Jacobian of mapping is nearly singular:
  - Condition number ~ 19
  - A few eigenvalues near 1
  - The majority close to 0
- We estimate rate of contraction, λ, using (Judd, 1998)

$$\lambda = \left[\left\| \exp\left(\delta^{k+2\Delta}\right) - \exp\left(\delta^{k+\Delta}\right) \right\| / \left\| \exp\left(\delta^{k+\Delta}\right) - \exp\left(\delta^{k}\right) \right\| \right]^{1/\Delta}$$

- λ is always close to 1 and often exceeds it, i.e. the Berry's approximate mapping diverges
- Decreasing T and J slows convergence which should facilitate convergence for a block diagonal system unless increasing size of the system increases chance of finding a local basin of attraction due to MC simulation error
- An identification problem?
- It appears that MC creates local regions of attraction where incorrect solutions are possible



Remark: Berry's mapping has trouble making progress for larger share values ($\Delta = 25$)

Future Research

Custom Monomial Rules:

- Given importance of mixed logit, use modern tools to create custom rules which exploit structure of economic problems such as mixed logit with normally-distributed taste shocks
- Ideal theory + homotopy
- For maximum efficiency and accuracy

Future Research

Convergence Issues:

- Is the approximate Berry mapping a contraction?
- Does the solution from Berry's map agree with better methods for solving non-linear equations such as homotopy?
- What drives non-convergence of solvers? Lack of identification?

Identification:

- Under what conditions is BLP identified?
- When does poor integration mask identification problems in BLP

Conclusion

There are several dangers to poor numerical approximations of integrals:

- Inaccurate results
- Much greater computational costs
- Masking of identification problems

For multidimensional problems of moderate dimension:

- Monomial-rules and Sparse Grids Integration are best options for integrals unless integrating over a very large number of dimensions
- Polynomial-based rules provide superior efficiency and accuracy

Of the MC methods, qMC was significantly more efficient than pMC and had much lower variance

Conclusion

Use of pMC in BLP models cause several problems:

- Incorrect point estimates
- Inaccurate share values
- Different solutions for different starting values
- Non-convergence of solver and Berry's mapping
- These problems are usually a sign of identification problems