High Performance Quadrature Rules

How Numerical Integration Affects a Popular Model of Product Differentiation

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

June 9, 2011

The Big Picture

The goals of this talk are:

- To demonstrate importance of fast, accurate approximations of multi-dimensional integrals
- To introduce polynomial-based quadrature methods
- To show how polynomial-based rules out perform Monte Carlo rules in the context of the Berry, Levinsohn, and Pakes (1995) model of differentiated products

Benefits

Better quadrature methods allow us to build richer models of behavior:

- Higher dimensional integrals
- Faster execution permits:
    - Nesting in estimation/optimization loop
    - Larger data sets
    - More robustness checks
    - Quicker feedback on ideas

Numerical approximation of integrals is often the limiting factor which determines the cost of numerical calculations.

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)

Integration is ubiquitous

Choice Models with Heterogeneity

The BLP model (and mixed logit) depends on equation equating predicted and observed market shares

$$s_{jt}\left(\delta_{jt};\theta_2\right) = \int \frac{\exp\left(\delta_{jt}+\mu_{jt}\left(\nu\right)\right)}{1+\sum\limits_{k\in J}\exp\left(\delta_{kt}+\mu_{kt}\left(\nu\right)\right)} dF\left(\nu\right)$$

- Accuracy $\Rightarrow$ correct point estimates
- Quickly $\Rightarrow$ complete calculations (Nested Fixed Point Algorithm)

Introduction to Numerical Integration

Numerical Integration Basics

Most rules approximate a (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

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

- The crucial issue is how to choose the nodes and weights, $\{w_j, y_j\}$
- Ideally, a rule should have $\lim\limits_{R \to \infty} Q^R[f] = I[f]$, i.e. converge to the truth

Overview of Methods

Approaches differ in how the nodes are chosen:

- pseudo-Monte Carlo (pMC)
- polynomial-based methods such as a Gaussian rule, e.g.:

Rule	$w(x)$	**Domain**
Gauss-Hermite	$\exp\left(-x^2\right)$	$(-\infty, \infty)$
Gauss-Legendre	1	$[-1, 1]$
Gauss-Laguerre	$\exp(-x)$	$[0, \infty)$

Example: Mixed Logit

The mixed logit is a common example.

- Conditional shares with linear utility & Type I Extreme value:

$$s_{ij}\left(\alpha_i\right) = \frac{\exp\left(-\alpha_i \log p_j + x_j^T \beta\right)}{\sum\limits_k \exp\left(-\alpha_i \log p_k + x_k^T \beta\right)}$$

- Computed market shares are then:

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

Note: mixed logit $\leftrightarrow$ random coefficients

Integration in higher dimensions

Some Terminology

A monomial is the product of powers of the different variables:

- $x_1^{p_1} \cdot x_2^{p_2} \cdots x_n^{p_n}$
- I.e, a monomial is $x^{\mathbf{p}} \equiv \prod_j x_j^{p_j}$
- where $\mathbf{p} = (p_1, p_2, \ldots, p_J)$
    - Degree is $\sum_j p_j$
- A fundamental building block of multi-dimensional polynomials
- Analogous to x^n in one dimension

Monte Carlo

Monte Carlo methods:

- Draw nodes $y_k = (x_1, \ldots, x_n)$ from a suitable distribution
- Uses weights $w_k = 1/R$, $\forall k$
- Intuition based on statistics
- Inefficient: need 100x more draws to increase precision by one decimal place!

Gaussian Tensor Products

A first attempt at a better multi-dimensional rule just takes tensor products of a one-dimensional Gaussian rule:

- Converges to the truth if Riemann-Stieltjes integral exists + regularity conditions
- Does not scale well: n-dimensional problem requires R^n nodes

Polynomial Rules

But, it is possible to exploit structure of the problem to create more efficient rules:

- Monomial Rules
  - Efficient
  - Derived by solving a system of polynomial equations
  - See Stroud (1971)
- Sparse Grids Integration (SGI)
  - Also parsimonious
  - See Heiss & Winschel (2008) and Gerstner & Griebel (1998)
- Both rules have desired properties:
  - Exact for all monomials $\leq$ chosen degree
  - Scale well as degree of exactness or number of dimensions increases
  - Number of nodes is polynomial in degree and exactness

	Sparse Grid	11-1 L	$\overline{\lvert pMC \rvert}$	$\sigma(pMC)$
1	6.7e-15	5.2e-12	2e-14	0
x_1^1	2.3e-17	7.1e-15	0.0075	0.0092
$x_1^1 x_2^1$	2.8e-17	0	0.008	0.01
$x_1^1 x_2^1 x_3^1 x_4^1 x_5^1$	0	0	0.0086	0.011
x_1^2	-5.2e-14	9.6e-13	0.012	0.014
x_1^4	-1.5e-13	3.9e-13	0.076	0.096
x_1^6	-7.6e-13	4e-13	0.76	0.94
$x_2^6 x_4^4$	-2.2e-12	-3.1e-13	7.4	9.4
x_1^{10}	-4.9e-11	2.9e-11	1.8e+02	2.1e+02
$x_1^5 x_2^4 x_3^2$	-6.9e-16	1.8e-15	3.9	6.8
x_1^{12}	-5e-10	-7.2e+02	3.2e+03	3.9e+03
x_1^{13}	-1e-11	-2.9e-11	1.3e+04	2e+04
x_1^{14}	-7.2e-09	-3.1e+04	6.2e+04	8.2e+04
x_1^{15}	-5.8e-11	2.3e-10	2.4e+05	4.3e+05
x_1^{16}	-3.4e+04	-8.9e+05	1.3e+06	1.8e+06
$x_1^6 x_2^6 x_3^4 x_4^2 x_5^2$	-4.3e+02	1.6e+05	8.5e+02	2.6e+03
$x_1^8 x_2^6 x_3^4 x_4^2 x_5^2$	-4e+03	1.8e+06	6.7e+03	1.9e+04
$x_1^{10} x_2^5 x_3^4 x_4^2 x_5^2$	0	5.8e-11	1.9e+04	6.5e+04

How quadrature rules affect results in BLP

BLP Estimation

Estimation uses GMM moments formed from ξ_{jt}:

- Invert observed vs. predicted shares to recover ξ_{jt}:

$$s_{jt}^{obs} = s_{jt}^{pred}\left(\xi; X, \theta\right)$$

where

$$s_{jt}^{pred}\left(\xi; X, \theta\right) = \int \frac{\exp\left(\delta_{jt} + \mu_{jt}\left(\nu\right)\right)}{1 + \sum_k \exp\left(\delta_{kt} + \mu_{kt}\left(\nu\right)\right)} dF\left(\nu\right)$$

- Perform GMM using residual
- Historically, used Nested Fixed Point Algorithm
- MPEC (Su & Judd (2010)) is current state of the art.

Overview of Results

Choice of quadrature rule impacts results:

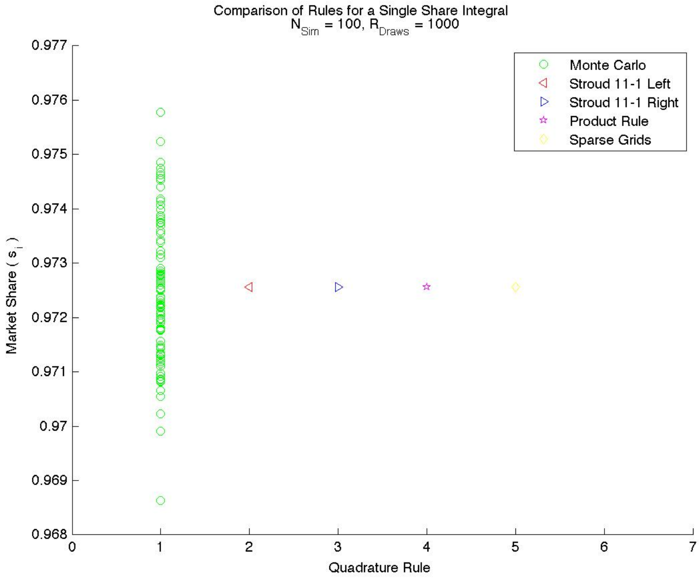
- Predicted market share integrals
- Computational cost & accuracy
- Point estimates
- Standard errors, especially for GMM asymptotic variance formula
- Solver convergence

To make these issues concrete, we examine how pMC and polynomial rules affect results in BLP model of differentiated products.

Monte Carlo Experiments

We investigate the performance of BLP using synthetic data:

- Simulate a typical BLP setup with five random coefficients and endogenous price
  - Currently five MC data sets
  - Code based on Dubé, Fox, & Su (2009)
- Sparse grids generated using code from Heiss & Winschel (2008)
- Monomial rule 11-1 from Stroud (1971)



Comparison of Rules for a Single Share Integral
N_Sim = 100, R_Draws = 1000
Monte Carlo
Stroud 11-1 Left
Stroud 11-1 Right
Product Rule
Sparse Grids
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

Market Shares in Numbers

Rule	N_{nodes}	**Ave Abs Error***
pMC	100	6.73236e-04
	1,000	2.19284e-04
	10,000	6.82600e-05
Gaussian Product Rule	$3^5 = 243$	1.60235e-05
	$4^5 = 1,024$	2.51356e-06
	$5^5 = 3,125$	5.42722e-07
	$7^5 = 16,807$	0*
Stroud 11-1	983	2.80393e-05
Sparse	993	4.09252e-06

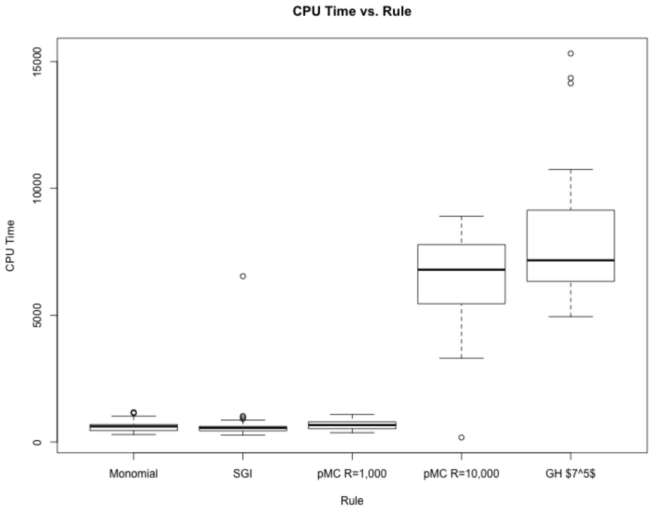
* Errors relative to Gaussian product rule with 7^5 nodes.

Results: Market Shares

Polynomial rules clearly superior to pMC:

- Clustered in center of pMC cloud, usually at exactly the same point
- Close to mean of pMC simulations, as expected, because pMC is unbiased.
- Monomial rule and SGI use many fewer nodes than GH product rule or MC.
- But, more problems with overflow/underflow because of better approximation of tails.
- Must increase number of pMC draws 100x for each additional decimal place of accuracy

Polynomial rules also approximate the gradient of the GMM objective function more accurately!


CPU Time vs. Rule
CPU Time
0
5000
10000
15000
Monomial
SGI
pMC R=1,000
pMC R=10,000
GH 7^5
Rule

Results: Optimization

Optimization across multiple starts and datasets shows:

- pMC $R = 1,000$ vs. Monomial and SGI rules:
  - Comparable numbers of nodes
  - Polynomial rules are $10 - 100x$ more accurate!
- pMC $R = 10,000$ vs. Monomial and SGI rules:
  - Polynomial rules are $10x$ faster!
  - Polynomial rules still more accurate....

Results: Optimization

pMC results are not reliable:

- Different starting values and the same pMC draws produce different local optima
- Same starting value and different pMC draws produce different local optima

Polynomial rules are robust when using the Dubé, Fox, & Su data generating process:

- Solver always finds the same optimum (point estimates)
- Solver also finds this optimum when started at the best pMC optimum

Point Estimates: SGI vs. pMC

	Bias		Mean Abs Dev		Med Abs Dev		RMSE	
	SGI	pMC	SGI	pMC	SGI	pMC	SGI	pMC
θ_{11}	0.96	12.34	2.29	13.25	1.20	3.64	4.00	28.92
θ_{12}	0.02	−0.13	0.52	0.38	0.22	0.33	0.94	0.48
θ_{13}	−0.28	−0.38	1.47	1.21	0.62	0.99	3.01	1.51
θ_{21}	22.57	128.22	23.01	128.24	2.62	34.06	81.76	253.87
θ_{22}	0.02	−0.04	0.12	0.16	0.07	0.13	0.19	0.20
θ_{23}	0.08	0.64	0.36	0.75	0.16	0.79	0.75	0.90

Table: Comparison of bias in point estimates : SGI vs. pMC for T=2 markets and J=24 products with 165 nodes.

Point Estimates

Polynomial rules dominate pMC for the same number of nodes:

- Need fewer starts to find best optimum because simulation error creates:
  - False local optima
  - Non-convexities in surface
- Polynomial rules produce
  - Much lower bias
  - More robust estimates
- Conjecture: errors in share integrals propagate to point estimates à la Dubé, Fox, and Su (2011).

Results: Standard Errors

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

θ_{21}	θ_{22}	θ_{23}	θ_{24}	θ_{25}
1.250e-07	1.055	1.578e-06	0.7183	0.3442
(5.628E+06)	(0.07972)	(4.830E+04)	(0.1011)	(0.04931)
1.639e-07	1.055	1.072e-06	0.7183	0.3442
(4.293E+06)	(0.07972)	(7.111E+04)	(0.1011)	(0.04931)
1.819e-06	1.055	5.292e-07	0.7183	0.3442
(3.868E+05)	(0.07972)	(1.440E+05)	(0.1011)	(0.04931)
1.852e-07	1.055	7.546e-07	0.7183	0.3442
(3.800E+06)	(0.07972)	(1.010E+05)	(0.1011)	(0.04931)
3.086e-06	1.055	2.230e-06	0.7183	0.3442
(2.280E+05)	(0.07972)	(3.417E+04)	(0.1011)	(0.04931)

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

Identification and Standard Errors

Polynomial rules often produce much larger standard errors than pMC:

- Simulation error increases curvature around local optima, making standard errors artificially small
- Polynomial-rules more accurately approximate derivatives and hence standard errors
- Polynomial rules can show when a model is poorly identified
    - pMC standard errors are too tight
    - Polynomial rules without sufficient exactness also mask identification problems
- Walker (2002) shows that taking too few draws will mask identification problems 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 should help any numerical method
- The fundamental problem with pMC is using an inaccurate method to approximate the integral

Conclusion

Using better quadrature rules has many benefits and essentially no drawbacks:

- $>10x$ more accurate for the same number of nodes
- $>10x$ faster for the same accuracy
- Reliable point estimates
- More accurate standard errors
- Improved performance permits:
  - Richer models, especially with costly estimation algorithms
  - Larger data sets
  - More robustness checks