Florian Heiss und Viktor Winschel:

Estimation with Numerical Integration on Sparse Grids

Munich Discussion Paper No. 2006-15

Department of Economics
University of Munich

Volkswirtschaftliche Fakultät
Ludwig-Maximilians-Universität München

Estimation with Numerical Integration on Sparse Grids*

Florian Heiss† Viktor Winschel‡

April 12, 2006

Abstract

For the estimation of many econometric models, integrals without analytical solutions have to be evaluated. Examples include limited dependent variables and nonlinear panel data models. In the case of one-dimensional integrals, Gaussian quadrature is known to work efficiently for a large class of problems. In higher dimensions, similar approaches discussed in the literature are either very specific and hard to implement or suffer from exponentially rising computational costs in the number of dimensions – a problem known as the "curse of dimensionality" of numerical integration. We propose a strategy that shares the advantages of Gaussian quadrature methods, is very general and easily implemented, and does not suffer from the curse of dimensionality. Monte Carlo experiments for the random parameters logit model indicate the superior performance of the proposed method over simulation techniques.

JEL Classification: C15, C25, C51
Keywords: Estimation, Quadrature, Simulation, Mixed Logit

*We would like to thank Alexander Ludwig, Axel Börsch-Supan, Melanie Lührmann, Daniel McFadden, Paul Ruud, and Joachim Winter for valuable comments and suggestions.

†University of Munich, Department of Economics, mail@florian-heiss.de

‡University of Mannheim, Department of Economics, winschel@rumms.uni-mannheim.de

1 Introduction

Many econometric models imply likelihood and moment functions that involve multidimensional integrals without analytically tractable solutions. This problem arises frequently in microeconometric latent dependent variable (LDV) models in which all or some of the endogenous variables are only partially observed. Other sources include unobserved heterogeneity in nonlinear models and models with dynamically optimizing agents.

There are different approaches for numerical integration. It is well known that Gaussian quadrature can perform very well in the case of one-dimensional integrals of smooth functions as suggested by Butler and Moffit (1982). Quadrature can be extended to multiple dimensions. The direct extension is a tensor product of one-dimensional quadrature rules. However, computing costs rise exponentially with the number of dimensions and become prohibitive for more than four or five dimensions. This phenomenon is also known as the curse of dimensionality of numerical integration.

The main problem of this product rule is that the class of functions in which it delivers exact results is not restricted to polynomials of a given total order. Unlike in the univariate case, general efficient quadrature rules for this class are much harder to directly derive and often intractable (Judd 1998, Cools 2003, Ch. 7.5). They are therefore usually considered impractical for applied research (Bhat 2001, Geweke 1996).

These problems led to the advancement and predominant use of simulation techniques for the numerical approximation of multidimensional integrals in the econometric literature, see for example McFadden (1989) or Börsch-Supan and Hajivassiliou (1993). Hajivassiliou and Ruud (1994) provide an overview over the general approaches of simulation and Train (2003) provides a textbook treatment with a focus on discrete choice models, one of the major classes of models for which these methods were developed and frequently used.

This paper proposes and investigates the performance of a different approach that can be traced back to Smolyak (1963). Sparse grids integration (SGI) has been advanced in recent research in numerical mathematics, see for example Novak and Ritter (1999). The class of functions for which it is exact is confined to polynomials of a given total order. This dramatically decreases computational costs in higher dimensions. It is based on one-dimensional quadrature but extends it to higher dimensions in a more careful way than the tensor product rule. This implies that it is easily implemented and very general since only one-dimensional quadrature nodes and weights have to be derived.

After discussing the general approaches to numerical integration, SGI is introduced in Section 3. Section 4 presents the Monte Carlo design and results. They directly address the question of interest for estimation: Which method delivers the best estimates with a given amount of computing costs? The experiments are based on a panel data random parameters logit models which are widely used in applied discrete choice analysis. They vary the panel data dimensions, the number of alternatives, the dimension of unobserved taste components and the parameterization of the data generating process. The results show that the SGI methods clearly outperform simulations based on both pseudo and quasi random numbers. Section 5 concludes.

2 Univariate Numerical Integration

We start the discussion with univariate integration in order to introduce the notation and to lay the ground for an extension to the multivariate case. Write a general univariate integration problem as

$$I_1[g] = \int_\Omega g(x)\, w(x)\, dx. \tag{1}$$

In limited dependent variable (LDV) and other econometric models, integrals often arise in the calculation of expected values. In this case, x represents a random variable, $g(x)$ is a function which is in the nontrivial case nonlinear in x and $w(x)$ represents the p.d.f. of x with support Ω. The integral $I_1[g]$ is the expected value of $g(x)$.

A leading example for this kind of problem in microeconometric panel data models are random effects (RE) models like the RE probit model discussed for example by Butler and Moffit (1982) – x represents the individual random effect, $g(x)$ is the probability of the sequence of observed outcomes conditional on the explanatory variables, parameters and x. The integral $I_1[g]$ is the marginal (with respect to x) probability. This value is needed for example for the evaluation of the likelihood function.

Since for nonlinear functions $g(x)$ the integral has in general no closed-form solution, it has to be approximated numerically. One possible approach is Monte Carlo simulation. Given a number R of replications, a set of random numbers or "nodes" $[x_1, ..., x_R]$ is generated such that each x_r is a draw from the distribution characterized by $w(x)$. The simulated integral is then equal to

$$S_{1,R}[g] = \frac{1}{R} \sum_{r=1}^{R} g(x_r). \tag{2}$$

Under very weak conditions, the simulated value is unbiased and $\sqrt{R}$-consistent by a law of large numbers. In the one-dimensional case, other strategies like simple Riemann sums with regularly-spaced nodes are straightforward - we will come back to equivalent strategies when discussing multiple integration.

A somewhat different strategy is taken by Gaussian and related quadrature rules. They provide a formula which delivers the exact integral for a class of functions – typically polynomials – that approximate $g(x)$. Gaussian quadrature rules construct nodes and weights such that the rule is exact for polynomials of a given order with a minimal number of nodes. Butler and Moffit (1982) suggest using Gaussian quadrature for the RE probit model and show evidence that it works very well compared to simulation.

Define a sequence of quadrature rules $V = \{V_i : i \in \mathbb{N}\}$. Each rule V_i specifies a set of nodes $\mathbb{X}_i \subset \mathbb{R}$ and a corresponding weight function $w_i : \mathbb{X}_i \to \mathbb{R}$ that is appropriate for w and Ω. The index i denotes the increasing level of accuracy which depends on the number of nodes in $\mathbb{X}_i$. The approximation of $I_1[g]$ by V_i is then given as

$$V_i[g] = \sum_{x \in \mathbb{X}_i} g(x) w_i(x). \tag{3}$$

If V_i is a Gaussian rule with R nodes, then $V_i[g] = I_1[g]$ if g is a polynomial of order $2R - 1$ or less.

The sets of nodes and weights depend on w and Ω, but not on g. While they are not trivial to determine, for the most common cases they are tabulated in the literature and efficient software is available, see for example Miranda and Fackler (2002) for implementations in Matlab. Given nodes and weights, Gaussian quadrature rules are straightforward to implement, since equation (3) merely requires to calculate a weighted sum of function values.

For general functions g, the sequence of approximations $V_i[g]$ converges to $I_1[g]$ under weak assumptions as the number of function evaluations rise. A sufficient condition is that g is bounded and Riemann-integrable. There are various results concerning the speed of convergence for additional smoothness properties. For example if g has n bounded derivatives, many Gaussian quadrature approximations converge to the true value at a rate of R^{-n}. This is much better than the $\sqrt{R}$-consistency of Monte Carlo simulation if $g(x)$ is sufficiently smooth. A more detailed discussion of Gaussian quadrature can be found in the literature, see for example Davis and Rabinowitz (1984).

3 Multivariate Numerical Integration

3.1 Formulation of the Problem

Many econometric models involve integration problems over several dimensions. In the RE probit model mentioned above, these problems arise for example if the error term follows a random process over time, has an error components structure, or multiple outcomes are modeled as in the multinomial probit model. In the Monte Carlo section below, we discuss a multinomial choice model with an error components structure – the mixed or random parameters logit model.

Write the integration problem in the mulivariate case as

$$I_D[g] \quad = \quad \int_{\Omega_1} \cdots \int_{\Omega_D} g(x_1, ..., x_D)\, \tilde{w}(x_1, ..., x_D)\, dx_D \cdots dx_1. \qquad (4)$$

We restrict the discussion to the case in which the weight function $\tilde{w}(x_1, ..., x_D)$ can be decomposed as

$$\tilde{w}(x_1, ..., x_D) = \prod_{d=1}^{D} w(x_d) \qquad (5)$$

and where $\Omega_d = \Omega$ for all $d = 1, ..., D$. In the interpretation with $[x_1, ..., x_D]$ representing random variables and $\tilde{w}(x_1, ..., x_D)$ their joint p.d.f., this restriction is equivalent to assuming that the random variables are independently and identically distributed. Independence is crucial for the remainder of this paper, while identical distributions are merely assumed for notational convenience to save on another subscript. This structure of the weighting function is less restrictive than it might seem. If independence is violated in the original formulation of the problem, a change of variables often leads to such a structure. If for example $\mathbf{z}$ denotes a vector of jointly normally distributed random variables with mean $\boldsymbol{\mu}$ and covariance matrix Σ, then $\mathbf{x} = L^{-1}(\mathbf{z} - \boldsymbol{\mu})$ with $LL' = \Sigma$ is distributed i.i.d. standard normal.

Monte Carlo simulation is the most commonly used technique in the econometric literature for the numerical approximation of integrals of the form (4) in the multivariate case $D > 1$. The only difference to the univariate case in equation (2) is that for each replication, a vector $[x_{1,r}, ..., x_{D,r}]$ is drawn from $\tilde{w}(x_1, ..., x_D)$ at which the function g is evaluated. The result of $\sqrt{R}$-consistency is independent of the number of dimensions D. This does of course not imply properties of the approximation with a finite number of replications.

Quasi-Monte Carlo methods and antithetic sampling algorithms distribute the nodes more evenly than pseudo-random nodes generated by a standard random number generator. Therefore, they generally achieve both a better approximation quality with a finite number of replications and in many cases also faster convergence rates.

As argued above, deterministic integration schemes such as Gaussian quadrature often work very well for univariate problems. Their extension to multiple dimensions is not as straightforward as for simulation. A natural goal for a quadrature rule in multiple dimensions is exactness for multivariate polynomials of a given total order, also known as complete polynomials (Judd 1998, Ch. 6.12). To be more specific, consider a D-variate polynomial

$$g(x_1, ..., x_D) = \sum_{t=1}^{T} a_t \prod_{d=1}^{D} x_d^{j_{t,d}} \tag{6}$$

for some $T \in \mathbb{N}$, $[a_1, ..., a_T] \in \mathbb{R}^T$ and $[j_{t,1}, ..., j_{t,D}] \in \mathbb{N}^D$ for all $t = 1, ..., T$. Define the total order of g as the maximal sum of exponents $\max_{t=1,...,T} \sum_{d=1}^{D} j_{t,d}$ Unlike in the univariate case, general efficient quadrature rules for this class in the multivariate case are much harder to directly derive and often intractable (Judd 1998, Ch. 7.5). As a result, there has been a large literature on very specific problems. For a collection of these rules, see for example Cools (2003). They are not only limited to very narrowly defined problems, but often also difficult to implement. They are therefore usually considered impractical for applied research (Bhat 2001, Geweke 1996).

We focus on two approaches which are easily implemented and cover a wide range of applications since they merely combine one-dimensional quadrature rules. We first discuss the well known product rule extension (Tauchen and Hussey 1991) in Section 3.2. It does not restrict its attention to polynomials of a given total order and suffers from exponentially rising cost with increasing dimensions. It is therefore inefficient for moderate dimensions $D > 1$ and infeasible for higher dimensions, say $D > 5$. In Section 3.3, we suggest to use an extension of Gaussian quadrature to higher dimensions which is computationally efficient, easy to implement, and very general.

3.2 Multivariate Quadrature: The Product Rule

Univariate quadrature rules can be extended easily to multiple dimensions by the product rule. Define the tensor product over univariate quadrature rules with potentially different accuracy levels in each dimension indicated

by the multi-index $[i_1, ..., i_D]$ as

$$(V_{i_1} \otimes \cdots \otimes V_{i_D})[g] \quad = \quad \sum_{x_1 \in \mathbb{X}_{i_1}} ... \sum_{x_D \in \mathbb{X}_{i_D}} g(x_1, ..., x_D) \prod_{d=1}^{D} w_{i_d}(x_d), \quad (7)$$

where the nodes $\mathbb{X}_{i_1}, ...\mathbb{X}_{i_D}$ and weights $w_{i_1}, ..., w_{i_D}$ are those implied by the underlying one-dimensional quadrature rules $V_{i_1}, ..., V_{i_D}$.

The widely known product rule $T_{D,k}$ for D-variate Gaussian quadrature with accuracy level k (Tauchen and Hussey 1991) is simply this tensor product with the same accuracy in each dimension $(V_k \otimes \cdots \otimes V_k)[g]$. If V_k is exact for all univariate polynomials of order o or less, then $T_{D,k}$ is exact for a special class of D-variate polynomials. Instead of a bound on the total order (that is the sum of exponents), it restricts the maximum exponent of all monomials to be at most o. For example a second-order Taylor approximation in two dimensions is a polynomial of total order 2: it contains terms in x_1^2, x_2^2, and $x_1 x_2$. Instead of being exact in this class of functions, the product rule is additionally exact for terms in $x_1^2 x_2$, $x_1 x_2^2$, and $x_1^2 x_2^2$. In higher dimensions and higher order, the number of these terms, which diminish at a higher rate in the approximations, rises quickly which makes the product rule inefficient and causes the curse of dimensionality (Judd 1998, Ch. 6.12).

The product rule evaluates the function g at the full grid of points $\mathbb{X}_k \otimes \cdots \otimes \mathbb{X}_k$. In D dimensions, the product rule therefore requires R^D evaluations of the function g if the underlying univariate rule V_k is based on R nodes. This exponential growth of computational costs with the number of dimensions is labelled "curse of dimensionality" of multivariate integration. While for example Gaussian quadrature exactly evaluates a univariate polynomial of order 7 with 4 function evaluations, the corresponding product rule with 20 dimensions requires $4^{20} = 1,099,511,627,776$ evaluations which is generally prohibitive.

3.3 Multivariate Quadrature on Sparse Grids

We suggest to extend univariate quadrature rules to multiple dimensions with a substantially smaller number of function evaluations in higher dimensions than the product rule. This can be achieved by combining the univariate rules in a different way than the product rule. The basic idea goes back to Smolyak (1963) and is a general method for multivariate extensions of univariate approximation and integration operators.

The construction of Smolyak can be defined as follows. For an underlying sequence of univariate quadrature rules, define $V_0[g] = 0$ and the difference of the approximation when increasing the level of accuracy from $i-1$ to i as

$$\Delta_i[g] = V_i[g] - V_{i-1}[g] \quad \forall i \in \mathbb{N}. \tag{8}$$

With $\mathbf{i} = [i_1, ..., i_D]$, define for any $q \in \mathbb{N}_0$

$$\mathbb{N}_q^D = \left\{ \mathbf{i} \in \mathbb{N}^D : \sum_{d=1}^{D} i_d = D + q \right\} \tag{9}$$

and $\mathbb{N}_q^D = \emptyset$ for $q < 0$. For example, $\mathbb{N}_2^2 = \{[1,3],[2,2],[3,1]\}$. The Smolyak rule with accuracy level $k \in \mathbb{N}$ for D-dimensional integration is defined as

$$A_{D,k}[g] \;=\; \sum_{q=0}^{k-1} \sum_{\mathbf{i} \in \mathbb{N}_q^D} \left(\Delta_{i_1} \otimes \cdots \otimes \Delta_{i_D}\right)[g]. \tag{10}$$

For $D = 1$, the underlying one-dimensional quadrature rule emerges as a special case:

$$A_{1,k}[g] \;=\; \sum_{i=1}^{k} \left(V_i[g] - V_{i-1}[g]\right) = V_k[g]. \tag{11}$$

This integration rule is designed to be exact for polynomials of a given total order:

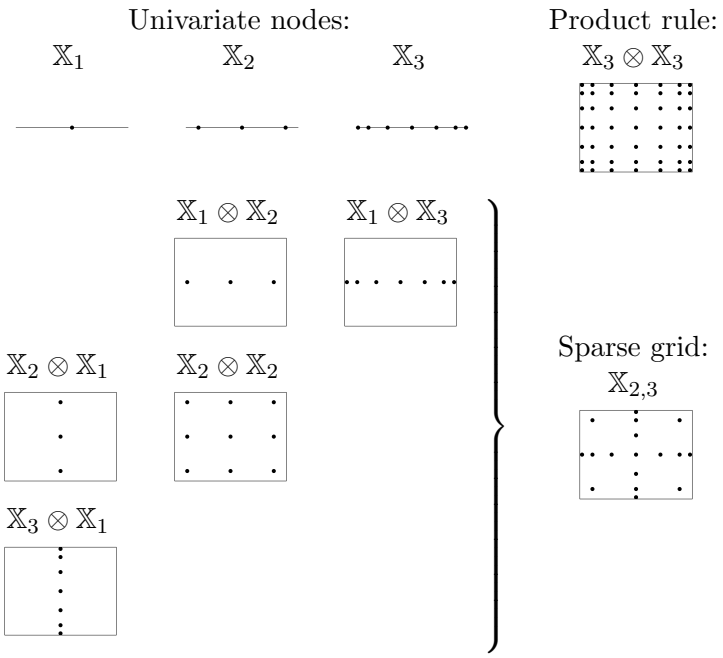
Theorem 1 *Assume that the sequence of univariate quadrature rules $V = \{V_i : i \in N\}$ is defined such that V_i is exact for Ω, w, and all univariate polynomials of order $2i-1$ or less. This implies the Smolyak rule $A_{D,k}$ using V as the univariate basis sequence is exact for D-variate polynomials of total order $2k-1$ or less.*

See the appendix for a proof.

It is instructive to express $A_{D,k}[g]$ directly in terms of the univariate quadrature rules instead of their differences. It can be written as (Wasilkowski and Woźniakowski 1995)

$$A_{D,k}[g] \;=\; \sum_{q=k-D}^{k-1} (-1)^{k-1-q} \binom{D-1}{k-1-q} \sum_{\mathbf{i} \in \mathbb{N}_q^D} (V_{i_1} \otimes \cdots \otimes V_{i_D})[g]. \tag{12}$$

Figure 1: Construction of the sparse grid in two dimensions



This rule is a weighted sum of product rules with different combinations of accuracy levels $\mathbf{i} = [i_1, ..., i_D]$. Their sum is bounded which has the effect that the tensor product rules with a relatively fine sequence of nodes in one dimension are relatively coarse in the other dimensions. This is analogous to the bound on the sum of exponents for multivariate polynomials of a total order.

Figure 1 demonstrates the construction of the sparse grid by the Smolyak rule for a simple example with $D = 2$ and $k = 3$. The nodes for a sequence of univariate quadrature rules $\mathbb{X}_1, \mathbb{X}_2$, and $\mathbb{X}_3$ are shown in the top of the figure. The product rule $\mathbb{X}_3 \otimes \mathbb{X}_3$ evaluates the function at all two-dimensional combinations of nodes prescribed by $\mathbb{X}_3$ which are shown in the upper right part of the figure. As equation (12) shows, the sparse grids rule combines tensor products of lower degree $\mathbb{X}_i \otimes \mathbb{X}_j$ such that $3 \leq i + j \leq 4$. The nodes of these products as well as the resulting sparse grid are shown in the lower part of the figure.

The set of nodes used by the sparse grids rule (12) can be written as

$$\mathbb{X}_{D,k} = \bigcup_{q=k-D}^{k-1} \bigcup_{\mathbf{i}\in\mathbb{N}_q^D} (\mathbb{X}_{i_1} \otimes \cdots \otimes \mathbb{X}_{i_D}). \tag{13}$$

The number of nodes in $\mathbb{X}_{D,k}$ depends on the univariate nodes $\mathbb{X}_1, ..., \mathbb{X}_k$ and can in general not be easily calculated. We first discuss the speed in which it grows as $D \to \infty$ if Gaussian quadrature is used for the sequence of underlying univariate rules. Remember that the product rule $T_{D,k}$ with underlying Gaussian quadrature rules uses k^D nodes so that the logarithm of the nodes is of order $O(D^{-1})$ as $D \to \infty$.

Theorem 2 *Consider the sparse-grids rule $A_{D,k}$ with underlying Gaussian quadrature rules $V = \{V_i : i \in N\}$ such that each $\mathbb{X}_i$ used by V_i has i nodes. For a given accuracy k and rising D, the logarithm of nodes in $\mathbb{X}_{D,k}$ is of order $O(\log(D)^{-1})$.*

We give a proof in the appendix. At least asymptotically, the number of nodes (its logarithm) does not rise exponentially (linearly) as for the product rule, but only polynomially (logarithmically). This is of course only of limited use in practice since realistic D are far from infinity. We therefore give precise numbers for different dimensions below. Before, we discuss alternatives to Gaussian quadrature as the underlying univariate rules.

Theorem 1 requires that in the sequence of quadrature rules $V_1, V_2...$ each V_i is exact for all univariate polynomials of order $2i-1$ or less. As discussed above, Gaussian quadrature rules achieve this requirement on univariate exactness with a minimal number of i nodes for each V_i. Obviously, a low number of univariate quadrature nodes helps to obtain a low total number of nodes in the sparse grids rule.

In the example presented in Figure 1, the sets of univariate nodes are nested in the sense that $\mathbb{X}_i \subseteq \mathbb{X}_j$ if $i \leq j$. Because the nodes are nested, the sets $\mathbb{X}_1 \otimes \mathbb{X}_2$ and $\mathbb{X}_2 \otimes \mathbb{X}_1$ do not add any distinct nodes to the sparse grid and also the other sets share a substantial number of points. This makes the union of the tensor products a much smaller set than in the other extreme case in which each set contains different nodes so $\mathbb{X}_i \cap \mathbb{X}_j = \emptyset$ if $i \neq j$. Gaussian quadrature rules are close to the latter case – generally, only the midpoint is shared by rules with an odd number of nodes.

An example for nested sequences of univariate quadrature rules are Kronrod-Patterson sequences (Patterson 1968). A Kronrod-Patterson rule with accuracy level i adds a number of points to the set of nodes $\mathbb{X}_{i-1}$ of the

preceding accuracy level and updates the weights. So by design, $\mathbb{X}_i \subseteq \mathbb{X}_j$ if $i < j$. The additional nodes are chosen such that a maximum polynomial exactness is achieved. Because of the restriction that all nodes in $\mathbb{X}_{i-1}$ are to be reused, Kronrod-Patterson rules generally require a higher number of nodes to achieve the same univariate polynomial exactness as Gaussian quadrature rules which optimally choose the nodes without the requirement of nested sets.[1] With this approach, the goal in constructing univariate sequences is to add as few nodes as possible to reduce the computational costs but enough to fulfill the requirements on polynomial exactness of Theorem 1. Petras (2003) discusses this problem for the case of unweighted integration (Gauss-Legendre equivalent) and Genz and Keister (1996) for the normal p.d.f. weights (Gauss-Hermite equivalent). We supply the sequences of both approaches with the accompanying Matlab and Stata code to this paper.

Table 1 shows the number of function evaluations required by different multivariate integration rules to achieve a given degree of polynomial exactness. The product rule suffers from the curse of dimensionality. The number of nodes for the Smolyak rule also rises with the number of dimensions, but substantially slower. As discussed, while in one dimension Gaussian quadrature is more efficient, in higher dimensions the Kronrod-Patterson rules need fewer nodes.

Sparse grids integration can be easily implemented in practice. It is not necessary to construct the grid of nodes and the corresponding weights according to equation (12) for each approximation of the integral. Since they do not depend on the integrand, it suffices to do this once or use precalculated values. We provide general Matlab and Stata code for these calculations. Existing code using simulation has to be changed only by using these nodes and weights instead of drawing random numbers and by replacing raw means of the results with weighted sums.

4 Monte Carlo Experiments

4.1 The Random Parameters Logit Model

When using numerical integration in the context of estimation, the ultimate goal is to achieve good parameter estimates. They obviously depend on the

[1] With one or three integration nodes, the Kronrod-Patterson rule and the Gaussian rule coincide. With $2^m - 1$ nodes for $m > 1$, Gaussian rules are exact for polynomials up to order $2(2^m - 1) - 1$, whereas Kronrod-Patterson rules are only exact for polynomials up to order $3 \cdot 2^{m-1} - 1$. So the ratio of both approaches 3/4 as m rises.

Table 1: Number of function evaluations

	Product rule	Smolyak rule	
Dimensions	**Gaussian**	**Gaussian)**	**KP**
Level $k = 2$, Polynomial exactness = 3			
$D = 1$	2	2	3
$D = 5$	32	11	11
$D = 10$	1024	21	21
$D = 20$	1048576	41	41
Level $k = 3$, Polynomial exactness = 5			
$D = 1$	3	3	3
$D = 5$	243	66	51
$D = 10$	59049	231	201
$D = 20$	3486784401	861	801
Level $k = 4$, Polynomial exactness = 7			
$D = 1$	4	4	7
$D = 5$	1024	286	151
$D = 10$	1048576	1771	1201
$D = 20$	1099511627776	12341	10001
Level $k = 5$, Polynomial exactness = 9			
$D = 1$	5	5	7
$D = 5$	3125	1001	391
$D = 10$	9765625	10626	5281
$D = 20$	95367431640625	135751	90561

approximation quality of the involved integrals. In this section we present Monte Carlo experiments to asses the relative performance of the numerical integration algorithms. Different random parameters logit (RPL) or mixed logit models are implemented. The RPL model is widely used for studying choices between a finite set of alternatives. McFadden and Train (2000) provide an introduction to this model and a discussion of its estimation by simulation methods. This model has also been used before to study the performance of different simulation methods (Bhat 2001, Hess, Train, and Polak 2006).

Consider a random sample of N individuals. The data has a panel structure, so that for each of the subjects T choices are observed. In each of these choice situations, the individual is confronted with a set of J alternatives and chooses one of them. These alternatives are described by K strictly exogenous attributes. The $(K \times 1)$ vectors $\mathbf{x}_{itj}$ collect these attributes of al-

ternative $j = 1, ..., J$ in choice situation $t = 1, ..., T$ of individual $i = 1, ..., N$.

Random utility maximization (RUM) models of discrete choices assume that the individuals pick the alternative which results in the highest utility. The researcher obviously does not observe these utility levels. They are modeled as latent variables for which the observed choices provide an indication. Let the utility that individual i attaches to alternative j in choice situation t be represented by the random coefficients specification

$$U_{itj} \quad = \quad \mathbf{x}'_{itj}\boldsymbol{\beta}_i + e_{itj}. \tag{14}$$

It is given by a linear combination of the attributes of the alternative, weighted with individual-specific taste levels $\boldsymbol{\beta}_i$. These individual taste levels are distributed across the population according to a parametric joint p.d.f. $f(\boldsymbol{\beta}_i; \boldsymbol{\theta})$ with support $\Psi \subseteq \mathbb{R}^K$. The i.i.d. random variables e_{itj} capture unobserved utility components. They are assumed to follow an Extreme Value Type I (or Gumbel) distribution.

Our goal is to estimate the parameters $\boldsymbol{\theta}$ of the taste level distribution. Let y_{itj} denote an indicator variable that has the value 1 if individual i chooses alternative j in choice situation t and 0 otherwise.[2] Denote the vector of observed individual outcomes as $\mathbf{y}_i = [y_{itj}; t = 1, ..., T, j = 1..., J]$ and the matrix of all strictly exogenous variables as $\mathbf{x}_i = [\mathbf{x}_{itj}; t = 1, ..., T, j = 1..., J]$. Then, the probability that the underlying random variable $\mathbf{Y}_i$ equals the observed realization $\mathbf{y}_i$ conditional on $\mathbf{x}_i$ and the individual taste levels $\boldsymbol{\beta}_i$ can be expressed as

$$P_i^*(\boldsymbol{\beta}_i) = \Pr(\mathbf{Y}_i = \mathbf{y}_i | \mathbf{x}_i, \boldsymbol{\beta}_i) \quad = \quad \prod_{t=1}^{T} \frac{\prod_{j=1}^{J} \exp(\mathbf{x}'_{itj}\boldsymbol{\beta}_i)^{y_{itj}}}{\sum_{j=1}^{J} \exp(\mathbf{x}'_{itj}\boldsymbol{\beta}_i)}. \tag{15}$$

The likelihood contribution of individual i is equal to the joint outcome probability as a function of $\boldsymbol{\theta}$. It can be written as

$$P_i(\boldsymbol{\theta}) = \Pr(\mathbf{Y}_i = \mathbf{y}_i | \mathbf{x}_i, \boldsymbol{\theta}) \quad = \quad \int_{\Psi} P_i^*(\boldsymbol{\beta}_i) f(\boldsymbol{\beta}_i; \boldsymbol{\theta}) \, d\boldsymbol{\beta}_i. \tag{16}$$

A solution for this K-dimensional integral does in general not exist in closed form and has to be approximated numerically.

4.2 Approximating the probabilities

We start with a simple case of the general model. Let $J = 2$ so that the model simplifies to a binary choice model. Also let there only be $K = 1$

[2] Note that by definition, $\sum_{j=1}^{J} y_{itj} \, \forall i, t$.

explanatory variable for which $x_{it2} - x_{it1} = 1$ for all t. Let the taste level β_i be normally distributed over the population with mean 1 and variance σ^2. Let the individual have chosen T_1 times alternative 1 and T_2 times alternative 2, so that there are in total $T_1 + T_2$ observations. The likelihood contribution in equation (16) can then be simplified to

$$\begin{aligned} P_i(\theta) &= \int_{-\infty}^{\infty} P_i^*(z)\phi(z)\,dz \\ \text{with } P_i^*(z) &= (1+\exp(-1-\sigma z))^{-T_1}(1+\exp(1+\sigma z))^{-T_2}. \end{aligned} \quad (17)$$

This univariate integral can either be simulated or approximated using standard Gaussian quadrature. Figure 2 shows the function $P_i^*(z)$ for two different cases and depicts the numerical approaches to its integration. The simulated probability with R simulation draws can be represented as the sum of R rectangles each of which has a width of $1/R$ and a height that corresponds to the function value at randomly chosen points. Quadrature exactly integrates a polynomial of a given degree that represents an approximation to the integrand.

How well $P_i^*(z)$ is approximated by a low-order polynomial depends on the parameters. With high T and σ^2, the function has large areas in which it is numerically zero (or unity). These areas create a problem for the polynomial fit. In Figure 2, two cases are presented. In the simple case of Model 1 with $T_1 = 0, T_2 = 1$, and $\sigma^2 = 1$, the function $P_i^*(z)$ – and therefore its integral – is already well approximated by a third-order polynomial. A ninth-order polynomial is indistinguishable from the original function. In order to integrate this ninth-order polynomial exactly, Gaussian quadrature rules only need $R = 5$ function evaluations.

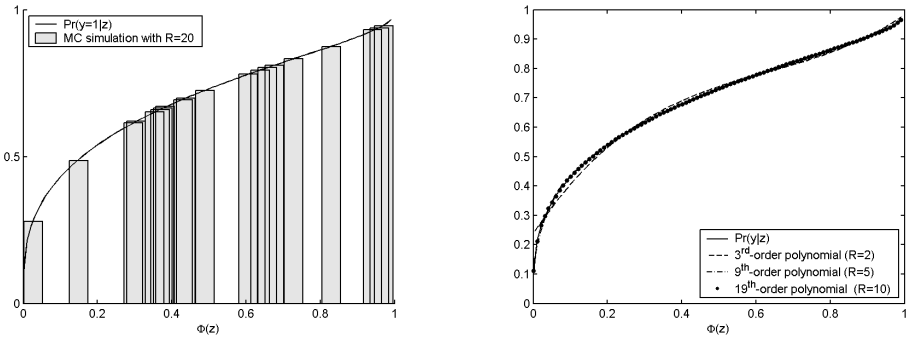
In the second model with $T = 20, T_1 = 8, T_2 = 12$, and $\sigma^2 = 5$, the problem of large tails with zero conditional probability is evident. A third-order polynomial does a poor job in approximating the function and there are noticeable differences between the original function and its ninth-order polynomial approximation. A 19^{th}-order polynomial for which Gaussian quadrature needs 10 function evaluations however is again indistinguishable from the true function. This can of course be arbitrarily problematic with even higher σ^2 and T. With a sharp and narrow peak, approximation by simulation can have poor properties, too. Intuitively, this is since only a small fraction of simulation draws are within the nonzero area.

This diagnosis directly suggests a remedy. The function can be transformed by a change of variables that leads to a better-behaved function. In the context of simulation, this approach is importance sampling and for

Figure 2: Approximating probabilities in one dimension

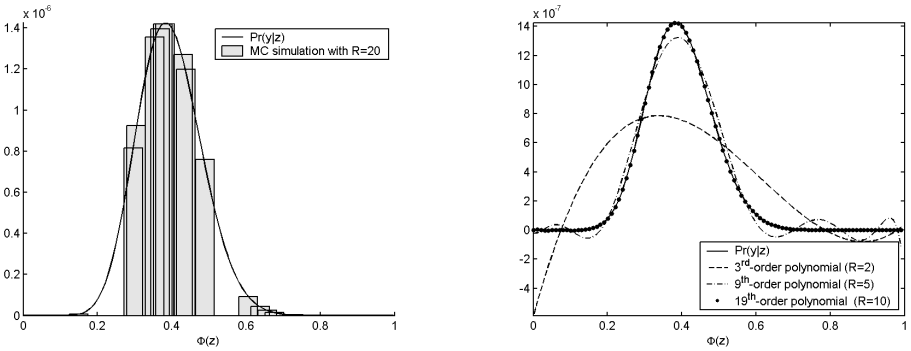
Model 1: $T = 1, T_1 = 0, T_2 = 1, \sigma^2 = 1$

Simulation ($R = 20$) Quadrature ($R = 2, 5, 10$)



Model 2: $T = 20, T_1 = 8, T_2 = 12, \sigma^2 = 4$

Simulation ($R = 20$) Quadrature ($R = 2, 5, 10$)



univariate quadrature, it works similarly well, see Liu and Pierce (1994). For the remainder of this paper, we will not use these improvements and leave the exploration of their advantages in the multivariate case for future research.

For the two models depicted in Figure 2 and two more extreme cases, Table 2 presents performance measures of simulation and Gaussian quadrature approximations of the choice probabilities (17). The numbers presented are absolute errors for the quadrature approximations and root mean squared errors for simulations which have been performed 1,000 times for each model and number of draws. All errors are defined relative to the value obtained by a Riemann sum with 10,000 grid points. For all models, Gaussian quadra-

Table 2: Approximating probabilities in one dimension: RMSE

	$R=2$	$R=5$	$R=10$	$R=100$	$R=1000$
Model 0: $T=1, T_0=0, T_1=1, \sigma^2=.25$					
Simulation	0.0944	0.0569	0.0421	0.0130	0.0043
Quadrature	0.0046	0.0008	0.0002	0.0000	–
Model 1: $T=1, T_0=0, T_1=1, \sigma^2=1$					
Simulation	0.1846	0.1111	0.0822	0.0253	0.0085
Quadrature	0.0102	0.0012	0.0002	0.0000	–
Model 2: $T=20, T_0=8, T_1=12, \sigma^2=4$					
Simulation	1.0206	0.6745	0.4841	0.1516	0.0484
Quadrature	0.8115	0.2472	0.0024	0.0000	–
Model 3: $T=30, T_0=10, T_1=20, \sigma^2=9$					
Simulation	1.5222	0.9658	0.6591	0.2121	0.0677
Quadrature	1.0000	0.6336	0.0402	0.0000	–

The reported numbers are root mean squared errors relative to the "true" value

ture with 10 nodes performs better than simulation with 1,000 draws.

To study more complex models, we turn to a setup where $J = T = 5$. The number of explanatory variables K determines the dimensionality of the integration problem. We chose $K = 3, 5, 10$, and 20 for the Monte Carlo studies reported in Table 3. The individual taste levels are specified as i.i.d. normal random variables with mean 1 and variance $2/K$ to hold the total variance of U_{itj} constant as K changes. Instead of using one set of predefined data, we draw 1,000 samples from the joint distribution of $\mathbf{y}_i$ and $\mathbf{x}_i$, where the x_{itj} are specified as independent uniform random variables and the conditional distribution of the Bernoulli random variables $\mathbf{y}_{itj}$ is given in equation (15). For each of these draws, we approximate the joint outcome probability using simulation and Smolyak integration with different numbers of nodes. For the calculation of the mean squared errors, we approximate the true value by simulation with 200,000 draws.

The rows denoted as "simulation" represent simulated probabilities using a standard random number generator. They perform worst in all cases. The "quasi Monte Carlo" results are obtained using modified latin hypercube sequences (MLHS) which are shown to work well for the estimation of RPL models by Hess, Train, and Polak (2006).[3] This method works much better than the standard simulation. The product rule performs better than

[3]We also experimented with Halton sequences which do not seem to make too much of a difference compared to MLHS. Results can be requested from the authors.

Table 3: RMSE of probabilities: $\sigma^5 = 2/K, J = T = 5$

$K = 3, R =$	7	8	87	125	495	512
Simulation	0.3373	0.2971	0.0879	0.0776	0.0372	0.0382
Quasi MC	0.2287	0.1802	0.0382	0.0331	0.0161	0.0152
Product rule		0.0669		0.0112		0.0048
Sparse grids	0.0303		0.0050		0.0020	
$K = 5, R =$	11	32	151	243	903	1024
Simulation	0.2663	0.1448	0.0705	0.0536	0.0303	0.0278
Quasi MC	0.1486	0.0796	0.0310	0.0257	0.0127	0.0130
Product rule		0.0567		0.0277		0.0171
Sparse grids	0.0243		0.0049		0.0043	
$K = 10, R =$	21	201	1024	1201		
Simulation	0.1987	0.0654	0.0284	0.0255		
Quasi MC	0.1076	0.0317	0.0135	0.0128		
Product rule			0.0420			
Sparse grids	0.0173	0.0211		0.0035		
$K = 20, R =$	41	801	10001			
Simulation	0.1428	0.0324	0.0095			
Quasi MC	0.0795	0.0156	0.0048			
Sparse grids	0.0125	0.0160	0.0027			

The reported numbers are root mean squared errors relative to the "true" value

both simulation methods in low dimensions, especially $K = 3$. In five dimensions, its advantage disappears and in ten dimensions, it is clearly the worst method. For $K = 20$, we did not obtain results since it is not computationally feasible. In all cases, the sparse grids method clearly outperforms all other methods. Table 5 in the appendix shows results for the more difficult case $\sigma^2 = 5/K$ and $J = T = 5$. While all errors rise, the relative performances remain unchanged.

4.3 Estimating the parameters

One of the main reasons why approximations of the outcome probabilities are interesting is that they are required for most estimators of the parameters $\boldsymbol{\theta}$. We discuss maximum simulated (or approximated) likelihood estimation such that the estimators are defined as

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \sum_i \log\left(\tilde{P}_i(\boldsymbol{\theta})\right),$$

where $\tilde{P}_i(\boldsymbol{\theta})$ is some approximation of the individual joint outcome probability $P_i(\boldsymbol{\theta})$. Alternatively, estimators could be based on simulated scores or moments. We chose this estimator since it is easiest to implement and by far the most widely used for these kinds of models. For a discussion of the various approaches, see for example Hajivassiliou and Ruud (1994).

It is clear that the quality of approximation of $\tilde{P}_i(\boldsymbol{\theta})$ translates into the properties of the estimators. We specify a number of different models and estimate the parameters using simulation, antithetic sampling, and numerical integration using different degrees of accuracy. As a starting point, a reference model is specified with $N = 1000$, $T = 5$, $J = 5$, $K = 10$, $\mu = 1$, and $\sigma = 0.5$. Then each of these numbers is varied separately to asses their impact on the approximation errors of the different methods. For each of these settings, estimates were obtained for 100 artificial data sets. The K-dimensional vectors of properties of the alternatives $\mathbf{x}_{itj}$ were drawn from a standard uniform distribution. The model parameters μ and σ are constrained to be equal for all properties to simplify the estimation and estimated for each data set. We used the same methods as discussed in the previous section pseudo-random Monte Carlo (PMC), quasi-random Monte Carlo (QMC) and sparse grids integration (SGI).

Table 4 shows results for different dimensions of integration. The simulation-based estimates are much better for μ than for σ. This can be explained by the fact that while the simulated probabilities $\tilde{P}_i(\mu, \sigma)$ are unbiased for the true values $P_i(\mu, \sigma)$, the log transformation introduces downward bias. This bias depends on the simulation variance which in turn depends on σ. This tends to bias $\hat{\sigma}$ downwards. As predicted from the results for the approximated probabilities, standard simulation is dominated by quasi-random simulation. SGI is again clearly the best method and for example in ten dimensions requires only 21 nodes for the same accuracy for which QMC needs 201 and PMC 1201 function evaluations.

In the appendix, results for other model parameter choices are presented. Table 6 shows variations of μ and σ and Table 7 varies N, T, and J. The basic finding are unaffected by these changes. As σ increases, the approximation error rises and therefore all methods perform worse.[4] With a larger number of i.i.d. cross-sectional observation units N or longitudinal observations T, the estimators improve. Their relative advantages remain unaffected.

[4] If σ has a very large value, all methods fail to give reasonable estimation results. As discussed above, adaptive rescaling might solve this problem.

Table 4: Errors of the estimated parameters with different K

	RMSE ($\hat{\mu}$)			RMSE ($\hat{\sigma}$)		
R	PMC	QMC	SGI	PMC	QMC	SGI
Dimension $K = 2$						
9	0.0486	0.0458	0.0448	0.3648	0.1648	0.1177
45	0.0458	0.0452	0.0448	0.1712	0.1246	0.1177
961	0.0449	0.0449	0.0448	0.1179	0.1170	0.1177
Dimension $K = 4$						
9	0.0411	0.0361	0.0340	0.3857	0.1985	0.0902
81	0.0341	0.0340	0.0339	0.1319	0.0963	0.0923
1305	0.0338	0.0339	0.0339	0.0921	0.0916	0.0923
Dimension $K = 10$						
21	0.0481	0.0353	0.0272	0.2951	0.1554	0.0668
201	0.0298	0.0277	0.0272	0.0874	0.0708	0.0654
1201	0.0276	0.0271	0.0271	0.0691	0.0662	0.0654
Dimension $K = 14$						
29	0.0507	0.0348	0.0240	0.2493	0.1321	0.0601
393	0.0252	0.0247	0.0252	0.0665	0.0618	0.0632
3361	0.0251	0.0252	0.0249	0.0629	0.0637	0.0631
Dimension $K = 20$						
41	0.0655	0.0465	0.0290	0.2323	0.1390	0.0620
801	0.0298	0.0285	0.0276	0.0634	0.0599	0.0564
10001	0.0289	0.0291	0.0291	0.0594	0.0585	0.0585

The reported numbers are RMSEs relative to the true value

5 Conclusions

Multidimensional integrals are prevalent in econometric estimation problems. Only for special cases, closed-form solutions exist. With a flexible model specification, the researcher frequently has to resort to numerical integration techniques. Previously discussed methods of quadrature in multiple dimensions are either very specific and difficult to implement or suffer from the "curse of dimensionality" – exponentially rising computational costs with increasing dimensions.

We suggest a method that solves both problems. It merely requires the derivation of quadrature nodes and weights for univariate integration. These are widely available in software implementations and tabulations. This makes the method easy to implement and very broadly applicable.

The reason why product rules suffer from the curse of dimensionality is that the class of functions for which they deliver exact results is not limited to polynomials of a given total order. The proposed method of integration on sparse grids is confined to this set and therefore requires a dramatically lower number of function evaluations in higher dimensions. The increase of computational costs is only polynomial instead of exponential.

As a result, this method can be used as an efficient alternative to simulation methods. An intuitive explanation of the advantage of quadrature-based methods over simulation is that it efficiently uses smoothness properties of the integrand to recover its shape over the whole domain.

After introducing the method and discussing its properties, we present extensive Monte Carlo evidence for the random parameters logit model. The results show that the computational costs to achieve a negligible approximation error are dramatically lower with the suggested approaches than with simulation estimators.

Recent research in numerical mathematics suggests possible refinements of integration on sparse grids. First, instead of predefining an approximation level in terms of the number of nodes, a critical value of the approximation error can be specified and the required number of function evaluations can be determined automatically. Second, the approximation does not have to be refined in each dimension symmetrically. It is also possible to invest more effort in the most relevant dimensions. These dimensions can also be determined automatically in an adaptive fashion (Gerstner and Griebel 2003). Third, quadrature-based methods can be refined to efficiently handle functions that are not well-behaved. This can be achieved either by a change of variables or by piecewise integration. These extensions are left for future research.

Appendix

Proof of Theorem 1

Note that

$$A_{D,k}\left[\sum_{t=1}^{T} a_t \prod_{d=1}^{D} x_d^{j_{t,d}}\right] = \sum_{t=1}^{T} a_t A_{D,k}\left[\prod_{d=1}^{D} x_d^{j_{t,d}}\right].$$

Therefore, it suffices to establish polynomial exactness for any of the T monomials. Consider $g = \prod_{d=1}^{D} x_d^{j_d}$ for some sequence $j_1, ..., j_D$ with

(i) $\sum_{d=1}^{D} j_d \leq 2k-1$.

The theorem states that this implies $A_{D,k}[g] = I_D[g]$. For the sequence of underlying univariate quadrature rules $V_1, V_2, ...$ we have by assumption

(ii) $V_k[x^j] = I_1[x^j]$ if $j \leq 2k-1$.

For the univariate case $D = 1$, we know that $A_{1,k} = V_k$ (see equation (11)) so the theorem follows immediately from (ii). For the multivariate case, a proof is presented via induction over D. Suppose that polynomial exactness has been established for $D-1$ dimensions:

(iii) $A_{D-1,\tilde{k}}\left[\prod_{d=1}^{D-1} x_d^{j_d}\right] = I_{D-1}\left[\prod_{d=1}^{D-1} x_d^{j_d}\right]$ if $\sum_{d=1}^{D-1} j_d \leq 2\tilde{k}-1$

It remains to be shown that this implies polynomial exactness for D dimensions.

First note that because of the multiplicative structure of the monomial integrand,

$$(\Delta_{i_1} \otimes \cdots \otimes \Delta_{i_D})\left[\prod_{d=1}^{D} x_d^{j_d}\right] = \prod_{d=1}^{D} \Delta_{i_d}[x_d^{j_d}].$$

Rewrite the general Smolyak rule (10) by separating the sum over the D^{th} dimension:

$$A_{D,k}[g] \;=\; \sum_{i_D=1}^{k} \sum_{\tilde{q}=0}^{k-i_D} \sum_{\mathbf{i}\in\mathbb{N}_{\tilde{q}}^{D-1}} (\Delta_{i_1} \otimes \cdots \otimes \Delta_{i_D})[g].$$

Combining these two expressions, we get

$$A_{D,k}\left[\prod_{d=1}^{D} x_d^{j_d}\right] \;=\; \sum_{i_D=1}^{k} \Delta_{i_D}\left[x_D^{j_D}\right] A_{D-1,k-i_D+1}\left[\prod_{d=1}^{D-1} x_d^{j_d}\right].$$

By (ii), we know that whenever $2(i_D - 1) > j_D$, $V_{i_D}\left[x_D^{j_D}\right] = V_{i_D-1}\left[x_D^{j_D}\right] = I\left[x_D^{j_D}\right]$. Therefore, $\Delta_{i_D}\left[x_D^{j_D}\right]$ and the summands are zero unless $j_D \geq 2(i_D - 1)$. This in turn implies together with (i) that for nonzero summands $\sum_{d=1}^{D-1} j_d \leq 2(k - i_D + 1) - 1$ and therefore $A_{D-1,k-i_D+1}\left[\prod_{d=1}^{D-1} x_d^{j_d}\right] = I_{D-1}\left[\prod_{d=1}^{D-1} x_d^{j_d}\right]$ by (iii).

With $i_D^* = .5(j_D + 1)$ for odd j_D and $i_D^* = .5j_D$ for even j_D, it follows that

$$\begin{aligned} A_{D,k}\left[\prod_{d=1}^{D} x_d^{j_d}\right] &= I_{D-1}\left[\prod_{d=1}^{D-1} x_d^{j_d}\right] \sum_{i_D=1}^{i_D^*} \Delta_{i_D}\left[x_D^{j_D}\right] \\ &= I_{D-1}\left[\prod_{d=1}^{D-1} x_d^{j_d}\right] V_{i_D^*}\left[x_D^{j_D}\right] \end{aligned}$$

By (ii), $V_{i_D^*}\left[x_D^{j_D}\right] = I_1\left[x_D^{j_D}\right]$ and the theorem follows.

Proof of Theorem 2

Let $R_{D,k}$ denote the number of distinct nodes in $\mathbb{X}_{D,k}$. For $D \geq k$, it can be bounded as

$$R_{D,k} \leq \sum_{q=0}^{k-1} \sum_{\mathbf{i} \in \mathbb{N}_q^D} \prod_{d=1}^{D} i_d, \tag{18}$$

since the univariate quadrature rule V_i has i nodes. The inequality comes from the fact that the midpoints appear repeatedly in the underlying quadrature rules. Note that the average element of $\mathbf{i} \in \mathbb{N}_q^D$ is $\frac{D+q}{D}$ and that the product is maximized if all elements have the same value. Therefore

$$\prod_{d=1}^{D} i_d \leq \left(\frac{D+q}{D}\right)^D \quad \forall \mathbf{i} \in \mathbb{N}_q^D.$$

The number of vectors $\mathbf{i} \in \mathbb{N}_q^D$ is $\binom{D-1+q}{D-1}$. So

$$R_{D,k} \leq \tilde{R}_{D,k} = k\binom{D-1+k}{D-1}\left(\frac{D+k}{D}\right)^D.$$

As $D \to \infty$, $\left(\frac{D+k}{D}\right)^D \to \exp(k)$, $\binom{D-1+k}{D-1} \to \frac{D^k}{k!}$ and therefore $\log\left(\tilde{R}_{D,k}\right) \to k - \log((k-1)!) + k\log(D) = O(\log(D)^{-1})$.

Further results

Table 5: RMSE of probabilities: $\sigma^2 = 5, J = T = 10$

$K = 3, R =$	7	8	87	125	495	512
Simulation	0.7284	0.8184	0.2093	0.1860	0.0902	0.0923
Quasi MC	0.6234	0.5849	0.1498	0.1147	0.0636	0.0600
Product rule		0.2060		0.0480		0.0221
Sparse grids	0.2696		0.0217		0.0055	
$K = 5, R =$	11	32	151	243	903	1024
Simulation	0.6628	0.3921	0.2264	0.1652	0.0807	0.0726
Quasi MC	0.5226	0.3517	0.1550	0.1106	0.0653	0.0564
Product rule		0.1828		0.0917		0.0566
Sparse grids	0.2434		0.0567		0.0151	
$K = 10, R =$	21	201	1024	1201		
Simulation	0.6708	0.2025	0.0950	0.0816		
Quasi MC	0.5278	0.1783	0.0784	0.0713		
Product rule			0.1574			
Sparse grids	0.1798	0.1058		0.0573		
$K = 20, R =$	41	801	10001			
Simulation	0.4928	0.1169	0.0348			
Quasi MC	0.3971	0.1176	0.0286			
Product rule						
Sparse grids	0.1034	0.0756	0.0395			

The reported numbers are root mean squared errors relative to the "true" value

Table 6: Errors of the estimated parameters with different μ and σ

	RMSE ($\hat{\mu}$)			RMSE ($\hat{\sigma}$)		
R	PMC	QMC	SGI	PMC	QMC	SGI
Parameters $\mu = 0.5, \sigma = 0.5$						
21	0.0268	0.0226	0.0208	0.2922	0.1533	0.0625
201	0.0211	0.0208	0.0209	0.0813	0.0638	0.0603
1201	0.0209	0.0209	0.0209	0.0631	0.0625	0.0605
Parameters $\mu = 2, \sigma = 0.5$						
21	0.0842	0.0598	0.0460	0.3050	0.1545	0.0712
201	0.0510	0.0470	0.0475	0.1034	0.0767	0.0738
1201	0.0478	0.0475	0.0472	0.0803	0.0732	0.0735
Parameters $\mu = 1, \sigma = 0.25$						
21	0.0264	0.0253	0.0257	0.1759	0.0984	0.0919
201	0.0250	0.0252	0.0255	0.0929	0.0888	0.0893
1201	0.0255	0.0256	0.0256	0.0868	0.0955	0.0923
Parameters $\mu = 1, \sigma = 1$						
21	0.1070	0.0867	0.0490	0.4095	0.3053	0.1754
201	0.0409	0.0363	0.0343	0.0994	0.0824	0.0746
1201	0.0316	0.0309	0.0303	0.0614	0.0597	0.0553

The reported numbers are RMSEs relative to the true value

Table 7: Errors of the estimated parameters with different N, T, and J

	RMSE ($\hat{\mu}$)			RMSE ($\hat{\sigma}$)		
R	PMC	QMC	SGI	PMC	QMC	SGI
$N = 500$						
21	0.0511	0.0441	0.0417	0.2993	0.1792	0.1101
201	0.0416	0.0414	0.0418	0.1284	0.1102	0.1126
1201	0.0413	0.0417	0.0417	0.1023	0.1109	0.1121
$N = 2000$						
21	0.0417	0.0275	0.0165	0.3124	0.1530	0.0547
201	0.0185	0.0171	0.0166	0.0769	0.0575	0.0515
1201	0.0168	0.0166	0.0166	0.0548	0.0517	0.0512
$T = 3$						
21	0.0513	0.0414	0.0374	0.3254	0.1836	0.1157
201	0.0378	0.0369	0.0383	0.1283	0.1220	0.1253
1201	0.0380	0.0381	0.0381	0.1208	0.1240	0.1234
$T = 10$						
21	0.0195	0.0189	0.0281	0.2284	0.1375	0.0457
201	0.0253	0.0269	0.0291	0.0572	0.0466	0.0377
1201	0.0284	0.0288	0.0294	0.0393	0.0385	0.0376
$J = 3$						
21	0.0619	0.0456	0.0363	0.2996	0.1485	0.0845
201	0.0389	0.0362	0.0370	0.1080	0.0931	0.0978
1201	0.0376	0.0372	0.0371	0.0972	0.0960	0.0964
$J = 10$						
21	0.0339	0.0271	0.0214	0.2638	0.1415	0.0540
201	0.0226	0.0215	0.0216	0.0705	0.0563	0.0561
1201	0.0215	0.0215	0.0215	0.0569	0.0558	0.0559

The reported numbers are RMSEs relative to the true value

References

Bhat, C. (2001): "Quasi-Random Maximum Simulated Likelihood Estimation of the Mixed Multinomial Logit Model," *Transportation Research B*, 35, 677–693.

Börsch-Supan, A., and V. Hajivassiliou (1993): "Smooth Unbiased Multivariate Probability Simulators for Maximum Likelihood Estimation of Limited Dependent Variable Models," *Journal of Econometrics*, 58, 347–368.

Butler, J. S., and R. Moffit (1982): "A Computationally Efficient Quadrature Procedure for the One-Factor Multinomial Probit Model," *Econometrica*, 50(3), 761–764.

Cools, R. (2003): "An Encyclopedia of Cubature Formulas," *Journal of Complexity*, 19, 445–453.

Davis, P. J., and P. Rabinowitz (1984): *Methods of Numerical Integration*. Academic Press, New York, 2nd edn.

Genz, A., and C. Keister (1996): "Fully symmetric interpolatory rules for multiple intergrals over infinite regions with Gaussian weights," *Journal of Computational and Applied Mathematics*, 71, 299–309.

Gerstner, T., and M. Griebel (2003): "Dimension-Adaptive Tensor-Product Quadrature," *Computing*, 71, 65–87.

Geweke, J. (1996): "Monte Carlo Simulation and Numerical Integration," in *Handbook of Computational Economics Vol. 1*, ed. by H. M. Amman, D. A. Kendrick, and J. Rust, pp. 731–800. Elsevier Science, Amsterdam.

Hajivassiliou, V. A., and P. A. Ruud (1994): "Classical Estimation Methods for LDV Models Using Simulation," in *Handbook of Econometrics Vol. IV*, ed. by R. F. Engle, and D. L. McFadden, pp. 2383–2441. Elsevier, New-York.

Hess, S., K. E. Train, and J. W. Polak (2006): "On the Use of a Modified Latin Hypercube Sampling (MLHS) Method in the Estimation of a Mixed Logit Model for Vehicle Choice," *Transportation Research Part B*, 40, 147–163.

Judd, K. L. (1998): *Numerical Methods in Economics*. MIT Press, Cambridge, Mass.

Liu, Q., and D. A. Pierce (1994): "A note on Gauss–Hermite quadrature," *Biometrika*, 81, 624–629.

McFadden, D. (1989): "A Method of Simulated Moments for Estimation of Discrete Choice Models Without Numerical Integration," *Econometrica*, 57, 995–1026.

McFadden, D., and K. Train (2000): "Mixed MNL Models for Discrete Response," *Journal of Applied Econometrics*, 15, 447–470, Unveröffentlichtes Manuskript, University of California, Berkeley.

Miranda, M. J., and P. L. Fackler (2002): *Applied Computational Economics and Finance*. MIT Press, Cambridge MA.

Novak, E., and K. Ritter (1999): "Simple cubature formulas with high polynomial exactness," *Constructive Approximation*, 15, 499–522.

Patterson, T. N. L. (1968): "The optimum addition of points to quadrature formulae," *Mathematics of Computation*, 22, 847–856.

Petras, K. (2003): "Smolyak Cubature of Given Polynomial Degree with Few Nodes for Increasing Dimension," *Numerische Mathematik*, 93, 729–753.

Smolyak, S. A. (1963): "Quadrature and Interpolation Formulas for Tensor Products of Certain Classes of Functions," *Soviet Mathematics Doklady*, 4, 240–243.

Tauchen, G., and R. Hussey (1991): "Quadrature-Based Methods for Obtaining Approximate Solutions to Nonlinear Asset Pricing Models," *Econometrica*, 59(2), 371–396.

Train, K. (2003): *Discrete Choice Methods with Simulation*. Cambridge University Press.

Wasilkowski, G. W., and H. Woźniakowski (1995): "Explicit cost bounds of algorithms for multivariate tensor product problems," *Journal of Complexity*, 8, 337–392.