The Econometrics Journal

Econometrics Journal (2022), volume **25**, pp. 301–321.
https://doi.org/10.1093/ectj/utab023

Designed quadrature to approximate integrals in maximum simulated likelihood estimation

PRATEEK BANSAL[†], VAHID KESHAVARZZADEH[‡], ANGELO GUEVARA[§], SHANJUN LI[‖] AND RICARDO A. DAZIANO[¶]

[†]*Department of Civil and Environmental Engineering, Imperial College London, Exhibition Rd, South Kensington, London SW7 2BX, UK.*
Email: pb422@cornell.edu

[‡]*Scientific Computing and Imaging Institute, University of Utah, 72 Central Campus Dr, Salt Lake City, UT 84112, USA.*
Email: vkeshava@sci.utah.edu

[§]*Departamento de Ingeniería Civil, Universidad de Chile, Chile, Instituto Sistemas Complejos de Ingeniería, Av. República 701, Santiago, Región Metropolitana, Chile.*
Email: crguevar@ing.uchile.cl

[‖]*Dyson School of Applied Economics and Management, Cornell University, 137 Reservoir Ave, Ithaca, NY 14853, USA.*
Email: sl2448@cornell.edu

[¶]*School of Civil and Environmental Engineering, Cornell University, 313 Campus Rd, Ithaca, NY 14853, USA.*
Email: daziano@cornell.edu

First version received: 7 December 2020; final version accepted: 24 April 2021.

Summary: Maximum simulated likelihood estimation of mixed multinomial logit models requires evaluation of a multidimensional integral. Quasi-Monte Carlo (QMC) methods such as Halton sequences and modified Latin hypercube sampling are workhorse methods for integral approximation. Earlier studies explored the potential of sparse grid quadrature (SGQ), but SGQ suffers from negative weights. As an alternative to QMC and SGQ, we looked into the recently developed designed quadrature (DQ) method. DQ requires fewer nodes to get the same level of accuracy as QMC and SGQ, is as easy to implement, ensures positivity of weights, and can be created on any general polynomial space. We benchmarked DQ against QMC in a Monte Carlo and an empirical study. DQ outperformed QMC in all considered scenarios, is practice ready, and has potential to become the workhorse method for integral approximation.

Keywords: *Designed quadrature*, *mixed logit*, *Monte Carlo integration*, *quasi-Monte Carlo*, *sparse grid quadrature*.

JEL codes: *C6*, *C8*, *D9*.

1. INTRODUCTION

Discrete choice models are widely applied across several disciplines such as marketing, economics, and travel behaviour. The mixed multinomial logit (MMNL) model currently dominates empirical choice modelling research as it can capture unobserved preference heterogeneity of decision-makers. The multinomial probit (MNP) model is also an attractive alternative to specify flexible substitution patterns across alternatives as well as to jointly model mixed types of dependent variables (Bhat, 2015). In the maximum likelihood estimator of both MMNL and MNP models, choice probabilities involve computation of a multidimensional integral (Train, 2009). Moreover, estimating design criteria in Bayesian D-efficient designs of choice experiments also requires computation of multidimensional integrals (Yu et al., 2010).

Among a handful of analytic solutions to the multidimensional integral problem, a simulation-free maximum approximate composite marginal likelihood (MACML) estimation approach is available for MNP models (Bhat, 2011), but the Geweke–Hajivassiliou–Keane (GHK) simulator (Geweke et al., 1994) is still more commonly used in practice. In the absence of a tractable analytical solution, these integrals are generally approximated through simulation in the estimation of logit models and in creating Bayesian D-efficient designs. In general, the abovementioned estimation problems include evaluation of integrals of the following type:

$$\int_{\Gamma} f(\boldsymbol{x})\omega(\boldsymbol{x})d\boldsymbol{x} \approx \sum_{q=1}^{n} f(\boldsymbol{x}_q)w_q,$$

where Γ is a set in the d-dimensional Euclidean space $\mathbb{R}^d$, ω is a probability density function (or positive weight function), and $f(.)$ is generally a conditional likelihood function. Instead of solving the actual integral, simulation-based inference considers a discrete approximation. The objective of computationally efficient simulation is to determine nodes $\boldsymbol{x}_q$ and weights w_q so that integration can be approximated with the minimum number of function evaluations (n).

Simulation-based inference in discrete choice models started with pseudo-Monte Carlo (PMC) methods. As an alternative to PMC, quasi-Monte Carlo (QMC) methods are now typically used to approximate multidimensional integrals (Bhat, 2001; Train, 2009). More specifically, low-discrepancy sequences,[1] such as randomised and scrambled Halton sequences (Bhat, 2003), and modified Latin hypercube sampling (MLHS) (Hess et al., 2006) dominate the empirical literature. QMC methods are preferred over PMC because QMC requires fewer draws (i.e., fewer loglikelihood function counts) to approximate the integrals due to their excellent coverage properties (Bhat, 2001).

To further reduce the approximation error with fewer integrand/function evaluations, multi-level high-order[2] (MLHO) QMC approaches have emerged in the last decade (Dick et al., 2017). Most theoretical advancements in this literature are inspired by solving inverse problems in partial differential equations that require approximating high-dimensional conditional expectations (Gantner et al., 2018). In MLHO approximations, multiple levels ranging from the fewest nodes at the coarsest level to the highest nodes at the finest level are defined, and the integral at each level is approximated using a higher-order QMC method such as interlaced polynomial lattice rules (Gantner, 2016; Goda and Dick, 2015). With an optimal number of nodes at each level,

[1] Dick and Pillichshammer (2014) illustrates that the lower the discrepancy of a sequence, the smaller the error in the Monte Carlo integration will be.

[2] As the approximation error rate is proportional to $F^{-\sigma}$, where F is the number of function evaluations and σ is the order of convergence, the objective in higher-order QMC methods is to achieve the highest possible value of σ.

aggregating level-specific approximated integrals with varying precision results in overall small approximation errors. Given that sample and solution spaces are different at each level, these methods require multiple user inputs such as: (a) mapping of sample spaces at different levels for consistent sampling, (b) mapping of solution spaces to combine level-specific solutions, and (c) the number of levels. The optimal number of nodes at each level is another user input that depends on the integrand's properties and the error convergence rate of the level-specific approximation method.

Gantner (2016) has developed a user-friendly interface to implement MLHO QMC methods, but these methods did not receive much attention in applied microeconometrics. Developers of MLHO QMC methods may find them generic and simple to implement, but applied microeconometricians would not think the same way as they benchmark the simplicity of a new method against traditional Halton sequences. For instance, Sándor and Train (2004) and Munger et al. (2012) showed the superiority of single-level higher-order *digital nets* over Halton sequences, but implementation simplicity and generalisability of the latter makes it a workhorse method in applied microeconometrics.

In this study, we illustrate that recent advancements in deterministic quadrature methods could be of interest in microeconometrics because they (a) are as generic and easy to implement as low-discrepancy sequences, (b) have a provably higher order of convergence than existing higher-order random QMC methods, and (c) are superior in approximating moderate dimension integrals that are often encountered in microeconometrics (Ryu and Boyd, 2015; Keshavarzzadeh et al., 2020).[3]

1.1. Quadrature methods and research gap

As an alternative to QMC, quadrature methods have been explored in the discrete choice literature (Heiss and Winschel, 2008; Heiss, 2010; Abay, 2015; Patil et al., 2017; Goos and Mylona, 2018). Quadrature methods mainly differ from QMC in two ways, as quadrature (a) generally assumes that the integrand can be approximated on a polynomial space and (b) uses deterministic draws (or *nodes*) that carry unequal weights.

The Gaussian quadrature method approximates one-dimensional integrals with just a few nodes.[4] Quadrature can be simply extended to multiple dimensions using the tensor product. However, this multidimensional extension of quadrature suffers from the curse of dimensionality—the number of nodes (i.e., function evaluations) increases exponentially with the number of dimensions, making it impractical beyond 4–5 dimensions. Smolyak (1963) proposed a way to extend the univariate quadrature rule to multiple dimensions in a method that is often called sparse grid quadrature (SGQ) in the literature. For example, whereas Gaussian quadrature can exactly compute an integral with a univariate polynomial of order 5 with three nodes, the same function in 20 dimensions requires $3^{20} = 3,486,784,401$ nodes in product rule quadrature and 841 nodes in SGQ, respectively (Heiss and Winschel, 2008).

Heiss and Winschel (2008) have demonstrated that SGQ performs much better than QMC in the estimation of the MMNL model, even with up to 20 random parameters. Further, Heiss

[3] The advancements in deterministic quadrature methods revolve around numerical optimisation on higher-order and high-dimensional polynomials. We acknowledge the fact that such numerical optimisation can be inefficient (or might be impossible) in extremely high orders and dimensions (e.g., above 200), but dimension-independent random sampling methods can be effective in such high dimensions. However, this is not a concern in our work because the dimension of integrals in microeconometric estimations is generally below twenty.

[4] A K-times differentiable integrand can be approximated by a polynomial of degree K, and thus the resulting integral with surrogate integrand can be approximated using just $\frac{K+1}{2}$ nodes (Golub and Welsch, 1969).

(2010) combined SGQ with the efficient importance sampler (EIS) (Richard and Zhang, 2007) to estimate MNP and panel binary probit models, and demonstrated superiority of this hybrid SGQ-EIS approach over traditional QMC methods.

Even if nodes and weights in SGQ can be pre-computed and stored for reuse as easily as in traditional QMC methods, SQG methods have not been adopted in practice due to three possible reasons. First, weights computed in SGQ can be negative. Whereas Heiss and Winschel (2008) discussed this concern as an eventual possibility, they claimed not to encounter any such issue—perhaps due to a very simplistic simulation design with a (low-variance) diagonal variance–covariance matrix. In contrast, in our experience we always encountered the issue of negative choice probability estimates for a few individuals coming from negative weights, which numerically led to imaginary (complex) loglikelihood values. Patil et al. (2017) also encountered convergence issues due to negative weights while applying the SGQ-EIS method in the estimation of multinomial probit. Second, the required number of nodes to accurately approximate the integral using SGQ depends on the functional properties of the integrand, but the researcher is generally not aware of these properties. Third, whereas SGQ reduces the number of nodes significantly as compared to the product rule, the cardinality remains very high relative to that of QMC for high-dimensional integrals. Concerns two and three can be illustrated with the following example. If the integrand in a ten-dimensional integral can be well approximated using a third-order polynomial, Gaussian SGQ just needs 21 nodes, but the number of required nodes and thus the number of function evaluations increases to 8,761 for a ninth-order polynomial (Heiss and Winschel, 2008). The combined consequences of concerns two and three is confirmed by Abay (2015) in the estimation of a panel binary probit—SGQ outperforms QMC for dimensions below or equal to four, but QMC starts dominating SGQ for higher dimensions, and the difference is apparent as panel covariance increases. This is because a higher panel covariance in binary probit makes the integrand (i.e., loglikelihood) less smooth and, therefore, a higher-order polynomial (i.e., higher number of nodes or function evaluations) is required to approximate the integral at the same level of accuracy.

1.2. Moment-base quadrature and contributions

More recent developments in quadrature methods could address the main concerns of SGQ. Whereas Ryu and Boyd (2015) showed that numerical quadrature can be obtained by solving an infinite-dimensional linear program (LP), Jakeman and Narayan (2018) used the same flexible moment-based optimisation framework to obtain a numerical quadrature rule. Recently, Keshavarzzadeh et al. (2018, 2021) simplified this moment-based strategy by solving a relaxed version of the original optimisation problem and came up with a new numerical quadrature rule known as designed quadrature (DQ).

DQ has many key features. This flexible framework allows the researcher to add a constraint to ensure the positivity of weights. Moreover, DQ rules can be constructed over nonstandard geometries of the support of the nonnegative weight function and on more general polynomial spaces (e.g., hyperbolic cross-polynomial space) instead of restricting to just total order polynomial spaces. For instance, Keshavarzzadeh et al. (2018) considered the support of weight function to be ‘U-shaped’ while generating DQ. In fact, DQ requires relatively fewer nodes than SGQ. For example, to approximate a 10-dimensional integral with a polynomial of total order 5 as integrand, DQ requires 148 nodes while nested SGQ needs 201 nodes.[5]

[5] In the absence of information about functional properties of the integrand, a priori assumption on the order of the polynomial (to approximate the integrand) persists in DQ. This theoretical issue does not adversely affect the empirical

To the best of our knowledge, the potential of moment-based numerical quadrature rules has not been explored in the econometrics literature. Thus, the contribution of the study is two-fold: (a) we address the bottlenecks of the traditional SGQ method by applying the recently developed DQ method (Keshavarzzadeh et al., 2018) in maximum simulated likelihood estimation of discrete choice models; (b) using a Monte Carlo study and an empirical application, we show the superiority of DQ over workhorse QMC methods in the estimation of MMNL with a varying number of random parameters (3, 5, and 10) and correlation structures (diagonal and full covariance).

The rest of the paper is organised as follows: Section 2 briefly describes the MMNL model and its estimation, Section 3 discusses univariate quadrature, multivariate quadrature, and DQ methods, Section 4 explains the Monte Carlo simulation design and summarises corresponding results, Section 5 compares QMC methods with DQ on an empirical study, and conclusions and future work are detailed in Section 6.

2. MIXED MULTINOMIAL LOGIT MODEL

Consider that the conditional indirect utility derived by decision-maker i from making choice j in choice situation t is:

$$U_{itj} = \boldsymbol{x}_{itj}^T \boldsymbol{\alpha} + \boldsymbol{z}_{itj}^T \boldsymbol{\beta}_i + \varepsilon_{itj},$$

where $i \in \{1, \ldots, N\}$, $j \in \{1, \ldots, J\}$, and $t \in \{1, \ldots, T\}$. The covariate vector $\boldsymbol{x}_{itj}$ has a fixed preference parameter vector $\boldsymbol{\alpha}$ and $\boldsymbol{z}_{itj}$ has a random, agent-specific parameter vector $\boldsymbol{\beta}_i$. The preference shock ε_{itj} is independent across individuals, choices, and time, and is an identically distributed Type-I extreme value. Thus, the probability of choosing alternative j by individual i in choice situation t, conditional on $\boldsymbol{\beta}_i$, has a logit link:

$$P_{itj}(\boldsymbol{\alpha}, \boldsymbol{\beta}_i) = \frac{\exp\left(\boldsymbol{x}_{itj}^T \boldsymbol{\alpha} + \boldsymbol{z}_{itj}^T \boldsymbol{\beta}_i\right)}{\sum_{k=1}^{J} \exp\left(\boldsymbol{x}_{itk}^T \boldsymbol{\alpha} + \boldsymbol{z}_{itk}^T \boldsymbol{\beta}_i\right)}.$$

For an individual i who chooses alternative j in choice situation t, we define the indicator $d_{itj} = \mathbb{I}(\text{j chosen}|i, t)$. For the sequence of choices made by individual i, the conditional likelihood $\mathcal{L}_i(\boldsymbol{\alpha}, \boldsymbol{\beta}_i)$ is:

$$\mathcal{L}_i(\boldsymbol{\alpha}, \boldsymbol{\beta}_i) = \prod_{t=1}^{T} \prod_{j=1}^{J} [P_{itj}(\boldsymbol{\alpha}, \boldsymbol{\beta}_i)]^{d_{itj}}.$$

Consider that the random parameter $\boldsymbol{\beta}_i$ is multivariate normally distributed with mean $\boldsymbol{\gamma}$ and variance–covariance matrix $\boldsymbol{\Delta}$. Thus, the loglikelihood $\ell(\boldsymbol{\psi})$ of the sample in terms of the

advantages of DQ because one can generate DQ rules for the highest possible polynomial order for a given number of nodes, which can be restored and reused in future (see Section 6 for a detailed discussion). Adaptive SGQ methods are capable of handling this challenge (Ma and Zabaras, 2009; Brumm and Scheidegger, 2017; Cagnone and Bartolucci, 2017; Bhaduri and Graham-Brady, 2018; Scheidegger and Treccani, 2018). Instead of restricting to polynomial basis functions, adaptive methods generally use hierarchical basis functions to capture the integrand's local behaviour. However, adaptive SGQ methods require specification of a *grid refinement strategy*, a criterion to include or exclude a grid point based on its relative contribution to the approximation. Such grid refinement strategies need to be fine-tuned based on the integrand's properties that make adaptive SGQs less general and difficult to implement than DQ.

unconditional likelihood $P_i(\boldsymbol{\psi})$ of individual i is:

$$\ell(\boldsymbol{\psi}) = \sum_{i=1} \ln\Big(P_i(\boldsymbol{\psi})\Big) = \sum_{i=1} \ln\left(\int_{\boldsymbol{\beta}} \mathcal{L}_i(\boldsymbol{\alpha}, \boldsymbol{\beta}) f(\boldsymbol{\beta}|\boldsymbol{\gamma}, \boldsymbol{\Delta}) d\boldsymbol{\beta}\right), \tag{2.1}$$

where $\boldsymbol{\psi} = \{\boldsymbol{\alpha}, \boldsymbol{\gamma}, \boldsymbol{\Delta}\}$.

As the sample loglikelihood $\ell(\cdot)$ in (2.1) is analytically intractable, the parameter vector $\boldsymbol{\psi}$ can be estimated by maximising the sample's simulated loglikelihood $\widetilde{\ell}(\cdot)$:

$$\widetilde{\ell}(\boldsymbol{\psi}) = \sum_{i=1}^{N} \ln\left(\sum_{r=1}^{R} \mathcal{L}_i(\boldsymbol{\alpha}, \boldsymbol{\beta}_{ir}) w_i(\boldsymbol{\beta}_{ir}|\boldsymbol{\gamma}, \boldsymbol{\Delta})\right).$$

Note that $\boldsymbol{\beta}_{ir}$ and $w_i(\boldsymbol{\beta}_{ir}|\boldsymbol{\gamma}, \boldsymbol{\Delta})$ are viewed as nodes and weights in the quadrature method, respectively. In the QMC simulation literature, nodes are generally denoted by draws and the weight $w_i(\boldsymbol{\beta}_{ir}|\boldsymbol{\gamma}, \boldsymbol{\Delta})$ attains the value of $\frac{1}{R}$ for all draws. Note that even though $\boldsymbol{\beta}_{ir}$ is a realisation of $\mathcal{N}(\boldsymbol{\gamma}, \boldsymbol{\Delta})$, the model is reparametrised in terms of the Cholesky decomposition of $\boldsymbol{\Delta}$ to ensure positive definiteness. Thus, when approximating the loglikelihood with quadrature or QMC methods, we always work with standard normal distributions.

3. QUADRATURE METHODS

3.1. Notation

We adopt the notation of Keshavarzzadeh et al. (2018) to illustrate the intuition and key results of different quadrature methods. We reconsider the integral approximation problem:

$$\int_{\Gamma} f(\boldsymbol{x})\omega(\boldsymbol{x})d\boldsymbol{x} \approx \sum_{q=1}^{n} f(\boldsymbol{x}_q) w_q, \tag{3.1}$$

where $\omega(\boldsymbol{x})$ is a given weight function (or a probability density function) whose support is $\Gamma \subset \mathbb{R}^d$. A point $\boldsymbol{x} \in \mathbb{R}^d$ has components $\boldsymbol{x} = \left(x^{(1)}, x^{(2)}, \ldots, x^{(d)}\right)$.

We define $\boldsymbol{\alpha} \in \mathbb{N}_0^d$ as a multi-index, and Λ as a downward closed set[6] of multi-indices:

$$\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_d), \qquad \boldsymbol{x}^{\boldsymbol{\alpha}} = \prod_{j=1}^{d} \left(x^{(j)}\right)^{\alpha_j}, \qquad |\boldsymbol{\alpha}| = \sum_{j=1}^{d} \alpha_j.$$

Our ultimate goal is to construct a set of n points $\left\{\boldsymbol{x}_q\right\}_{q=1}^{n} \subset \Gamma$ and positive weights $w_q > 0$ in (3.1), but we attempt to achieve this by enforcing equality in (3.1) for a subspace Π of polynomials such that:

$$\begin{aligned} &\int_{\Gamma} f(\boldsymbol{x})\omega(\boldsymbol{x})d\boldsymbol{x} = \sum_{q=1}^{n} f(\boldsymbol{x}_q) w_q, \quad f \in \Pi \\ &\Pi = \operatorname{span}\left\{\boldsymbol{x}^{\boldsymbol{\alpha}} \;\middle|\; \boldsymbol{\alpha} \in \Lambda\right\}. \end{aligned} \tag{3.2}$$

Thus, solving for $\left\{\boldsymbol{x}_q\right\}_{q=1}^{n}$ and $w_q > 0$ using (3.2) should provide a good approximation of the integral in (3.1).

[6] If $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{N}_0^d$, then $\boldsymbol{\alpha} \leq \boldsymbol{\beta}$ if and only if all component-wise inequalities are true. Using this definition, a multi-index set Λ is called *downward closed* if $\boldsymbol{\alpha} \in \Lambda \implies \boldsymbol{\beta} \in \Lambda \quad \forall\, \boldsymbol{\beta} \leq \boldsymbol{\alpha}$.

Whereas Keshavarzzadeh et al. (2018) proposed a numerical method to solve (3.2) for a general polynomial subspaces, we restrict discussion to *total order* (represented by subscript $\mathcal{T}_{(.)}$) polynomial spaces with the total order being r:

$$\Pi_{\mathcal{T}_r} = \text{span}\left\{\boldsymbol{x}^{\boldsymbol{\alpha}} \;\middle|\; \boldsymbol{\alpha} \in \Lambda_{\mathcal{T}_r}\right\}, \quad \text{where } \Lambda_{\mathcal{T}_r} = \left\{\boldsymbol{\alpha} \in \mathbb{N}_0^d \;\middle|\; |\boldsymbol{\alpha}| \leq r\right\}.$$

3.2. Univariate quadrature

We need to define first the basis for the polynomial space $\Pi_{\mathcal{T}_k}$. Note that a basis of orthonormal polynomials exists with elements $p_m(\cdot)$ such that $\deg p_m = m$. The family of these polynomials satisfies the following recursive relation (Askey, 1975):

$$xp_m(x) = \sqrt{b_m}\,p_{m-1}(x) + a_m\,p_m(x) + \sqrt{b_{m+1}}\,p_{m+1}(x),$$

$$a_m = (xp_m,\, p_m) \quad b_m = \frac{(p_m,\, p_m)}{(p_{m-1},\, p_{m-1})}.$$

After characterising the one-dimension polynomial space, we present a theorem which is the foundation of the quadrature literature:

THEOREM 3.1. *Let $x_1, \ldots, x_n$ be the roots of the nth orthogonal polynomial $p_n(x)$ and let $w_1, \ldots, w_n$ be the solution of the system of equations*

$$\sum_{q=1}^{n} p_j(x_q)w_q = \begin{cases} \sqrt{b_0}, & \text{if } j = 0 \\ 0, & \text{for } j = 1, \ldots, n-1. \end{cases} \tag{3.3}$$

Then $x_q \in \Gamma$ and $w_q > 0$ for $q = 1, 2, \ldots, n$, and

$$\int_{\Gamma} p(x)\omega(x)dx = \sum_{q=1}^{n} p(x_q)w_q \tag{3.4}$$

holds for all polynomials $p \in \Pi_{\mathcal{T}_{2n-1}}$.

According to Theorem 3.1, nodes and weights in (3.4)—which is a one-dimensional version of (3.2)—can be exactly obtained by solving the system of equations (moment-matching conditions) summarised in (3.3). A more intuitive implication of Theorem 3.1 is that if the integrand can be exactly specified on a polynomial space of order $2n-1$, only n nodes are required to compute the corresponding univariate integral precisely. Golub and Welsch (1969) and Davis and Rabinowitz (2007) provide a detailed procedure to compute this univariate quadrature rule.

3.3. Multivariate quadrature

In product rules, univariate quadrature can be simply extended to multivariate quadrature using a tensor product. More specifically, the weight function $\omega(\boldsymbol{x})$ and its support Γ can be written as follows:

$$\Gamma = \times_{j=1}^{d}\Gamma_j, \qquad \omega(\boldsymbol{x}) = \prod_{j=1}^{d}\omega_j\left(x^{(j)}\right),$$

where $\Gamma_j \subset \mathbb{R}$ is a univariate domain and $\omega_j(\cdot)$ is a univariate weight. If $p_n^{(j)}(\cdot)$ is the univariate orthonormal polynomial family corresponding to ω_j over Γ_j, then the family of multivariate

polynomials orthonormal under ω can be written as:

$$\pi_{\boldsymbol{\alpha}}(\boldsymbol{x}) = \prod_{j=1}^{d} p_{\alpha_j}^{(j)}\left(x^{(j)}\right), \qquad \boldsymbol{\alpha} \in \mathbb{N}_0^d.$$

The corresponding polynomial space is: $\Pi_{\mathcal{T}_r} = \text{span}\left\{\pi_{\boldsymbol{\alpha}} \;\middle|\; \boldsymbol{\alpha} \in \Lambda_{\mathcal{T}_r}\right\}$. After characterising the polynomial subspace, the moment-matching conditions of Theorem 3.1 can be extended to the multivariate case as follows:

PROPOSITION 3.1. *Let Λ be a multi-index set with $\mathbf{0} \in \Lambda$. Suppose that $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ and $w_1, \ldots, w_n$ are the solution of the system of equations*

$$\sum_{q=1}^{n} \pi_{\boldsymbol{\alpha}}(\boldsymbol{x}_q) w_q = \begin{cases} 1/\pi_{\mathbf{0}}, & \text{if } \boldsymbol{\alpha} = \mathbf{0} \\ 0, & \text{if } \boldsymbol{\alpha} \in \Lambda \backslash \{\mathbf{0}\} \end{cases} \tag{3.5}$$

then

$$\int_{\boldsymbol{\Gamma}} \omega(\boldsymbol{x}) \pi(\boldsymbol{x}) d\boldsymbol{x} = \sum_{q=1}^{n} \pi(\boldsymbol{x}_q) w_q,$$

holds for all polynomials $\pi \in \Pi_{\Lambda}$.

Note that unlike Theorem 3.1, Proposition 3.1 neither guarantees the positivity of weights nor ensures that nodes belong to the support Γ. Although sparse grid quadrature (SGQ) provides an efficient way to combine multiple dimensions so as to reduce the function evaluations, it does not provide remedy for these issues. We did not consider SGQ in this study, because (a) in our initial test runs, negative weights in SGQ led to complex (imaginary) loglikelihood values in the estimation of MMNL with full variance–covariance matrix; and (b) based on extensive simulations studies, Keshavarzzadeh et al. (2018) confirmed that DQ requires many fewer nodes than SGQ. Heiss and Winschel (2008) can be referred for intuitive and theoretical discussion on SGQ rules.

3.4. Designed quadrature (DQ)

DQ solves a relaxed version of the moment-matching conditions given in (3.5), which enforces positivity of weights and also ensure nodes to fall in the support of the probability density function. Keshavarzzadeh et al. (2018) reformulates the moment-matching conditions as follows:

For a given index set Λ with size $M = |\Lambda|$, consider the matrix $\boldsymbol{X} \in \mathbb{R}^{d \times n}$ with columns $\boldsymbol{x}_j$, and let $\boldsymbol{w} \in \mathbb{R}^n$ be a vector containing the n weights. Let $\boldsymbol{V}(\boldsymbol{X}) \in \mathbb{R}^{M \times n}$ denote the Vandermonde-like matrix with entries

$$(V)_{k,j} = \pi_{\boldsymbol{\alpha}(k)}\left(\boldsymbol{x}_j\right), \quad k = 1, \ldots, M \quad j = 1, \ldots, n,$$

where elements of Λ are considered with ordering $\boldsymbol{\alpha}(1), \ldots \boldsymbol{\alpha}(M)$ and $\boldsymbol{\alpha}(1) = \mathbf{0}$. The system (3.5) can then be written as:

$$\boldsymbol{V}(\boldsymbol{X})\, \boldsymbol{w} = \boldsymbol{e}_1/\pi_{\mathbf{0}}, \tag{3.6}$$

where $\boldsymbol{e}_1 = (1, 0, 0, \ldots, 0)^T \in \mathbb{R}^M$. Instead of solving the moment-matching conditions exactly in (3.6), Keshavarzzadeh et al. (2018) proposed to obtain the approximate solution $(\boldsymbol{X}, \boldsymbol{w})$ that

satisfies:

$$\|\boldsymbol{V}(\boldsymbol{X})\,\boldsymbol{w} - \boldsymbol{e}_1/\pi_{\boldsymbol{0}}\|_2 = \epsilon \geq 0. \tag{3.7}$$

In fact, Keshavarzzadeh et al. (2018) provide bounds on the integral error $\left|\int f(\boldsymbol{x})\omega(\boldsymbol{x})d\boldsymbol{x} - \sum_{q=1}^{n} f(\boldsymbol{x}_q)w_q\right|$ in terms of tolerance ϵ, which is computable for a given quadrature rule.

Thus, for a given polynomial subspace, DQ aims to compute nodes $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \in \Gamma^n$ and positive weights $\boldsymbol{w} \in (0, \infty)^n$ that solves the following constrained optimisation problem:

$$\begin{aligned} \min_{\boldsymbol{X},\boldsymbol{w}} \; & ||\boldsymbol{V}(\boldsymbol{X})\boldsymbol{w} - \boldsymbol{e}_1/\pi_{\boldsymbol{0}}||_2 \\ \text{subject to } & \boldsymbol{x}_j \in \Gamma, \qquad\qquad j = 1, \ldots, n \\ & w_j > \boldsymbol{0}, \qquad\qquad j = 1, \ldots, n. \end{aligned} \tag{3.8}$$

Readers can refer to Keshavarzzadeh et al. (2018) for more insights about strategies (e.g., constrained optimisation problem) to solve the above optimisation problem.

3.5. Discussion

In the context of this study, we explore possibilities of approximating the unconditional choice probability integral, as in (3.1), in MMNL using DQ. We assume that the conditional choice probability—integrand in (3.1)—can be approximated on total order polynomial space. Given that properties of the integrand vary with the data generating process and are thus not known beforehand, the performance of the approximation will depend on the assumed order of the polynomial space. Moreover, whereas SGQ predetermines the exact number of nodes based on the order (r) of the polynomial space and dimension of the integral, DQ rules can be obtained (i.e., the optimisation problem in (3.8) can be solved) for a various possible number of nodes (n). Thus, for a given integral dimension, one can generate DQ rules for different total order (r) polynomial spaces and a different number of nodes (n).

In both parametric and nonparametric MMNL models, the choice probability integral can generally be reparameterised such that the weight function $\omega(.)$ in DQ turns out to be a probability density function of a standard normal (e.g., normal or lognormal mixing distributions) or a standard uniform distribution (e.g., semiparametric logit-mixed logit model).[7] Just as in QMC methods, we can generate, store, and reuse DQ rules for both standardised distributions. Thus, DQ offers the same flexibility. The researcher can solve the optimisation problem in (3.8) beforehand for different combinations of dimensions, order of polynomial, and number of nodes, and then reuse the stored nodes and weights.

In sum, DQ may appear more cumbersome than QMC methods at first, but reusability of the nodes and weights not only makes DQ equally easy to implement in practice and, in fact, even fewer function evaluations are needed (i.e., lower computation time is achieved). Nevertheless, for a given dimension of integral, whereas QMC needs tuning of the number of draws to get stable parameter estimates, DQ requires to tune the total order of polynomial spaces and the corresponding number of draws. In the next section we conduct a detailed simulation study to make recommendations about selection of these parameters in the context of MMNL.

[7] The support Γ of standard normal and standard distributions are whole real line and [0,1], respectively.

4. MONTE CARLO STUDY

4.1. Simulation design

The objective of the simulation study is to evaluate the performance of DQ relative to QMC methods in MMNL estimation. We considered modified Latin hypercube sampling (MLHS) (Hess et al., 2006) and randomised and scrambled Halton sequences (Bhat, 2003) as representative QMC methods. In the data generating process (DGP), we considered a sample of 1,000 decision-makers who are assumed to choose a utility maximising alternative from a set of five alternatives across five choice situations. As the number of random parameters governs the dimension of the choice probability integral in MMNL, we compared performance of DQ and QMC methods in MMNL with three, five, and ten normally distributed random parameters. For each random parameter scenario, we considered two covariance structures: zero (diagonal) and full covariance across random parameters. The considered covariance matrix $\mathbf{\Delta}^5_{full\ cov.}$ for five random parameters is illustrated below; similar structures were considered for dimensions three and ten. This sensitivity analysis is crucial because the performance of DQ depends on the smoothness of the integrand (i.e., conditional likelihood), which in turn depends on the structure of the covariance matrix.

$$\mathbf{\Delta}^5_{full\ cov.} = \begin{bmatrix} 1.5 & 0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & 1 & 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & 1 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0.5 & 1 & 0.5 \\ 0.5 & 0.5 & 0.5 & 0.5 & 1.5 \end{bmatrix}.$$

We generated 150 data sets in total for each covariance structure: 50 data sets for each random parameter scenario. For each of 300 data sets, we performed maximum simulated likelihood estimation (with analytical gradient) using a different number of QMC draws and different total order polynomial subspaces and nodes of DQ. We summarise results by computing the following five metrics across resamples: (a) average *loglikelihood* at convergence, (b) absolute percentage bias (APB), (c) average estimation time, (d) average number of loglikelihood evaluations in the estimation process, and (e) the *t-distributed test statistic* under the null hypothesis that the point estimate is equal to the true population parameter. As the test statistic gets smaller, we become more confident that the estimated parameter is close to the population parameter.

We compute the APB and the *t-distributed test statistic* of a parameter for a sample as follows:

$$\text{APB} = \left| \frac{\text{Parameter Estimate} - \text{True Parameter Value}}{\text{True Parameter Value}} \right| \times 100,$$

$$\text{Test statistic} = \frac{\text{Mean of the Point Estimate across Resamples} - \text{True Parameter Value}}{\text{Finite sample standard error (FSSE)}}.$$

The mean values of APB and the test statistic across all parameters and resamples are reported for succinctness. To avoid empirical identification issues, we computed FSSE, APB, and *t*-value for parameter ratios. We wrote MATLAB code to generate DQ rules and perform MMNL estimation with analytical gradients. DQ rules were generated beforehand, stored, and reused for estimation. While generating DQ rules, we considered tolerance ϵ in (3.7) to be 10^{-8}. We performed sensitivity analysis with tighter tolerances, but those did not improve accuracy. The simulation study was conducted using MATLAB (R2020b) on a Dell computer with an Intel Core i-7 (8th generation, 1.99 GHz) processor, 16 GB RAM, and 64-bit operating system. We did not use any parallel computation explicitly, but some built-in MATLAB functions inherently

use multiple cores. MATLAB codes to replicate the Monte Carlo study and generate DQ rules can be found at our GitHub repository: https://github.com/p-bansal19/Designed-Quadrature.

4.2. Results and discussion

The results of the Monte Carlo study for random parameters (integral dimensions) three, five, and ten are summarised in Tables 1, 2, and 3, respectively.

As the dimension of the integral increases, the minimum number of nodes required to generate the appropriate DQ rule at a given polynomial order (r; also known as accuracy level) increases. For example, we could generate the DQ rule for higher-order $r = 6$ with just 30 nodes for three dimensions (see Table 1), but to solve the DQ optimisation problem (up to a prespecified tolerance ϵ) for the same order in five dimensions needed at least 100 nodes (see Table 2). Also, for a given dimension of the integral, more nodes are required to generate DQ in higher-order polynomial spaces. For example, we could generate the DQ rule for ten dimensions with 100 nodes for a polynomial of order $r = 4$, but needed a minimum of 200 nodes for $r = 5$ (see Table 3).

We now compare the model fit (loglikelihood) of DQ and QMC methods. In the diagonal variance–covariance case, DQ outperformed both QMC methods by a significant margin, even when the DQ rule was generated on polynomial spaces with relatively low order. For the five-dimensional case, DQ achieved a similar model fit (loglikelihood: −5354.6) with just 100 nodes at $r = 6$ as of the fit obtained using 1,000 Halton and MLHS draws (loglikelihood: −5353.8 and −5353.9). In fact, DQ with 100 nodes at $r = 4$ (loglikelihood: −5017.7) outperformed Halton and MLHS with 300 draws (loglikelihood: −5021.6 and −5020.9) in approximating the higher (i.e., ten) dimensional integral. The model fit values of Halton and MLHS are indistinguishable.

DQ also outperformed QMC methods in the full variance–covariance scenario, but a higher order of polynomial subspaces are desirable in this nonindependent case. These observations are aligned with intuition: introducing covariance makes the integrand more complex (Abay, 2015), which can be better approximated on higher-order polynomial subspaces. For example, in the case of five random parameters with a full covariance DGP, whereas DQ could achieve a model fit of −5759.5 with 200 nodes at $r = 7$, MLHS and Halton required 300 draws to achieve virtually the same model fit; however, 300 nodes of DQ at $r = 5$ (loglikelihood: −5061.0) were slightly outperformed by 300 Halton and MLHS draws (loglikelihood: −5758.4 and −5758.6). As expected, we generally observed that increasing the order of polynomial subspaces results in a better model fit. As a general trend, across all dimensions and covariance structures, the highest order in DQ ($r = 7$, 7, and 5 for dimensions 3, 5, and 10) resulted in better model fit than those of QMC methods at a given number of draws.

Across DQ and QMC methods, parameter recovery metrics—APB—decreases with an increase in the number of draws (or nodes). Consistent with the model fit, DQ surpassed MLHS by a significant margin in recovering true parameters if the variance–covariance matrix is diagonal. For instance, DQ could achieve similar APB value with 200 nodes (on polynomial space of order $r = 6$) compared to what we obtained using QMC with 1,000 draws in the DGP with five random parameters (see Table 2). In fact, DQ also recovered parameters better than QMC methods across correlated covariance structures, but at higher-order polynomial subspaces. For example, in DGP with five highly correlated random parameters, APB using 200 QMC draws is 15.8%, but for the same number of DQ nodes whereas APB is relatively higher at $r = 5$ (16.8%), it is relatively lower at $r = 7$ (13.9%) (see Table 2). In Figure 1, we also plot APB versus computation time for all considered methods across all simulation settings to illustrate the APB reduction in DQ relative

Table 1. Comparison of designed quadrature and quasi-Monte Carlo sequences (Monte Carlo study, random parameters = 3).

	(−)Loglikelihood				Absolute percentage bias				Estimation time (seconds)				Loglikelihood function counts				*t*-value			
Draws	Halton	MLHS	DQ		Halton	MLHS	DQ		Halton	MLHS	DQ		Halton	MLHS	DQ		Halton	MLHS	DQ	
			$r = 6$	$r = 7$			$r = 6$	$r = 7$			$r = 6$	$r = 7$			$r = 6$	$r = 7$			$r = 6$	$r = 7$
Diagonal																				
30	5752.9	5749.8	5725.3		7.1	7.7	5.2		2.4	2.3	1.9		24	24	23		0.31	0.28	0.37	
50	5739.7	5740.7	5725.1	5724.5	6.4	6.9	5.2	5.2	3.8	3.7	3.4	3.4	24	24	23	23	0.22	0.24	0.32	0.34
100	5732.7	5731.4	5724.8	5724.4	5.4	6.1	5.1	5.2	7.3	7.2	6.9	6.7	23	24	24	23	0.32	0.22	0.33	0.33
1,000	5724.5	5724.6			5.2	5.2			73.3	73.9			23	23			0.34	0.33		
Full covariance																				
30	5921.0	5916.0	5860.4		15.1	14.5	13.0		2.2	2.0	1.6		22	21	20		0.53	0.42	0.26	
50	5865.4	5863.2	5849.0	5835.0	14.1	12.6	11.5	11.4	3.5	3.5	3.0	2.9	23	23	21	21	0.37	0.37	0.36	0.19
100	5827.5	5827.1	5827.9	5810.7	10.8	10.5	10.7	9.3	7.0	7.1	6.4	6.4	24	24	23	23	0.31	0.20	0.28	0.22
1,000	5794.5	5794.7			8.7	9.2			73.0	73.0			25	24			0.21	0.18		

Table 2. Comparison of designed quadrature (DQ) and quasi-Monte Carlo (QMC) sequences (Monte Carlo study, random parameters = 5).

	(−) Loglikelihood					Absolute percentage bias					Estimation time (seconds)					Loglikelihood function counts					*t*-value				
Draws	Halton	MLHS	DQ			Halton	MLHS	DQ			Halton	MLHS	DQ			Halton	MLHS	DQ			Halton	MLHS	DQ		
			$r=5$	$r=6$	$r=7$			$r=5$	$r=6$	$r=7$			$r=5$	$r=6$	$r=7$			$r=5$	$r=6$	$r=7$			$r=5$	$r=6$	$r=7$
Diagonal																									
50	5390.8	5388.1	5359.4			8.8	9.5	6.1			11	11	10			33	34	30			0.36	0.28	0.38		
100	5371.7	5371.8	5355.5	5354.6		7.2	6.8	5.6	5.5		20	20	20	19		31	31	30	30		0.28	0.33	0.28	0.28	
200	5363.3	5362.5	5354.6	5353.2	5352.8	6.2	5.8	5.5	5.3	5.4	41	39	39	38	38	32	30	30	30	29	0.20	0.26	0.26	0.23	0.23
300	5359.0	5359.1	5354.0	5352.2	5352.3	5.9	5.9	5.5	5.3	5.2	60	60	57	57	57	30	30	29	30	30	0.23	0.24	0.24	0.23	0.21
1,000	5353.8	5353.9				5.2	5.3				196	195				30	29				0.21	0.21			
Full covariance																									
50	5895.5	5886.2	5840.3			23.4	24.7	18.9			12	11	10			30	28	25			0.33	0.36	0.57		
100	5817.8	5819.2	5785.6	5794.0		18.3	18.5	17.8	16.9		23	23	22	21		30	30	29	28		0.21	0.24	0.32	0.29	
200	5773.7	5772.5	5772.5	5761.3	5759.5	15.9	15.8	16.8	15.0	13.9	46	45	44	43	44	30	30	29	28	29	0.16	0.14	0.26	0.16	0.24
300	5758.4	5758.6	5761.0	5749.5	5743.5	14.8	14.9	15.8	14.2	12.9	72	71	68	65	66	31	31	30	29	29	0.15	0.13	0.21	0.17	0.16
1,000	5733.1	5733.0				12.5	12.7				238	231				31	30				0.13	0.11			

Table 3. Comparison of designed quadrature (DQ) and quasi-Monte Carlo (QMC) sequences (Monte Carlo study, random parameters = 10).

Draws	(−)Loglikelihood				Absolute percentage bias				Estimation time (seconds)				Loglikelihood function counts				*t*-value			
	Halton	MLHS	DQ		Halton	MLHS	DQ		Halton	MLHS	DQ		Halton	MLHS	DQ		Halton	MLHS	DQ	
			$r=3$	$r=4$	$r=5$			$r=3$	$r=4$	$r=5$			$r=3$	$r=4$	$r=5$				$r=3$	$r=4$	$r=5$
Diagonal																					
100	5035.3	5034.3	5022.5	5017.7		23.5	23.2	17.7	14.4		248	237	225	226		99	89	84	83		0.58	0.53	0.50	0.45	
200	5026.1	5024.7	5019.6	5015.1	5015.5	17.7	17.3	14.9	12.5	11.8	451	448	437	421	422	84	85	81	75	76	0.46	0.40	0.42	0.38	0.36
300	5021.6	5020.9	5018.4	5014.7		15.8	16.2	14.7	10.9		667	669	673	605		83	85	84	71		0.38	0.39	0.43	0.30	
Full covariance																									
100	5914.1	5912.3	5895.7	5881.4		85.9	85.9	80.1	75.8		236	235	230	228		87	86	87	84		0.36	0.36	0.37	0.30	
200	5875.6	5873.1	5867.5	5858.7	5859.0	70.8	74.3	69.1	69.1	65.9	474	486	431	443	430	87	91	76	80	77	0.28	0.28	0.26	0.25	0.25
300	5857.1	5856.2	5863.0	5847.1		66.7	64.0	63.0	54.6		702	684	661	659		87	81	80	78		0.24	0.22	0.28	0.20	

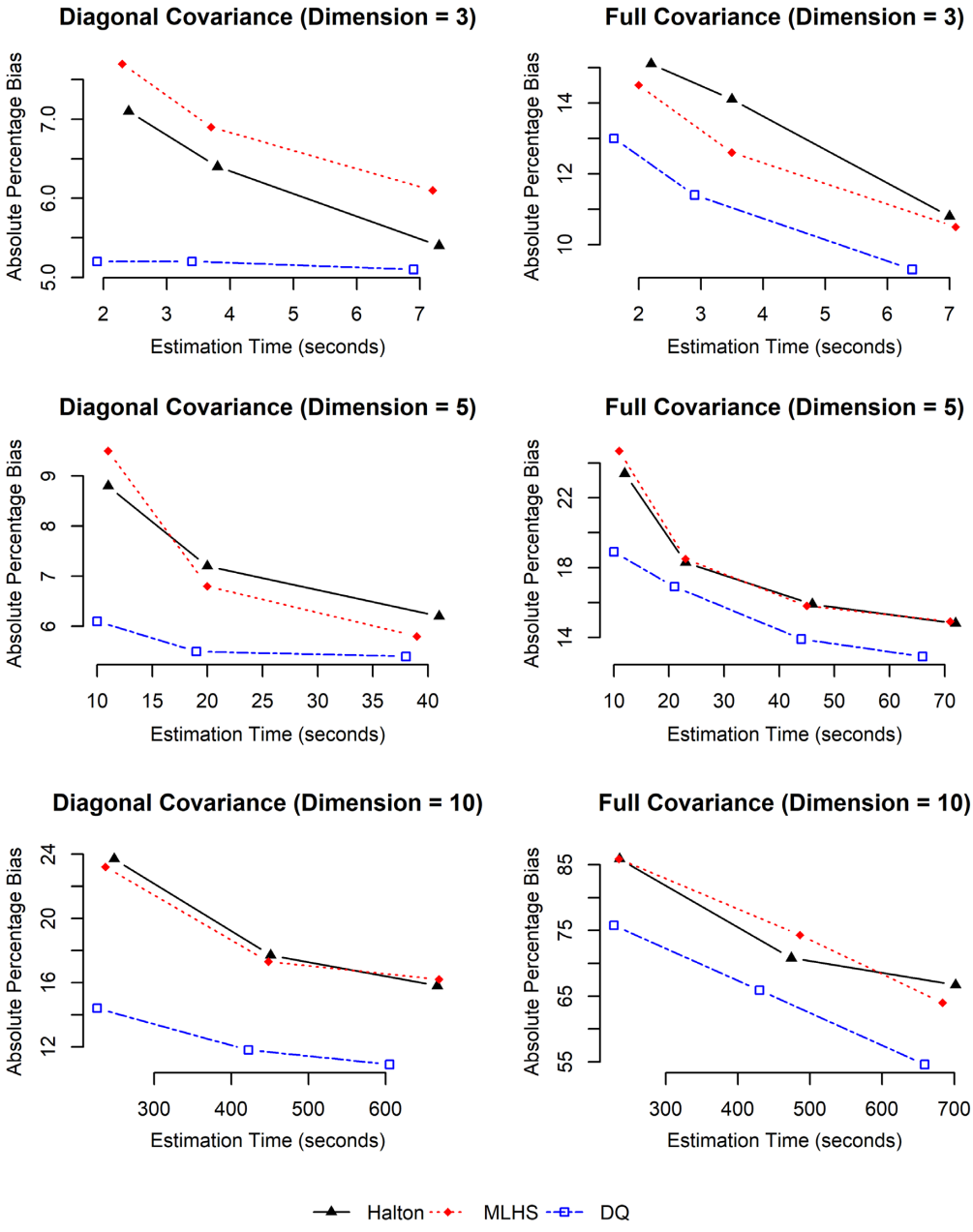

Figure 1. Bias and computation time trade-off (MLHS: modified Latin hypercube sampling; DQ: designed quadrature with the highest possible order).

to QMC methods at a given computational budget. We derived similar insights by comparing *t*-distributed test statistic of DQ and QMC methods across the considered DGP scenario.

One may speculate that DQ might take more evaluations of the loglikelihood than those required in QMC to achieve convergence, leading to higher estimation time than QMC for the same number of draws/nodes. To test this hypothesis, we present the average estimation time, and loglikelihood function evaluation across all resamples. The results indicate that DQ requires the same or fewer loglikelihood evaluations than those needed in QMC methods. Given that each loglikelihood evaluation takes the same time in both DQ and QMC for a given number of draws, the estimation time of DQ is similar or lower than that of QMC methods. Thus, the superiority of DQ in terms of model fit and parameter recovery directly translates into computational time savings.

In sum, for a given number of draws/nodes, better model fit, and more precise parameter recovery in DQ across all dimensions and covariance structures make it a computationally-efficient substitute to QMC methods in practice.

5. EMPIRICAL STUDY

We now compare the performance of DQ and QMC while studying the preference of travellers in New York City (NYC) for mobility-on-demand (MoD) services (e.g., Uber and UberPool).

5.1. Experiment design

We conducted a stated preference survey in NYC. The survey included a discrete choice experiment (DCE) in which each respondent was asked to choose the best and the worst travel mode from a set of three choices: Uber (without ridesharing), UberPool (with ridesharing), and their current travel mode (the one used most often on their most frequent trips). We first conducted a pilot study ($N = 298$) using the D-efficient design with zero priors in February 2017. We then used prior parameter estimates from the pilot study to create a pivot-efficient design[8] with six blocks (seven choice situations per block). Table 4 shows the attribute levels of the DCE design and an instance of choice situation. More details about the experiment design can be found in Liu et al. (2019). We conducted the main study during October–November 2017. Preferences of 1,000 respondents were used in estimation.

5.2. Estimation and results

We considered marginal utilities of all five alternative-specific variables to be normally distributed. This specification led to a five dimensional integral in MMNL estimation. Similar to the simulation study, we compared DQ against randomised and scrambled Halton draws and MLHS in this empirical study. The number of draws/nodes was varied from 100 to 1,000. We considered 50 different starting values, and for each starting value 13 models were estimated considering different QMC draws and DQ nodes.

Table 5 summarises the average of model fit, estimation time, and loglikelihood function evaluations across different starting values. Across all considered scenarios, MLHS resulted in a similar model fit as the one obtained using Halton draws. The performance of DQ is consistent

[8] In pivot-efficient designs, attribute levels shown to the respondents are pivoted from reference alternatives for each respondent. In this study, the travel mode used on the most frequent trips was considered as the reference alternative.

Table 4. Experiment design for mode choice study.

Attribute levels in the experiment design			
	Uber (without ridesharing)	UberPool (with ridesharing)	Current mode
Walking and waiting time	25%, 50%, 75%, 100%	25%, 50%, 75%, 100%	asked (100%)
In-vehicle travel time	80%, 95%, 110%, 125%	90%, 105%, 120%,135%	asked (100%)
Trip cost per mile ($) (excluding parking cost)	0.55, 0.70, 0.85, 1.0, 1.2	0.45, 0.60, 0.70, 0.80	asked or computed
Parking cost	0	0	asked
Powertrain	Gas, electric	Gas, electric	Gas
Automation	Yes, no	Yes, no	No
Instance of a choice situation			
	Uber (without ridesharing)	UberPool (with ridesharing)	Current mode: car
Walking and waiting time	6 minutes	9 minutes	12 minutes
In-vehicle travel time	38 minutes	50 minutes	48 minutes
Trip cost (excluding parking cost)	$11	$8	$6
Parking cost	–	–	$6
Powertrain	Electric	Gas	Gas
Automation	Service with driver	Automated (no driver)	–

Note: All % are relative to the reference alternative.

Table 5. Comparison of model fit and computational efficiency in the case study.

	(−)Loglikelihood				Estimation time (seconds)				Loglikelihood function counts			
Draws	Halton	MLHS	DQ		Halton	MLHS	DQ		Halton	MLHS	DQ	
			$r=5$	$r=6$			$r=5$	$r=6$			$r=5$	$r=6$
100	5470.3	5470.6	5445.5		46	47	48		87	94	96	
200	5459.9	5462.6	5491.5	5445.2	90	86	87	85	84	86	84	84
300	5455.7	5457.8	5457.3	5437.2	138	134	142	136	89	89	93	89
1,000	5453.0	5453.3			449	449			85	86		

with the Monte Carlo study—whereas QMC methods dominated DQ when rules were generated at the lower order $r = 5$, DQ generated at the higher-order $r = 6$ always outperformed QMC methods across all considered draws. In fact, 200 nodes in DQ at order $r = 6$ could achieve better model fit (loglikelihood: −5445.2) than those of 1,000 Halton (loglikelihood: −5453.0) or MLHS (loglikelihood: −5453.3) draws. These gains directly translate into computational efficiency as DQ and QMC methods take a similar number of loglikelihood function evaluations to achieve convergence.

Table 6 shows the average parameter estimates and z-scores for selected QMC draws and DQ nodes. The mean estimates of all three methods are similar and in fact, z-score values are also in a similar range. The Cholesky components (e.g., L22 and L33) which are statistically significant in QMC remains significant in DQ and as expected, corresponding point estimates are also more stable across the considered draws.

Table 6. Comparison of estimates and standard errors in the case study.

	Estimates						z-scores					
	Halton		MLHS		DQ (r=6)		Halton		MLHS		DQ (r=6)	
Covariates	200	1,000	200	1,000	200	300	200	1,000	200	1,000	200	300
Mean												
OVTT/100 (minute)	−1.5	−1.6	−1.7	−1.6	−1.8	−1.5	−3.1	−3.0	−3.4	−3.0	−4.0	−3.0
IVTT/100 (minute)	−10.8	−11.0	−10.8	−11.0	−11.0	−11.3	−14.9	−14.5	−14.8	−14.5	−15.3	−14.4
Trip cost/10 ($)	−3.1	−3.1	−3.1	−3.1	−3.5	−3.4	−15.3	−14.4	−15.1	−14.3	−16.0	−15.4
Electric?	−0.4	−0.4	−0.4	−0.4	−0.4	−0.4	−5.9	−5.7	−5.8	−5.7	−5.8	−5.7
Automation?	−0.5	−0.5	−0.5	−0.5	−0.5	−0.5	−6.5	−6.4	−6.5	−6.4	−6.3	−6.2
Cholesky components												
L11	0.7	0.0	0.4	0.3	0.2	−1.3	1.0	−0.1	0.7	0.5	0.1	−1.6
L21	0.5	0.1	1.1	0.1	2.6	1.3	0.5	0.1	1.1	0.0	2.4	1.2
L22	9.7	10.1	9.1	10.6	9.9	7.9	12.0	11.9	11.3	12.4	11.3	10.4
L31	0.3	0.0	0.3	0.1	1.0	0.2	1.6	0.1	1.3	0.5	5.1	0.1
L32	1.4	1.4	1.4	1.5	2.0	2.5	8.6	6.3	8.0	6.8	8.7	12.2
L33	2.9	3.2	2.9	2.8	3.1	2.5	13.5	13.2	13.1	12.0	14.4	10.0
L41	−0.1	0.0	0.0	0.0	0.1	0.3	−0.6	0.0	−0.3	−0.4	1.1	2.7
L42	0.2	0.2	0.2	0.2	0.1	0.3	1.4	1.6	1.6	1.6	1.0	2.3
L43	0.5	0.4	0.5	0.4	0.5	0.4	4.0	3.0	4.6	2.9	5.0	3.0
L44	−0.1	0.2	0.1	−0.1	0.1	0.2	−0.5	1.1	0.8	−0.9	1.7	1.7
L51	−0.1	0.0	0.0	0.0	0.1	0.2	−0.6	0.0	−0.4	−0.3	0.4	1.9
L52	0.1	0.2	0.2	0.2	0.1	0.2	1.0	1.1	1.2	1.1	0.5	1.9
L53	0.5	0.4	0.5	0.4	0.6	0.4	3.7	2.6	4.2	2.6	5.2	2.6
L54	−0.1	0.1	0.0	−0.1	−0.2	0.0	−0.3	0.5	0.2	−0.5	−1.3	0.2
L55	0.0	0.0	0.0	0.0	0.2	−0.1	0.1	0.0	0.1	0.1	1.2	0.0

Note: OVTT = out-of-vehicle (walking and waiting) time; IVTT = in-vehicle travel time.

6. CONCLUSIONS

In this study, we have proposed the use of designed quadrature (DQ) to approximate multidimensional integrals in maximum simulated likelihood estimation of discrete choice models. We have compared performance of DQ with workhorse QMC methods in a Monte Carlo and an empirical case study.

Whereas traditional sparse grid quadrature methods suffer from the problem of complex-valued loglikelihood due to negative weights, DQ could estimate MMNL smoothly for DGPs with varying covariance structures, owing to positivity of weights. The simulation study confirmed that DQ requires far fewer function evaluations than QMC if the variance–covariance matrix is diagonal. Even in DGPs with nondiagonal matrices and full covariance structures, DQ always outperforms MLHS in terms of model fit and parameter recovery when the quadrature rule is generated on higher order polynomial subspaces.

In sum, features such as positivity of weights, computational efficiency due to fewer function evaluations, and easy implementation due to reusability of quadrature rules make DQ an attractive alternative to QMC methods. In future work, we plan to test the performance of DQ in other discrete choice models (e.g., multinomial probit, and semiparametric logit models). Furthermore, to ensure better performance of DQ over QMC, the key question is: for a given dimension and number of draws, on what maximum order of polynomial subspaces, DQ rule can be generated? Taking advantage of the reusability feature of DQ, we plan to create software that can store the DQ rules on the highest possible order for commonly encountered dimensions, weight functions, and the number of nodes. With said software, DQ is as easy to use as any other QMC method, but with better performance. In other words, similar to QMC methods, the user would just need to choose the number of draws for the given dimension and software can provide the best DQ rule.

ACKNOWLEDGEMENTS

The authors are thankful to the National Science Foundation CAREER Award CBET-1253475, Agencia Nacional de Investigación y Desarrollo (ANID) FONDECYT 1191104, and ANID PIA/BASAL AFB180003 for financially supporting this research. They also thank Kenneth Train for sharing his initial thoughts on digital nets and Chandra Bhat for his comprehensive feedback. The comments from two anonymous reviewers and the editor have also helped in improving the manuscript substantially. The authors remain responsible for any remaining error/issues.

REFERENCES

Abay, K. A. (2015). Evaluating simulation-based approaches and multivariate quadrature on sparse grids in estimating multivariate binary probit models. *Economics Letters 126*, 51–6.

Askey, R. (1975). *Orthogonal Polynomials and Special Functions*. Regional Conference Series in Applied Mathematics, 21. Philadelphia, PA: SIAM.

Bhaduri, A. and L. Graham-Brady (2018). An efficient adaptive sparse grid collocation method through derivative estimation. *Probabilistic Engineering Mechanics 51*, 11–22.

Bhat, C. R. (2001). Quasi-random maximum simulated likelihood estimation of the mixed multinomial logit model. *Transportation Research Part B: Methodological 35*(7), 677–93.

Bhat, C. R. (2003). Simulation estimation of mixed discrete choice models using randomized and scrambled Halton sequences. *Transportation Research Part B: Methodological 37*(9), 837–55.

Bhat, C. R. (2011). The maximum approximate composite marginal likelihood (MACML) estimation of multinomial probit-based unordered response choice models. *Transportation Research Part B: Methodological 45*(7), 923–39.

Bhat, C. R. (2015). A new generalized heterogeneous data model (GHDM) to jointly model mixed types of dependent variables. *Transportation Research Part B: Methodological 79*, 50–77.

Brumm, J. and S. Scheidegger (2017). Using adaptive sparse grids to solve high-dimensional dynamic models. *Econometrica 85*(5), 1575–612.

Cagnone, S. and F. Bartolucci (2017). Adaptive quadrature for maximum likelihood estimation of a class of dynamic latent variable models. *Computational Economics 49*(4), 599–622.

Davis, P. J. and P. Rabinowitz (2007). *Methods of Numerical Integration*. Mineola, NY: Dover Publications.

Dick, J., R. N. Gantner, Q. T. Le Gia and C. Schwab (2017). Multilevel higher-order quasi-Monte Carlo Bayesian estimation. *Mathematical Models and Methods in Applied Sciences 27*(5), 953–95.

Dick, J. and F. Pillichshammer (2014). Discrepancy theory and quasi-Monte Carlo integration. In W. Chen, A. Srivastav and G. Travaglini (Eds.), *A Panorama of Discrepancy Theory*, 539–619. New York: Springer.

Gantner, R. N. (2016). A generic C++ library for multilevel quasi-Monte Carlo. In *Proceedings of the Platform for Advanced Scientific Computing Conference*, 1–12. New York: ACM.

Gantner, R. N., L. Herrmann and C. Schwab (2018). Quasi–Monte Carlo integration for affine-parametric, elliptic PDEs: Local supports and product weights. *SIAM Journal on Numerical Analysis 56*(1), 111–35.

Geweke, J., M. Keane and D. Runkle (1994). Alternative computational approaches to inference in the multinomial probit model. *Review of Economics and Statistics 76*, 609–32.

Goda, T. and J. Dick (2015). Construction of interlaced scrambled polynomial lattice rules of arbitrary high order. *Foundations of Computational Mathematics 15*(5), 1245–78.

Golub, G. H. and J. H. Welsch (1969). Calculation of Gauss quadrature rules. *Mathematics of Computation 23*(106), 221–30.

Goos, P. and K. Mylona (2018). Quadrature methods for Bayesian optimal design of experiments with nonnormal prior distributions. *Journal of Computational and Graphical Statistics 27*(1), 179–94.

Heiss, F. (2010). The panel probit model: Adaptive integration on sparse grids. In W. Greene and R. C. Hill (Eds.), *Maximum Simulated Likelihood Methods and Applications*, 41–64. Bingley, UK: Emerald Group.

Heiss, F. and V. Winschel (2008). Likelihood approximation by numerical integration on sparse grids. *Journal of Econometrics 144*(1), 62–80.

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: Methodological 40*(2), 147–63.

Jakeman, J. D. and A. Narayan (2018). Generation and application of multivariate polynomial quadrature rules. *Computer Methods in Applied Mechanics and Engineering 338*, 134–61.

Keshavarzzadeh, V., R. M. Kirby and A. Narayan (2018). Numerical integration in multiple dimensions with designed quadrature. *SIAM Journal on Scientific Computing 40*(4), A2033–A2061.

Keshavarzzadeh, V., R. M. Kirby and A. Narayan (2020). Generation of nested quadrature rules for generic weight functions via numerical optimization: Application to sparse grids. *Journal of Computational Physics 400*, 108979.

Keshavarzzadeh, V., R. M. Kirby and A. Narayan (2021). Multilevel designed quadrature for partial differential equations with random inputs. *SIAM Journal on Scientific Computing 43*, A1412–A1440.

Liu, Y., P. Bansal, R. Daziano and S. Samaranayake (2019). A framework to integrate mode choice in the design of mobility-on-demand systems. *Transportation Research Part C: Emerging Technologies 105*, 648–65.

Ma, X. and N. Zabaras (2009). An adaptive hierarchical sparse grid collocation algorithm for the solution of stochastic differential equations. *Journal of Computational Physics 228*(8), 3084–113.

Munger, D., P. L'Ecuyer, F. Bastin, C. Cirillo and B. Tuffin (2012). Estimation of the mixed logit likelihood function by randomized quasi-Monte Carlo. *Transportation Research Part B: Methodological 46*(2), 305–20.

Patil, P. N., S. K. Dubey, A. R. Pinjari, E. Cherchi, R. Daziano and C. R. Bhat (2017). Simulation evaluation of emerging estimation techniques for multinomial probit models. *Journal of Choice Modelling 23*, 9–20.

Richard, J.-F. and W. Zhang (2007). Efficient high-dimensional importance sampling. *Journal of Econometrics 141*(2), 1385–411.

Ryu, E. K. and S. P. Boyd (2015). Extensions of Gauss quadrature via linear programming. *Foundations of Computational Mathematics 15*(4), 953–71.

Sándor, Z. and K. Train (2004). Quasi-random simulation of discrete choice models. *Transportation Research Part B: Methodological 38*(4), 313–27.

Scheidegger, S. and A. Treccani (2018). Pricing American options under high-dimensional models with recursive adaptive sparse expectations. *Journal of Financial Econometrics 19*, 258–90.

Smolyak, S. A. (1963). Quadrature and interpolation formulas for tensor products of certain classes of functions. *Doklady Akademii Nauk 148*, 1042–45.

Train, K. E. (2009). *Discrete Choice Methods with Simulation*. Cambridge: Cambridge University Press.

Yu, J., P. Goos and M. Vandebroek (2010). Comparing different sampling schemes for approximating the integrals involved in the efficient design of stated choice experiments. *Transportation Research Part B: Methodological 44*(10), 1268–89.

SUPPORTING INFORMATION

Additional Supporting Information may be found in the online version of this article at the publisher's website:

Replication Package

Co-editor Victor Chernozhukov handled this manuscript.