SIAM/ASA J. Uncertainty Quantification
Vol. 6, No. 1, pp. 213–242

Cubature Formulas for Multisymmetric Functions and Applications to Stochastic Partial Differential Equations[*]

Clemens Heitzinger[†], Gudmund Pammer[†], and Stefan Rigger[‡]

Abstract. The numerical solution of stochastic partial differential equations and numerical Bayesian estimation is computationally demanding. If the coefficients in a stochastic partial differential equation exhibit symmetries, they can be exploited to reduce the computational effort. To do so, we show that permutation-invariant functions can be approximated by permutation-invariant polynomials in the space of continuous functions as well as in the space of p-integrable functions defined on $[0, 1]^s$ for $1 \leqslant p < \infty$. We proceed to develop a numerical strategy to compute cubature formulas that exploit permutation-invariance properties related to multisymmetry groups in order to reduce computational work. We show that in a certain sense there is no curse of dimensionality if we restrict ourselves to multisymmetric functions, and we provide error bounds for formulas of this type. Finally, we present numerical results, comparing the proposed formulas to other integration techniques that are frequently applied to high-dimensional problems such as quasi-Monte Carlo rules and sparse grids.

Key words. stochastic partial differential equation, quadrature and cubature formulas, multivariate integration, permutation-invariance, multisymmetric polynomials

AMS subject classifications. 13P25, 54C35, 58J70, 65D30, 65D32

DOI. 10.1137/17M1125418

1. Introduction.

When solving stochastic partial differential equations, the numerical approximation of the solutions is, in general, very computationally demanding because of the possibly large number of stochastic dimensions in addition to the spatial dimensions. If the coefficients of a stochastic partial differential equation exhibit an additional structure such as symmetries, the symmetries can be used to reduce the computational work significantly as shown here.

Numerical multivariate integration suffers from the so-called *curse of dimensionality*, meaning that for several classes of smooth functions the amount of function evaluations needed to achieve an error less than ϵ for all functions of a class (i.e., in the worst case) grows exponentially in dimension s (i.e., the number of arguments) [13, 14]. The efficient numerical treatment of high-dimensional problems can, however, be achieved by assuming a priori knowledge on the "importance" of the function arguments, for example, through the use of quasi-Monte Carlo rules adapted to function spaces endowed with weighted norms; see [9] for a survey. This a priori knowledge is often available if stochastic partial differential

[*]Received by the editors April 12, 2017; accepted for publication (in revised form) November 14, 2017; published electronically February 22, 2018.
http://www.siam.org/journals/juq/6-1/M112541.html
Funding: The authors acknowledge support by FWF (Austrian Science Fund) START project Y660 *PDE Models for Nanotechnology*.
[†]TU Wien (Clemens.Heitzinger@TUWien.ac.at, Gudmund.Pammer@TUWien.ac.at).
[‡]Corresponding author. TU Wien (a1028127@univie.ac.at).

equations are to be solved. Recently, the idea to exploit permutation-invariance conditions as another kind of a priori knowledge has been enunciated [25, 26], and it has been shown that the complexity of such integration problems can be significantly reduced if the permutation-invariance conditions are exploited in the construction of quasi-Monte Carlo rules [16]. In [17], a component-by-component construction scheme for a quasi-Monte Carlo method utilizing permutation-invariance properties has been proposed. This work also features a semi-constructive scheme for cubature formulas that is built on the idea of variance reduction and also makes use of permutation invariance.

We propose to develop interpolatory cubature formulas for permutation-invariance conditions related to *multisymmetry groups,* motivated by the following example.

Let u be a real-valued function of $s = nm$ variables defined on $\mathbb{R}^s$ that represents a physical attribute of a multiparticle system comprising n particles. Let the relevant parameters of the physical state of the n particles necessary to compute the variable u be the vector $(x_1, \ldots, x_s)$, where each particle is parametrized by an m-vector $\mathbf{x}_i = (x_{(i-1)m+1}, \ldots, x_{im}) \in \mathbb{R}^m$ for $i \in \{1, \ldots, n\}$. Assume that the particles are of the same type and are perfectly indistinguishable. Interchanging the role of two such particles in a representation of a physical state then results in a representation of the identical state. Translating this property to the function u means that $u(x_1, \ldots, x_s) = u(\mathbf{x}_1, \ldots, \mathbf{x}_n) = u(\mathbf{x}_{\pi(1)}, \ldots, \mathbf{x}_{\pi(n)}))$ holds for any transposition $\pi \in S_n$. Examples are $u(x_1, \ldots, x_s)$ being the gravitational or electrostatic force exerted on a particle located at $\mathbf{x} = 0$ by n point masses with identical weight that are distributed in m-dimensional space, where the coordinate vector of the ith particle is given by $\mathbf{x}_i$. Clearly, interchanging the coordinate vectors of two such particles does not change the value of u. This motivates the following definition.

Definition 1. *Let m and n be natural numbers and $s := mn$. Let $a, b, \ldots, z$ be an alphabet of n letters. We organize the arguments of a function $u\colon [0,1]^s \to \mathbb{R}$ in a matrix*

$$X := \begin{pmatrix} a_1 & \ldots & z_1 \\ a_2 & \ldots & z_2 \\ \vdots & \ddots & \vdots \\ a_m & \ldots & z_m \end{pmatrix}$$

*of indeterminates $a_1, \ldots, z_m \in [0,1]$. The function u is called (n,m)-*multisymmetric *if u is unchanged under permutations of the columns of X. We denote the corresponding group of permutations of $\mathbb{R}^s$ by $S_{n,m}$.*

This property is sometimes called MacMahon symmetry or vector symmetry. It is easy to see that for any s-variate real-valued function u, the set $\{\sigma \in S_s \mid u \circ \sigma = u\}$ is a subgroup of S_s, so the following notion is natural.

Definition 2. *Let $(G, \circ)$ be a subgroup of $(S_s, \circ)$. A function $u\colon [0,1]^s \to \mathbb{R}$ is called* G-invariant *if*

$$u(\sigma(x_1, \ldots, x_s)) = u(x_1, \ldots, x_s) \quad \forall \sigma \in G \quad \forall (x_1, \ldots, x_s) \in [0,1]^s. \tag{1}$$

The ultimate goal of this paper is to develop interpolatory cubature formulas for multisymmetric functions and hence stochastic partial differential equations with multisymmetric coefficients, aiming to save function or sample evaluations by exploiting the property of permutation invariance. The theoretical results are proven for the more general case of G-invariant functions, while numerical results are presented for the case of multisymmetric functions.

Remark 3. The formulas developed here are different from what are called *G-invariant/symmetric cubature formulas* in the literature. G-invariant cubature formulas are intended to work for *any* kind of smooth function, and the nodes and weights of the formula are supposed to satisfy G-invariance. In contrast, our formulas *exclusively* work for G-invariant functions, while the cubature formulas do not necessarily satisfy G-invariance properties.

We would also like to mention that the notion of *multisymmetric functions* in the algebraic sense refers to elements of an abstract, category-theoretical construction that is not canonically related to real-valued functions. In this paper, whenever we refer to multisymmetric functions, we consider real-valued functions defined on $[0,1]^s$ that satisfy permutation-invariance properties related to a multisymmetry group as defined earlier.

The idea to exploit permutation-invariance properties to facilitate multivariate high-dimensional numerical integration is still young, and the work performed so far [25, 26, 16, 17] has been focused on a mathematical setting tailored to the Schrödinger equation, and as a consequence employs different notions of permutation invariance. Our definition of G-invariance includes the symmetry properties introduced in [25, 26], but not the respective antisymmetry properties. It seems that quasi-Monte Carlo methods or variants thereof are usually employed for high-dimensional problems. We will show in this paper that in the case of multisymmetric integrands, interpolatory formulas (besides sparse grids) can be a reasonable alternative.

The rest of this paper is organized as follows: In section 2, we show that G-invariant functions can be approximated by G-invariant polynomials in the space of continuous functions as well as in L^p-spaces for $1 \leqslant p < \infty$. In section 3, we provide error bounds for cubature formulas for multisymmetric functions with positive weights. In section 4, we outline a possible approach to computing cubature formulas for G-invariant functions. In section 5, we introduce results about bases of spaces of multisymmetric polynomials and slightly strengthen and generalize them. In the second part of this section, we explain how to execute the algorithm discussed in section 4 more explicitly and how the special structure of multisymmetry groups proves to be advantageous. Finally, we present numerical results in section 6, comparing the proposed formulas to quasi-Monte Carlo rules and adaptive sparse-grid schemes. The conclusions follow in section 7.

2. Approximation of G-invariant functions by G-invariant polynomials.

From now on, G denotes a subgroup of S_s. We denote the s-dimensional unit cube by $I := [0,1]^s$. If X is a vector space of real-valued functions defined on a subset of $\mathbb{R}^s$, we denote its subspace of G-invariant functions by X^G, where G-invariance is to be understood in an almost-everywhere sense if $X = L^p$. We denote the space of polynomial functions defined on $[0,1]^s$ by $\mathcal{P}$, and the space of polynomial functions of degree less than or equal to d by $\mathcal{P}_{\leqslant d}$. Symmetrization operators like the one considered in Proposition 4 are sometimes considered in invariant theory, e.g., in the proof of the Hilbert basis theorem, and are also called *Reynolds* operators in the literature. Analytical properties of such operators have been derived in [25, 26, 16], although not for the precise setting we are interested in.

Proposition 4. *Let $X := (C(I), \|.\|_\infty)$ or $X := (L^p(I), \|.\|_p)$ for $1 \leqslant p \leqslant \infty$. Then the linear averaging operator*

$$\begin{aligned} S\colon X &\to X, \\ f &\mapsto \frac{1}{|G|} \sum_{\sigma \in G} f \circ \sigma \end{aligned}$$

satisfies

$$S(1) = 1, \tag{3a}$$

$$S(S(f)g) = S(f) \cdot S(g) \quad \forall f, g \in X. \tag{3b}$$

Furthermore, S is a continuous linear projection with range $R(S) = X^G$.

Proof. The well-definedness of S in the case $X = L^p(I)$ follows from the fact that for any measurable subset $M \subset \mathbb{R}^N$ and any $\sigma \in S_s$ we have $\lambda(M) = \lambda(\sigma^{-1}(M))$. We now prove formula (3). Clearly, $S(1) = 1$. Let $f, g \in X$. For any $\pi \in G$, we have $G \circ \pi = G$ and, therefore,

$$S(S(f)g) = S\left(g \cdot \frac{1}{|G|} \sum_{\sigma \in G} f \circ \sigma\right) = \frac{1}{|G|} \sum_{\pi \in G} \left(g \circ \pi \cdot \frac{1}{|G|} \sum_{\sigma \in G} f \circ \sigma \circ \pi\right) = S(f) \cdot S(g)$$

holds.

Setting $g := 1$ in (3) shows $S^2 = S$. For $\sigma \in S_N$, the application $f \mapsto f \circ \sigma$ is an isometry on X. By the triangle inequality, S is bounded with $\|S\| = 1$ and therefore a continuous linear projection. Note that $S(f) \in X^G$ for $f \in X$ and $S(g) = g$ for any $g \in X^G$, so we must have that $R(S) = X^G$. ■

Theorem 5 (G-invariant continuous functions). *The set of continuous, G-invariant functions $(C(I)^G, \|.\|_\infty)$ is a Banach algebra. The G-invariant polynomials $\mathcal{P}(I)^G$ form a dense subalgebra of $(C(I)^G, \|.\|_\infty)$.*

Proof. Consider the operator S defined in Theorem 4. As $R(S) = C(I)^G$ and S is a continuous projection, we see that $C(I)^G$ is a closed subalgebra of $C(I)$. For the second part of the theorem, let $f \in C(I)^G$. Due to the classical Weierstrass approximation theorem, there is a sequence of polynomials p_n converging uniformly to f. As S is continuous, we have $S(p_n) \to S(f)$ as $n \to \infty$, and since $S(f) = f$ this proves the claim. ■

Theorem 6 (L^p-spaces of G-invariant functions.). *For $1 \leqslant p < \infty$, the space $L^p(I)^G$ is a Banach space. $\mathcal{P}(I)^G$ is a dense subspace of $L^p(I)^G$.*

Proof. The proof is analogous to the proof of Theorem 5, making use of the fact that $\mathcal{P}(I)$ is dense in $L^p(I)$. ■

Theorem 7. *The Taylor polynomials of a (sufficiently smooth) G-invariant function $f : [0,1]^s \to \mathbb{R}$ centered at a G-invariant point $\mathbf{a} \in I$ (i.e., $\sigma(\mathbf{a}) = \mathbf{a}$ for all $\sigma \in G$) are G-invariant.*

Proof. The Taylor polynomial of order k centered at $\mathbf{a} = (a_1, \ldots, a_s)$ has the form

$$T_k(x_1, \ldots, x_s) = \sum_{\substack{|\alpha| \leqslant k \\ \alpha \in \mathbb{N}_0^s}} \frac{1}{\alpha!} D^\alpha f(\mathbf{a}) \prod_{i=1}^{s} (x_i - a_i)^{\alpha_i},$$

whereby $\alpha! := \prod_{i=1}^s \alpha_i!$. For $\sigma \in G$, note that $\alpha! = \sigma(\alpha)!$, $a_i = a_{\sigma(i)}$, and

$$\prod_{i=1}^{s} (x_i - a_i)^{\alpha_i} = \prod_{i=1}^{s} (x_{\sigma(i)} - a_{\sigma(i)})^{\alpha_{\sigma(i)}}.$$

Thus,

$$\begin{aligned} T_k(\sigma(x_1, \ldots, x_s)) &= \sum_{\substack{|\alpha| \leqslant k \\ \alpha \in \mathbb{N}_0^s}} \frac{1}{\alpha!} D^\alpha f(\mathbf{a}) \prod_{i=1}^{s} (x_{\sigma(i)} - a_i)^{\alpha_i} \\ &= \sum_{\substack{|\alpha| \leqslant k \\ \alpha \in \mathbb{N}_0^s}} \frac{1}{\sigma(\alpha)!} D^{\sigma(\alpha)} f(\mathbf{a}) \prod_{i=1}^{s} (x_{\sigma(i)} - a_{\sigma(i)})^{\alpha_{\sigma(i)}} \\ &= \sum_{\substack{|\alpha| \leqslant k \\ \alpha \in \mathbb{N}_0^s}} \frac{1}{\alpha!} D^{\sigma(\alpha)} f(\mathbf{a}) \prod_{i=1}^{s} (x_i - a_i)^{\alpha_i}. \end{aligned}$$

In order to show that $D^\alpha f(\mathbf{a}) = D^{\sigma(\alpha)} f(\mathbf{a})$, we show the claim

$$D^\alpha f(\mathbf{x}) = D^{\sigma(\alpha)} f(\sigma(\mathbf{x})) \quad \forall \mathbf{x} \in (0,1)^s$$

inductively on the order of the multi-index α: For $\alpha = 0$, the claim is trivial by the G-invariance of f. Assume that $D^\alpha f(\mathbf{x}) = D^{\sigma(\alpha)} f(\sigma(\mathbf{x}))$ and let $\mathbf{e}_i \in \mathbb{R}^s$ such that $\mathbf{e}_i(j) = \delta_{ij}$. This yields

$$\begin{aligned} D^{e_i + \alpha} f(\mathbf{x}) &= \lim_{h \to 0} \frac{D^\alpha f(\mathbf{x} + h\mathbf{e}_i) - D^\alpha f(\mathbf{x})}{h} \\ &= \lim_{h \to 0} \frac{D^{\sigma(\alpha)} f(\sigma(\mathbf{x} + h\mathbf{e}_i)) - D^{\sigma(\alpha)} f(\sigma(\mathbf{x}))}{h} = D^{\sigma(e_i + \alpha)} f(\sigma(\mathbf{x})), \end{aligned}$$

which concludes the proof. ■

Remark 8. Without the assumption that the center of the Taylor polynomials be G-invariant, Theorem 7 does not hold in general: Consider, for example, the first-order Taylor polynomial of $f(x, y) := xy$ centered at $\mathbf{a} = (1, 0)$.

3. Error bounds.

Definition 9 (cubature formula and error functional). *An N-point cubature formula Q is a linear functional on $C(I)$ of the form*

$$Q(f) = \sum_{i=1}^{k} \omega_i f(\mathbf{x}_i),$$

where $\omega_i \in \mathbb{R}$ and $\mathbf{x}_i \in I$ for all $i \in \{1, \dots, k\}$. The associated error functional *E is defined by*

$$E(f) = \int_I f(\mathbf{x})d\mathbf{x} - Q(f).$$

We say that a cubature formula is of degree d if $E(p) = 0$ for all $p \in \mathcal{P}_{\leqslant d}$. Cubature formulas satisfying $E(p) = 0$ for all $p \in \mathcal{P}_{\leqslant d}$ for a $d \in \mathbb{N}$ are also called *interpolatory* or *monomial*. Many different cubature rules for elementary regions such as the s-cube or s-sphere have been developed in the past; see [23, 5] for compilations. In the monograph [6, p. 376], it is stated that "Two properties of approximate integration rules are considered particularly desirable: The abscissas should lie in the region and the weights should be positive." The cubature formulas we propose have nodes inside the unit cube and positive weights. In this section, we will prove error bounds that apply to any kind of monomial rule for G-invariant functions with positive weights and nodes inside the unit cube, although our error bounds are not fully explicit.

Estimates for the optimal approximation of real-valued, smooth functions by polynomials are known as Jackson theorems. In order to prove a priori estimates for multivariate cubature formulas, we quote the following relatively recent Jackson-type theorem.

Theorem 10. *Let K be a connected compact subset of $\mathbb{R}^s$ such that any two points a and b of K can be joined by a rectifiable arc in K with length no greater than $\sigma|a - b|$, where σ is a positive constant. Let f be a function of class C^m on an open neighborhood of K where $0 \leqslant m < \infty$. Then for each nonnegative integer n, there is a polynomial p_n of degree at most n on $\mathbb{R}^s$ with the following property: For each multi-index α with $|\alpha| \leqslant \min(m, n)$, we have*

$$\|D^\alpha(f - p_n)\|_\infty \leqslant \frac{C}{n^{m-|\alpha|}} \sum_{|\gamma| \leqslant m} \|D^\gamma f\|_\infty, \tag{4}$$

where C is a positive constant depending only on s, m, and K and $\|.\|_\infty$ denotes the supremum norm on K.

Proof. See [1, Theorem 2] for the proof. ■

The following error bounds are a simple consequence of the properties of the symmetrization operator S and the above Jackson theorem.

Theorem 11 (error bounds). *Let Q be an N-point cubature formula with positive weights ω_i and error functional E that integrates every G-invariant polynomial of degree at most n exactly, i.e., $\omega_i \geqslant 0$ for all $i \in \{1, \dots, k\}$ and $E(p) = 0$ for all $p \in \mathcal{P}^G_{\leqslant n}$. It follows that there is a constant $K > 0$ depending only on s and m such that*

$$|E(f)| \leqslant \frac{K(s, m)}{n^m} \sum_{|\gamma| \leqslant m} \|D^\gamma f\|_\infty$$

holds for all G-invariant functions $f \colon I \to \mathbb{R}$ of class C^m.

Proof. Let $f \in C(I)^G$ and Q be as stated in the assumptions. Consider again the linear operator S introduced in Proposition 4. Let p_n denote a polynomial approximation of f

satisfying the error bound in Theorem 10. Setting $s_n := S(p_n)$, we obtain

$$\|s_n - f\|_\infty = \|S(p_n - f)\|_\infty \leqslant \|S\| \cdot \|p_n - f\|_\infty \leqslant \frac{C}{n^m} \sum_{|\gamma| \leqslant m} \|D^\gamma f\|_\infty. \tag{5}$$

Note that s_n is G-invariant and of degree at most n, thus $E(s_n) = 0$. Moreover, as Q integrates constants exactly, we have $\sum_{i=1}^k \omega_i = 1$. This yields

$$\begin{aligned} |E(f)| = |E(f - s_n)| &\leqslant \int_I |f(\mathbf{x}) - s_n(\mathbf{x})| d\mathbf{x} + \sum_{i=1}^k \omega_i |f(\mathbf{x}_i) - s_n(\mathbf{x}_i)| \\ &\leqslant 2\|s_n - f\|_\infty \leqslant \frac{2C}{n^m} \sum_{|\gamma| \leqslant m} \|D^\gamma f\|_\infty, \end{aligned}$$

which concludes the proof. ■

Remark 12. Setting $G := \{\mathrm{id}\}$ in Theorem 10, one obtains an error bound for the classical case. Thus, the error bounds formulated above also apply to the formulas constructed in [7]. The same reasoning given in equation (5) shows that for $X = L^p(I)$ or $X = C(I)$ and $f \in X^G$, one has

$$\operatorname{dist}(\mathcal{P}_{\leqslant d}, f) = \operatorname{dist}\left(\mathcal{P}^G_{\leqslant d}, f\right). \tag{6}$$

4. Computing cubature formulas for G-invariant functions. In this section, an approach for computing cubature formulas with positive weights for G-invariant functions on $[0,1]^s$ is proposed. For the remainder of this section d is assumed to be odd (this is due to the fact that an N-point univariate Gaussian quadrature rule has degree of exactness of $2N-1$). Let $B := \{p_1, \ldots, p_{n^*}\}$ be a basis of $\mathcal{P}^G_{\leqslant d}$ and define $n^* := \dim \mathcal{P}^G_{\leqslant d}$. Let $\mathcal{N} := \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ be a collection of points in $[0,1]^s$. A cubature formula based on the nodes $\mathcal{N}$ integrates every G-invariant polynomial of degree at most d exactly if and only if the weights $\omega_1, \ldots, \omega_N$ satisfy the system

$$\sum_{j=1}^N p_i(\mathbf{x}_j)\omega_j = \int_{[0,1]^s} p_i(\mathbf{x}) d\mathbf{x} \quad \forall i \in \{1, \ldots, n^*\}. \tag{7}$$

In the univariate case, system (7) has a unique solution for any choice of distinct $\mathbf{x}_1, \ldots, \mathbf{x}_d$. In the multivariate case, system (7) is not always solvable and if a solution exists, it is not unique in general.

Definition 13. *The natural action of G on a G-invariant set $C \subseteq [0,1]^s$ is the group action given by*

$$\begin{aligned} \phi \colon G \times C &\to C, \\ (\sigma, \mathbf{x}) &\mapsto \sigma(\mathbf{x}). \end{aligned}$$

4.1. Basic scheme. We suggest the following algorithm, which is a variation of the approach presented in [7].

1. Generate a basis $p_1, \ldots, p_{n^*}$ of $\mathcal{P}^G_{\leqslant d}$.
2. Calculate the integrals $(\mathbf{b})_i := \int_I p_i(\mathbf{x}) d\mathbf{x}$ for all $i \in \{1, \ldots, n^*\}$.
3. Calculate the nodes of a univariate degree d Gaussian quadrature formula on $[0,1]$, denoted by $\mathcal{N}_0$. Let $C := \times_{i=1}^{s} \mathcal{N}_0$. Consider the natural action of G on C. Choose elements $\mathbf{x}_1, \ldots, \mathbf{x}_k$ such that the disjoint union of the orbits of the $\mathbf{x}_i$ equals all of C.
4. Define $A \in \mathbb{R}^{n^* \times k}$ by $(A)_{ij} := (p_i(\mathbf{x}_j))$. Solve the linear programming problem (LPP) with a trivial objective function

$$\begin{array}{ll} \text{minimize} & 0 \cdot \omega \\ \text{subject to} & A\omega = \mathbf{b} \\ \text{and} & \omega \geqslant \mathbf{0}. \end{array}$$

This algorithm is guaranteed to terminate with a cubature formula with positive weights, of whom at most n^* are strictly positive. First off, notice that the LPP solved in the fourth step is always feasible. To see this, note that the s-dimensional tensor product of a univariate degree-d Gaussian quadrature rule solves system (7) with $\mathcal{N} = C$. Let $\mathbf{y}_j$ denote the nodes and w_j denote the weights of the tensor product formula for $j \in \{1, \ldots, (\frac{d+1}{2})^s\}$. If $\mathbf{y} \in O_{\mathbf{x}}$, where $O_{\mathbf{x}}$ denotes the orbit of $\mathbf{x}$, there is a $\sigma \in G$ such that $\mathbf{y} = \sigma(\mathbf{x})$, and as the p_i are G-invariant, we have $p(\mathbf{x}) = p(\mathbf{y})$. Let $\mathbf{x}_1, \ldots, \mathbf{x}_k$ be defined as in the third step and set $J_i := \{j : \mathbf{y}_j \in O_{\mathbf{x}_i}\}$ for $i \in \{1, \ldots, k\}$. Then, the vector ω defined by

$$\omega_i := \sum_{j \in J_i} w_j$$

solves the system $A\omega = \mathbf{b}$ and we have $\omega \geqslant 0$. Therefore the LPP defined in step 4 is feasible. Clearly, the feasible region is bounded as $\omega \geqslant 0$ and the zeroth-order equation gives $\sum_{i=1}^{k} \omega_k = 1$. Therefore, if the LPP is solved using an algorithm that produces extreme point solutions (like the simplex method), the solution will have at most n^* strictly positive weights, as extreme point solutions correspond to basic feasible solutions for bounded LPPs.

We will proceed to present constructive algorithmic solutions to steps 1–3 of the fundamental algorithm. Step 4 can then be carried out using any LPP solver that produces extreme point solutions. Although our algorithms are theoretically viable for any subgroup G of S_s, they are not computationally efficient. We will present an optimized version of the basic scheme for the multisymmetric case in section 5.

4.2. Generating a basis of $\mathcal{P}^G_{\leqslant d}$. Let $\mathcal{I}$ be the set of all multi-indices α satisfying $|\alpha| \leqslant d$, i.e., $\mathcal{I} = \{(\alpha_1, \ldots, \alpha_s) \mid \alpha_i \in \mathbb{N}_0,\ \sum_{i=1}^{s} \alpha_i \leqslant d\}$. Consider the natural action of G on $\mathcal{I}$. Choose $\beta_1, \ldots, \beta_{n^*}$ such that the disjoint union of the orbits of the β_i is all of $\mathcal{I}$. The fact that $\{S(\mathbf{x}^{\beta_i}) \mid i \in \{1, \ldots, n^*\}\}$ is a basis of $\mathcal{P}^G_{\leqslant d}$ seems to be regarded as a basic fact and is often mentioned without proof or reference. For the sake of completeness, we provide an elementary proof.

Lemma 14. *The family $(S(\mathbf{x}^{\beta_i}))_{i=1,\ldots,n^*}$ is a basis of $\mathcal{P}^G_{\leqslant d}$, where S is the operator introduced in Proposition 4.*

Proof. In order to show linear independence of the $S(\mathbf{x}^{\beta_i})$, note that

$$D^{\beta}(\mathbf{x}^{\alpha}) = \alpha!\delta_{\alpha\beta} \quad \forall \beta \in \mathbb{N}_0^s \text{ with } |\beta| = |\alpha| \tag{8}$$

holds for all $\alpha \in \mathcal{I}$. Let $c_1, \dots, c_{n^*}$ be real numbers such that

$$\sum_{i=1}^{n^*} c_i S(\mathbf{x}^{\beta_i}) = \frac{1}{|G|} \sum_{i=1}^{n^*} c_i \sum_{\sigma \in G} \mathbf{x}^{\sigma(\beta_i)} = 0. \tag{9}$$

Let β_j be a multi-index of order d, i.e., $|\beta_j| = d$. Because $O_{\beta_i} \cap O_{\beta_j} = \varnothing$ for $i \neq j$, we have $\beta_j \neq \sigma(\beta_i)$ for all $\sigma \in G$ and $i \neq j$. Setting $m := |\{\sigma \in G \mid \sigma(\beta_j) = \beta_j\}|$, we obtain

$$D^{\beta_j}\left(\sum_{i=1}^{n^*} c_i \sum_{\sigma \in G} \mathbf{x}^{\sigma(\beta_i)}\right) = c_j m \beta_j! = 0. \tag{10}$$

Repeating this reasoning inductively yields $c_1 = \dots = c_{n^*} = 0$ and thus linear independence.

We proceed to prove that the vectors $S(\mathbf{x}^{\beta_i})$ generate $\mathcal{P}^G_{\leqslant d}$. As $S \colon \mathcal{P}_{\leqslant d} \to \mathcal{P}^G_{\leqslant d}$ is surjective, the family $(S(\mathbf{x}^{\alpha}))_{\alpha \in \mathcal{I}}$ generates $\mathcal{P}^G_{\leqslant d}$. If $\alpha \in O_{\beta}$, we have $S(\mathbf{x}^{\alpha}) = S(\mathbf{x}^{\beta})$. Since $\beta_1, \dots, \beta_{n^*}$ are chosen in a way such that $\bigcup_{i=1}^{n^*} O_{\beta_i} = \mathcal{I}$, we obtain

$$\operatorname{span}\{S(\mathbf{x}^{\alpha}) \mid \alpha \in \mathcal{I}\} = \operatorname{span}\left\{S\left(\mathbf{x}^{\beta_i}\right) \mid i \in \{1, \dots, n^*\}\right\}, \tag{11}$$

which proves the claim. ∎

Using Lemma 14, a basis of $\mathcal{P}^G_{\leqslant d}$ can be generated as follows.

Algorithm 4.1 Basis Generation.

```
Set $\mathcal{I} := \{(\alpha_1, \dots, \alpha_s) \mid \alpha_i \in \mathbb{N}_0, \sum_{i=1}^s \alpha_i \leqslant d\}$
Set $B := \varnothing$
while $\mathcal{I} \neq \varnothing$ do
  Pick $\alpha \in \mathcal{I}$
  Update $B := B \cup \{S(\mathbf{x}^{\alpha})\}$
  Update $\mathcal{I} := \mathcal{I} \backslash \{\sigma(\alpha) : \sigma \in G\}$
end while
return $B$
```

4.3. Calculating integrals of basis polynomials. Having generated a basis of the form $S(\mathbf{x}^{\beta_1}), \dots, S(\mathbf{x}^{\beta_{n^*}})$ as outlined in section 4.2, calculating the integrals is straightforward because of the equality

$$\int_{[0,1]^s} S(\mathbf{x}^{\beta_i}) d\mathbf{x} = \int_{[0,1]^s} \mathbf{x}^{\beta_i} d\mathbf{x}. \tag{12}$$

Algorithm 4.2 Node Generation.

```
Set $C := \bigtimes_{i=1}^{s} \mathcal{N}_0$
Set $\mathcal{N} := \varnothing$
while $C \neq \varnothing$ do
    Pick $\mathbf{x} \in C$
    Update $\mathcal{N} := \mathcal{N} \cup \{\mathbf{x}\}$
    Update $C := C \backslash \{\sigma(\mathbf{x}) \mid \sigma \in G\}$
end while
return $\mathcal{N}$
```

4.4. Generating nodes modulo G. Calculating univariate Gaussian quadrature formulas is a prominent problem, and there are many mathematical software libraries that are able to efficiently compute formulas of this type. Significant advances have been made recently in [2]. Let $\mathcal{N}_0$ denote the nodes of a univariate degree-d Gaussian quadrature formula. A naive way to obtain a full representative system of $C := \bigtimes_{i=1}^{s} \mathcal{N}_0$ modulo G is given in Algorithm 4.2.

Steps 1 and 3 of the basic scheme require iterations over the group G. For large $|G|$, the computational complexity of these steps will therefore be high and the proposed algorithms will be impractical. We suggest an algorithm that scales well with dimension for the case of multisymmetry groups in the next section.

5. The multisymmetric case.

5.1. Finding a suitable basis—some algebraic results. From now on, let n and m be positive integers, and $s := nm$. As mentioned earlier, the generally applicable algorithms proposed in section 4.2 are highly inefficient with respect to dimensional scaling. The study of multisymmetric polynomials is an old and developed one, going back as far as 1852 [21]. A modern and extensive introduction to the topic can be found in [4]. (Minimal) generating sets for spaces of multisymmetric polynomials have been exposed in [24, 20]. We begin by exhibiting a basis of $\mathcal{P}_{\leqslant d}^{S_{m,n}}$.

Remark 15. We are only interested in $P_{\leqslant d}^{S_{m,n}}$ as an $\mathbb{R}$-vector space; therefore, we will not strictly distinguish between polynomials in the algebraical sense and polynomial functions defined on $\mathbb{R}^s$.

Definition 16. *A* vector partition *is a finite multiset of elements of* $\mathbb{N}_0^m \backslash \{\mathbf{0}\}$.

Let $\boldsymbol{\alpha} = (\alpha^{(1)}, \ldots, \alpha^{(k)})$ be a vector of k elements of $\mathbb{N}_0^m$. We denote the vector partition that is obtained by dropping the $\mathbf{0}$-terms and the order of the terms in $(\alpha^{(1)}, \ldots, \alpha^{(k)})$ by $[\boldsymbol{\alpha}] := [(\alpha^{(1)}, \ldots, \alpha^{(k)})]$. Let $\mathfrak{p}$ be a vector partition. We can find a vector of nonzero vectors $\boldsymbol{\alpha} = (\alpha^{(1)}, \ldots, \alpha^{(k)})$ such that $\mathfrak{p} = [\boldsymbol{\alpha}]$. We will refer to the terms $\alpha^{(i)}$ in $\boldsymbol{\alpha}$ as the *parts* of $\mathfrak{p}$ and to the integer k as the *length* of the vector partition and will denote it by $\ell_{\mathfrak{p}}$. We call $\alpha^{(1)} + \cdots + \alpha^{(k)}$ the *sum* of $\mathfrak{p}$ and will denote it by $s(\mathfrak{p})$. If we have $s(\mathfrak{p}) = \gamma$ for $\gamma \in \mathbb{N}_0^m \backslash \{\mathbf{0}\}$, we call $\mathfrak{p}$ a *partition* of γ. Let Π_m denote the set of vector partitions with parts in $\mathbb{N}_0^m \backslash \{\mathbf{0}\}$. For an integer vector γ, let $|\gamma|$ denote the sum of its components.

Example 17. The four vector partitions of $(2,1)$ are given by

$$\begin{gathered}[(1,0),(1,0),(0,1)],\\ [(2,0),(0,1)],\\ [(1,1),(1,0)],\\ [(2,1)].\end{gathered}$$

Let $a, b, \dots, z$ denote an alphabet of n letters as in the introduction. As was shown in section 4.2, we can find a basis of $\mathcal{P}_{\leqslant d}^{S_{m,n}}$ by taking the symmetrizations of the standard monomial basis. Using the notions we just introduced, we are now able to write the basis obtained in this way in a more explicit fashion.

Definition 18. *Let $\mathfrak{p}$ be a vector partition with parts in $\mathbb{N}_0^m$ of length at most n. The* monomial multisymmetric function with index $\mathfrak{p}$ *is defined as*

$$m_{\mathfrak{p}} = \sum_{(\alpha^{(a)},\dots,\alpha^{(z)})\in\mathfrak{I}(\mathfrak{p})} \boldsymbol{a}^{\alpha^{(a)}} \boldsymbol{b}^{\alpha^{(b)}} \cdots \boldsymbol{z}^{\alpha^{(z)}},$$

where $\mathfrak{I}(\mathfrak{p})$ is the set of all $\boldsymbol{\alpha} = (\alpha^{(a)}, \dots, \alpha^{(z)})$ such that $[\boldsymbol{\alpha}] = \mathfrak{p}$.

Example 19. Let $m := 2$, $n := 3$, and $\mathfrak{p} := [(1,0),(1,0),(1,1)]$. Then we have

$$\begin{aligned} m_{[(1,0),(1,0),(1,1)]} &= \boldsymbol{a}^{(1,0)}\boldsymbol{b}^{(1,0)}\boldsymbol{c}^{(1,1)} + \boldsymbol{a}^{(1,0)}\boldsymbol{b}^{(1,1)}\boldsymbol{c}^{(1,0)} + \boldsymbol{a}^{(1,1)}\boldsymbol{b}^{(1,0)}\boldsymbol{c}^{(1,0)}\\ &= a_1b_1c_1c_2 + a_1b_1b_2c_1 + a_1a_2b_1c_1.\end{aligned}$$

Theorem 20. *The monomial multisymmetric functions $m_{\mathfrak{p}}$, where $\mathfrak{p}$ is a vector partition with parts in $\mathbb{N}_0^m$ of length at most n, together with the constant function 1 form a basis of $\mathcal{P}^{S_{m,n}}$.*

Similarly to Lemma 14, Theorem 20 appears to be seen as a basic fact and is often mentionend without proof or reference. Again, we include a proof for the sake of completeness.

Proof. We may write any monomial in the form $\boldsymbol{x}^{\boldsymbol{\alpha}} = \boldsymbol{a}^{\alpha^{(a)}}\boldsymbol{b}^{\alpha^{(b)}}\cdots\boldsymbol{z}^{\alpha^{(z)}}$. We associate the vector partition $[\boldsymbol{\alpha}] = [\alpha^{(a)}, \alpha^{(b)}, \dots, \alpha^{(z)}]$ with the multiindex $\boldsymbol{\alpha} \in \mathbb{N}_0^s$. Letting $S_{m,n}$ act naturally on $\mathbb{N}_0^s$, we see that the orbit of an element $\boldsymbol{\alpha} \in \mathbb{N}_0^s$ can be described by $\mathfrak{I}([\boldsymbol{\alpha}])$, i.e., all the sequences $\boldsymbol{\beta} = (\beta^{(a)}, \dots, \beta^{(z)})$ such that $[\boldsymbol{\beta}] = [\boldsymbol{\alpha}]$. In particular, $S(\mathbf{x}^{\boldsymbol{\alpha}})$ agrees with $m_{[\boldsymbol{\alpha}]}$ up to a nonzero factor; here, S is the symmetrization operator introduced in Proposition 4. As $\mathfrak{p}$ runs through the vector partitions of length at most n with parts in $\mathbb{N}_0^m\backslash\{\mathbf{0}\}$, $\mathfrak{I}(\mathfrak{p})$ runs through the orbits of $\mathbb{N}_0^s\backslash\{\mathbf{0}\}$ under $S_{m,n}$. Thus, the claim follows from Lemma 14. ■

As a corollary of Theorem 20, we find that

$$\dim \mathcal{P}(I)_{\leqslant d}^{S_{m,n}} = 1 + |\{\mathfrak{p} \in \Pi_m \mid |s(\mathfrak{p})| \leqslant d, \ell_{\mathfrak{p}} \leqslant n\}|. \tag{13}$$

Formula (13) exhibits a property that will prove to be advantageous for our endeavor: If $n \geqslant d$, the condition $\ell_{\mathfrak{p}} \leqslant n$ is implied by $|s(\mathfrak{p})| \leqslant d$. Therefore, we see that the dimension of $\mathcal{P}_{\leqslant d}^{S_{m,n}}$ is constant in n for $n \geqslant d$. Recall that $\dim \mathcal{P}(I)_{\leqslant d}^{S_{m,n}}$ gives an upper bound on the amount of weights needed to integrate all polynomials of $\mathcal{P}(I)_{\leqslant d}^{S_{m,n}}$ exactly using our method, therefore, this quantity can be bounded independently of n. In other words, there is no *curse of dimensionality* on the amount of nodes needed to integrate all multisymmetric polynomials of a given maximal degree exactly.

For the practical calculation of the cubature formulas, we prefer working with *elementary multisymmetric functions* instead of monomial multisymmetric functions. They are defined as follows. (We choose a definition similar to the one in [24].)

Definition 21. *Let $P \in \mathbb{R}[X_1, \dots, X_m]$ be of positive degree. We define the* elementary multisymmetric functions associated with P *over the following generating function:*

$$\sum_{k=0}^{n} t^k e_k(P) := (1 + tP(\boldsymbol{a})) \cdot (1 + tP(\boldsymbol{b})) \cdots (1 + tP(\boldsymbol{z})). \tag{14}$$

It follows easily from this definition that

$$e_1(P) = P(\boldsymbol{a}) + P(\boldsymbol{b}) + \dots + P(\boldsymbol{z}) \tag{15}$$

holds.

Polynomials of the type $e_1(\boldsymbol{x}^\alpha)$ are usually referred to as *power sum multisymmetric monomials* and denoted by p_α.

Example 22. Let $n := 3, m := 2$, and $\mu := X_1^2 X_2$. Then we have

$$e_1(\mu) = a_1^2 a_2 + b_1^2 b_2 + c_1^2 c_2.$$

We will only need functions $e_1(P)$ with $P \in \mathbb{R}[X_1, \dots, X_m]^+$, where the $+$ means that we only take polynomials of positive degree. Similarly, let $\mathcal{M}_m^+$ denote the set of monomials in $\mathbb{R}[X_1, \dots, X_m]$ of positive degree.

Proposition 23. *The $\mathbb{R}$-algebra $\mathcal{P}^{S_{m,n}}$ is generated by the elementary symmetric polynomials of the form $e_1(\mu)$ with $\mu \in \mathcal{M}_m^+$ and total degree of μ smaller than or equal to n.*

Proof. See [24, Theorem 1] for this proof. ■

We introduce a multigrading with values in $\mathbb{N}_0^m$ on $\mathcal{P}^{S_{m,n}}$: For x in the alphabet $a, b \dots, z$, we give the variable x_i the multidegree ξ_i, where ξ_i is the ith vector of the canonical basis of $\mathbb{Z}^m$. We write mdeg(P) for the multidegree of a polynomial that is homogeneous relative to this multigrading.

Example 24. Let $m := 2$ and $n := 2$. We have $\operatorname{mdeg}(a_1) = \operatorname{mdeg}(b_1) = (1, 0)$. Furthermore,

$$\begin{aligned} \operatorname{mdeg}(a_1 + b_1) &= (1, 0), \\ \operatorname{mdeg}(a_1 b_1) &= (2, 0), \\ \operatorname{mdeg}(a_2 b_2) &= (0, 2). \end{aligned}$$

Theorem 25. *Define the set $\mathcal{B}^{n,m}_{\leqslant d}$ as*

$$\mathcal{B}^{n,m}_{\leqslant d} := \left\{ \prod_{i=1}^{k} e_1(\mu_i) \;\middle|\; k \in \mathbb{N},\ \mu_i \in \mathcal{M}^+_m,\ |\operatorname{mdeg}(\mu_i)| \leqslant n, \left|\sum_{i=1}^{k} \operatorname{mdeg}(\mu_i)\right| \leqslant d \right\}.$$

For $n < d$, the set $\mathcal{B}^{n,m}_{\leqslant d}$ is a generating system for $\mathcal{P}^{S_{n,m}}_{\leqslant d}$ as an $\mathbb{R}$-vector space. For $n \geqslant d$, the set $\mathcal{B}^{n,m}_{\leqslant d}$ is a basis of $\mathcal{P}^{S_{n,m}}_{\leqslant d}$ as an $\mathbb{R}$-vector space.

Proof. We begin by enumerating the elements of $\mathcal{B}^{n,m}_{\leqslant d}$. Observe that for any collection of $(\mu_i)_{i=1}^k$ with $\mu_i \in \mathcal{M}^+_m$, we have

$$\operatorname{mdeg}\left(\prod_{i=1}^{k} e_1(\mu_i)\right) = \sum_{i=1}^{k} \operatorname{mdeg}(e_1(\mu_i)) = \sum_{i=1}^{k} \operatorname{mdeg}(\mu_i). \tag{16}$$

Given α a part of $\mathfrak{p} \in \Pi_m$, the sum of its entries is called the norm of α. We establish the mapping

$$\varphi\colon \{\mathfrak{p} \in \Pi_m \mid \text{parts of } \mathfrak{p} \text{ have norm less than } n, |s(\mathfrak{p})| \leqslant d\} \to \mathcal{B}^{n,m}_{\leqslant d} \backslash \{1\},$$
$$\mathfrak{p} = [\alpha_1, \dots, \alpha_k] \mapsto \prod_{i=1}^{k} e_1(\boldsymbol{x}^{\alpha_i}),$$

where we take all the α_i in the definition to be nonzero. It is straightforward to see that φ is a bijection. The fact that $\mathcal{B}^{n,m}_{\leqslant d}$ is a generating system for $\mathcal{P}^{S_{n,m}}_{\leqslant d}$, as an $\mathbb{R}$-vector space, is only a rewording of Proposition 23, which is all there was to show for the case $n < d$.

If $n \geqslant d$, the condition $|s(\mathfrak{p})| \leqslant d$ implies that all of the parts of $\mathfrak{p}$ have norm not greater than n, so in that case this condition is obsolete. We obtain

$$\left|\mathcal{B}^{n,m}_{\leqslant d}\right| = 1 + |\{\mathfrak{p} \in \Pi_m \mid |s(\mathfrak{p})| \leqslant d\}|,$$

which shows that $\mathcal{B}^{n,m}_{\leqslant d}$ is in fact a basis by formula (13). ■

Remark 26. In the case $m = 1$, the set $\mathcal{B}^{n,m}_{\leqslant d}$ is in fact also a basis of $\mathcal{P}^{n,m}_{\leqslant d}$ if $n < d$. This follows from the well-known fact that the number of integer partitions of $k \in \mathbb{N}$ into exactly l parts is equal to the number of integer partitions of k, where the largest part has size exactly l. However, for $m > 1$, this is not true in general. The smallest counterexample we could find occured for the parameters $m = n = 2$ and $d = 4$. Because the dimension of $\mathcal{P}^{S_{2,2}}_{\leqslant 4}$ is equal to 38, it would be cumbersome to write down the full counterexample.

At this point, we are just one small step away from the basis that we will actually use to compute the cubature formulas. We will slightly generalize the ideas behind Theorem 25.

Definition 27. *Let $(q_i)_{i=0}^{\infty}$ with $q_i \in \mathbb{R}[X]$ be a collection of univariate polynomials such that q_i is of degree i for $i \in \mathbb{N}_0$. Define*

$$\mathcal{T}_m := \{q_{j_1} \otimes q_{j_2} \otimes \cdots \otimes q_{j_m} \mid (j_1, \dots, j_m) \in \mathbb{N}_0^m\}.$$

We view $\mathcal{T}_m$ as a subset of $\mathbb{R}[X_1, \dots, X_m]$. Again, denote by $\mathcal{T}^+_m$ the elements of $\mathcal{T}_m$ of positive degree.

Corollary 28. *Define the set* $\mathcal{C}^{n,m}_{\leqslant d}$ *as*

$$\mathcal{C}^{n,m}_{\leqslant d} := \left\{ \prod_{i=1}^{k} e_1(P_i) \;\middle|\; k \in \mathbb{N},\ P_i \in \mathcal{T}_m^+,\ |\operatorname{mdeg}(P_i)| \leqslant n,\ \left|\sum_{i=1}^{k} \operatorname{mdeg}(P_i)\right| \leqslant d \right\}.$$

For $n < d$, *the set* $\mathcal{C}^{n,m}_{\leqslant d}$ *is a generating system for* $\mathcal{P}^{S_{n,m}}_{\leqslant d}$ *as an* $\mathbb{R}$*-vector space. For* $n \geqslant d$, *the set* $\mathcal{C}^{n,m}_{\leqslant d}$ *is a basis of* $\mathcal{P}^{S_{n,m}}_{\leqslant d}$ *as an* $\mathbb{R}$*-vector space.*

Proof. From the definition of $\mathcal{T}_m$, it is easy to see that $|\mathcal{C}^{n,m}_{\leqslant d}| = |\mathcal{B}^{n,m}_{\leqslant d}|$, so by Theorem 25 it is sufficient to show that $\mathcal{C}^{n,m}_{\leqslant d}$ is a generating system for $\mathcal{P}(I)^{S_{n,m}}_{\leqslant d}$. Let $\mu_1, \dots, \mu_{k*}$ $(P_1, \dots, P_{k*})$ be an enumeration of all polynomials $\mu \in \mathcal{M}_m^+$ $(P \in \mathcal{T}_m^+)$ such that $|\operatorname{mdeg}(\mu)| \leqslant \min(n,d)$ $(|\operatorname{mdeg}(P)| \leqslant \min(n,d))$, respectively. We show that $\{e_1(P_1), \dots, e_1(P_{k*})\}$ generates $\mathcal{P}^{S_{m,n}}_{\leqslant d}$ as an $\mathbb{R}$-algebra. Let $M \in \mathcal{P}(I)^{S_{n,m}}_{\leqslant d}$. By Theorem 25, there is a polynomial $Q \in \mathbb{R}[X_1, \dots, X_{k*}]$ such that

$$M = Q(e_1(\mu_1), \dots, e_1(\mu_{k*})).$$

As both $\mathcal{M}_m$ and $\mathcal{T}_m$ are bases of $\mathbb{R}[X_1, \dots, X_m]$, we may write any μ_i as a linear combination of $P_1, \dots, P_{k*}$ as

$$\mu_i = \sum_{j=1}^{k*} a_{i,j} P_j \quad \forall i \in \{1, \dots, k^*\},$$

where $a_{i,j} \in \mathbb{R}$. This implies

$$e_1(\mu_i) = \sum_{j=1}^{k*} a_{i,j} \cdot e_1(P_j) \quad \forall i \in \{1, \dots, k^*\}.$$

We define $\hat{Q} \in \mathbb{R}[X_1, \dots, X_{k*}]$ as

$$\hat{Q}(X_1, \dots, X_{k*}) := Q\left(\sum_{j=1}^{k*} a_{1,j} X_j, \dots, \sum_{j=1}^{k*} a_{k*,j} X_j \right),$$

which implies

$$\hat{Q}(e_1(P_1), \dots, e_1(P_{k*})) = Q(e_1(\mu_1), \dots, e_1(\mu_{k*})) = M,$$

concluding the proof. ∎

5.2. Implementation of the basic scheme.

5.2.1. Basis generation. For the remainder of this section d is assumed to be odd. As indicated earlier, we will work with a "basis" of the form $\mathcal{C}^{n,m}_{\leqslant d}$ (as in Corollary 28) from now on. The fact that $\mathcal{C}^{n,m}_{\leqslant d}$ is not a basis if $n < d$ is of no concern, as it is sufficient to integrate all polynomials of a generating system of $\mathcal{P}(I)^{S_{n,m}}_{\leqslant d}$ exactly in order to integrate all polynomials of $\mathcal{P}(I)^{S_{n,m}}_{\leqslant d}$ exactly.

Furthermore, in the cases we are interested in, d is typically small, so the additional computational overhead of having a larger A-matrix than necessary when $n < d$ is not problematic. For the assembly of the matrix A (as defined in subsection 4.1), we only have to evaluate polynomials of the form $e_1(P)$ with $P \in \mathcal{T}_m^+$, that means we only evaluate $\binom{m+d}{d} - 1$ polynomials consisting of n terms. This is a significant advantage compared to monomial multisymmetric polynomials, which could have a length of $d!\binom{n}{d}$ terms in the worst case.

We introduced the $\mathcal{C}_{\leqslant d}^{n,m}$-polynomials to alleviate a flaw of the $\mathcal{B}_{\leqslant d}^{n,m}$-polynomials: None of the polynomials of the form $e_1(\mu)$ with $\mu \in \mathcal{M}_m^+$ evaluate to 0 on $I \backslash \{\mathbf{0}\}$, resulting in a fully dense A-matrix. To force more entries of A to be 0, we make the following choice for the q_i (as defined in Definition 27): Let y_j denote the nodes of the univariate degree-d Gaussian quadrature formula on $[0, 1]$ for $j \in \{1, \ldots, \frac{d+1}{2}\}$. Define

$$q_j := \prod_{i=1}^{j} (x - y_i), \quad j \in \left\{0, \ldots, \frac{d+1}{2}\right\}.$$

We chose q_j for $j > \frac{d+1}{2}$ to be a multiple of $q_{(d+1)/2}$. The precise form of the multiples of $q_{(d+1)/2}$ does not have a great impact on the sparsity of A. Using this particular basis, we obtained 30% to 72% zero entries, decreasing as n and d increase.

5.2.2. Calculating integrals of basis polynomials. The usage of a basis like $\mathcal{C}_{\leqslant d}^{n,m}$ increases the difficulty of calculating the integrals of basis polynomials (the right-hand side vector $\boldsymbol{b}$ in section 4.1). However, we can still find closed forms for the integrals whose complexity with respect to n is essentially constant, assuming that d is small.

Definition 29. *Let l be a positive integer. Define the set $\mathfrak{P}(\{1, \ldots, l\})$ to be the set of all (set-)partitions of $\{1, \ldots, l\}$.*

Lemma 30. *Let $P_i \in \mathcal{T}_m^+$ for all $i \in \{1, \ldots, l\}$. Then the equation*

$$\int_{[0,1]^s} \prod_{i=1}^{l} e_1(P_i)\, \mathrm{d}\boldsymbol{x} = \sum_{\substack{\mathcal{D} \in \mathfrak{P}(\{1,\ldots,l\}) \\ |\mathcal{D}| \leqslant n}} n(n-1)\cdots(n-|\mathcal{D}|+1) \prod_{B \in \mathcal{D}} \int_{[0,1]^m} \prod_{i \in B} P_i(\boldsymbol{y})\, \mathrm{d}\boldsymbol{y}$$

holds.

Proof. We calculate

$$\begin{aligned}\int_{[0,1]^s} \prod_{i=1}^{l} e_1(P_i)\, \mathrm{d}\boldsymbol{x} &= \int_{[0,1]^s} \prod_{i=1}^{l} (P_i(\boldsymbol{a}) + \cdots + P_i(\boldsymbol{z}))\, \mathrm{d}\boldsymbol{x} \\ &= \int_{[0,1]^s} \sum_{(\boldsymbol{x}_1,\ldots,\boldsymbol{x}_l) \in \{\boldsymbol{a},\ldots,\boldsymbol{z}\}^l} \prod_{i=1}^{l} P_i(\boldsymbol{x}_i)\, \mathrm{d}\boldsymbol{x}.\end{aligned}$$

We can treat $\boldsymbol{a}, \ldots, \boldsymbol{z}$ as dummy variables for integration; therefore, we can partition $\{1, \ldots, l\}$ into parts that have the same integral without having to remember the variable name. For example, if $l = 4$, then the term $P_1(\boldsymbol{a})P_2(\boldsymbol{a})P_3(\boldsymbol{b})P_4(\boldsymbol{c})$ has the same integral as $P_1(\boldsymbol{c})P_2(\boldsymbol{c})P_3(\boldsymbol{a})P_4(\boldsymbol{b})$. Both terms would induce the partition $\{\{1, 2\}, \{3\}, \{4\}\}$. For any partition $\mathcal{D}$ of $\{1,\dots,l\}$ into at most n parts, there are $n(n-1)\cdots(n-|\mathcal{D}|+1)$ terms associated with this partition and, therefore, they have the same integral, which is equal to $\prod_{B\in\mathcal{D}}\int_{[0,1]^m}\prod_{j\in B}P_j(\boldsymbol{y})d\boldsymbol{y}$. Hence we conclude

$$\int_{[0,1]^s}\prod_{i=1}^{l}e_1(P_i)\,\mathrm{d}\boldsymbol{x} = \sum_{\substack{\mathcal{D}\in\mathfrak{P}(\{1,\dots,l\}),\\ |\mathcal{D}|\leqslant n}} n(n-1)\cdots(n-|\mathcal{D}|+1)\prod_{B\in\mathcal{D}}\int_{[0,1]^m}\prod_{i\in B}P_i(\boldsymbol{y})\,\mathrm{d}\boldsymbol{y}. \qquad \blacksquare$$

Using Lemma 30, we only have to calculate integrals of products of $P_i\in\mathcal{T}_m^+$ in m dimensions in order to integrate the basis polynomials of $\mathcal{C}_{\leqslant d}^{n,m}$. This comes at the cost of iterating over partitions of $\{1,\dots,l\}$, but as $l\leqslant d$ and d typically is small, this is usually inexpensive.

5.2.3. Generating nodes modulo $S_{m,n}$. Let $\mathcal{N}_0$ denote the nodes of a univariate degree-d Gaussian quadrature formula. Then, taking all n-combinations of $\times_{i=1}^{m}\mathcal{N}_0$ with repetitions gives a full representative system of the nodes modulo $S_{m,n}$. Therefore, the number of orbits k of $\times_{i=1}^{s}\mathcal{N}_0$ under the natural action of $S_{m,n}$ is equal to

$$k=\binom{n+\left(\frac{d+1}{2}\right)^m-1}{n}. \tag{17}$$

5.2.4. Saving memory. The limiting factor when the algorithm is applied in the way we propose is memory consumption. For large n, the number k of columns in the matrix $A\in\mathbb{R}^{n^*\times k}$ is usually very large (see (17)), and executing the simplex algorithm with a large constraint matrix uses sizable amounts of memory. Observe that in the critical cases we have $k\gg n^*$, and n^* is an upper bound for the number of nonzero entries in a solution. Leaving out columns of A corresponds to searching for solutions that have entries equal to 0 at the nodes associated with the columns that were left out. Heuristically, as $k\gg n^*$, we should be able to leave out a lot of columns and still obtain results. This motivates the following algorithm.

Algorithm 5.1 Reduction of Columns.

```
Initialize $A_0 := \varnothing$
while $A_0$ infeasible according to the basic scheme in subsection 4.1 do
    Add a small subset of columns of $A$ to $A_0$
end while
Compute solution of the system
```

This procedure proved to be very efficient, enabling us to calculate lots of formulas that would otherwise have been out of reach.

6. Numerical results.

We would like to preface this section with a quote [18, section 11, Paragraph 2]:

"When good results are obtained in integrating a high-dimensional function, we should conclude first of all that an especially tractable integrand was tried

Table 1

The left table shows the required amount of cubature nodes for a fully symmetric cubature formula $(m = 1)$ *of degree d. The right table shows the required amount of cubature nodes for a multisymmetric cubature formula* $(m = 2)$ *of degree d.*

	degree d				
n	3	5	7	9	11
1	2	3	4	5	6
2	3	6	10	15	21
3	4	9	18	30	48
4	3	9	24	46	46
5	3	11	28	38	51
6	4	12	30	38	57
7	4	12	24	43	52
8	4	12	25	42	56

	degree d			
n	3	5	7	9
1	4	9	16	25
2	6	30	100	225
3	8	67	248	714
4	13	84	367	1196
5	13	90	432	1659
6	13	90	457	1581
7	13	90	465	1618
8	13	90	465	1564

and not that a generally successful method has been found. A secondary conclusion is that we might have made a very good choice in selecting an integration method to exploit whatever features of f made it tractable."

In this section, we compare the cubature formula for multisymmetric functions obtained with the method presented in section 5 to other numerical integration methods such as quasi-Monte Carlo methods, e.g., the Sobol sequence [3, 15], the Clenshaw–Curtis sparse grid, and tensor-product formulas. There are quasi-Monte Carlo methods that exploit smoothness of functions better than the Sobol sequence [8]. However, for ease of implementation we chose the Sobol sequence, which is well known and implementations are readily available in many different software libraries.

The code for generating the cubature formulas (written in the Julia language) for multisymmetric functions as well as the cubature formulas themselves can be found in a GitHub repository [19].

The amount of function evaluations for a proposed cubature formula of degree d can be looked up in Table 1. We refer by N to the number of function evaluations of the Monte Carlo methods and sparse grid method. As a reminder, we have $s = mn$, where s is the dimension, n corresponds to the number of exchangeable m-tuples. In this section, we compare the proposed formulas of a given degree d to the other techniques which have the amount of evaluations fixed at some number N.

In the implementation it became apparent that the numerically most costly parts of the algorithm proposed in section 5 were finding a feasible solution to the possibly large system of linear equations (7), e.g., via linear programming and the computation of the integrals of the basis polynomials as described in Lemma 30. In section 5.2.4, an efficient work-around for the former problem was proposed using that the system is greatly overdetermined. The computational cost to calculate the integrals of the basis polynomials increases exponentially in the maximal degree d—which is, in fact, a weakness of this algorithm. Indeed, this was the major restriction while computing formulas of higher degree. The great advantage of this approach is that the amount of necessary evaluations, e.g., the amount of cubature nodes, can be bounded by a constant for fixed m and d and all n. This can be observed in Table 1. However, in our implementation, the system tends to become numerically unstable as n or d

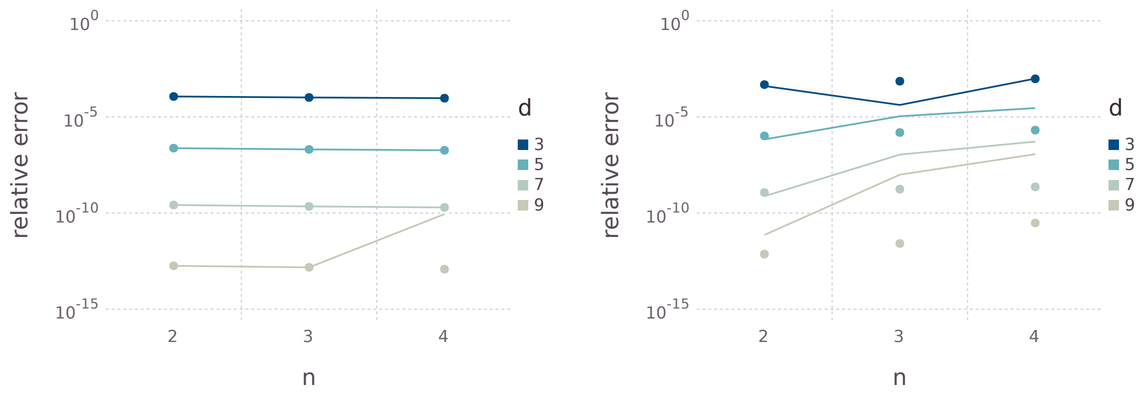

Figure 1. *Relative error of g_1 (left) and g_2 (right) as a function of dimension n compared to tensor-product quadrature formulas indicated by circles. The degree of the formulas used is denoted by d.*

grows large, e.g., for $n \geqslant 100$ (resp., $d \geqslant 11$ and $m = 1$), since the value range of (7) becomes wider and wider.

6.1. Low-dimensional test cases. In the multisymmetric case ($m = 2$), we computed formulas up to a maximal degree of $d = 9$. The following multisymmetric test integrands

$$g_1(x_1, y_1, \ldots, x_n, y_n) := \sum_i^n \left(\exp\left(\frac{x_i}{10}\right) + \exp(y_i) + \frac{1}{2} \sum_{j \neq i}^n \exp\left(\frac{x_i x_j}{10}\right) + \exp(y_i y_j) \right),$$

$$g_2(x_1, y_1, \ldots, x_n, y_n) := \sin\left(\sum_i^n \frac{x_i}{10} + y_i \right),$$

$$g_3(x_1, y_1, \ldots, x_n, y_n) := \exp\left(\sum_i^n -\frac{x_i^2}{10} - y_i^2 \right),$$

$$g_4(x_1, y_1 \ldots, x_n, y_n) := \frac{1}{\sqrt{\sum_i^n \frac{x_i}{10} + y_i}}$$

were examined, whose integrals can be derived easily analytically.

Figures 1 and 2 show a comparison of Gauss–Legendre tensor-product formulas and the proposed cubature rules. It is to expect that the proposed formulas fare worse than the tensor-product rule, since more (multisymmetric) polynomials are exactly integrated by the latter one. For example, if $n = 2$, $m = 1$, and $d = 2$, the polynomial x^2y^2 would be exactly integrated by the product rule but not by the proposed one. As shown in Theorem 7, the Taylor expansion of a G-invariant function at a G-invariant expansion point is G-invariant, again. This fact stands out particularly for g_1 and g_3, where the dominant terms in the expansion are integrated exactly by the proposed formulas as well. The results show that the proposed formulas yield comparably good results for the functions g_1 and g_3.

We attribute the fact that the error does not converge in a better fashion, as shown in Figures 3 and 4, to a numerical instability of our approach and the choice of polynomials which

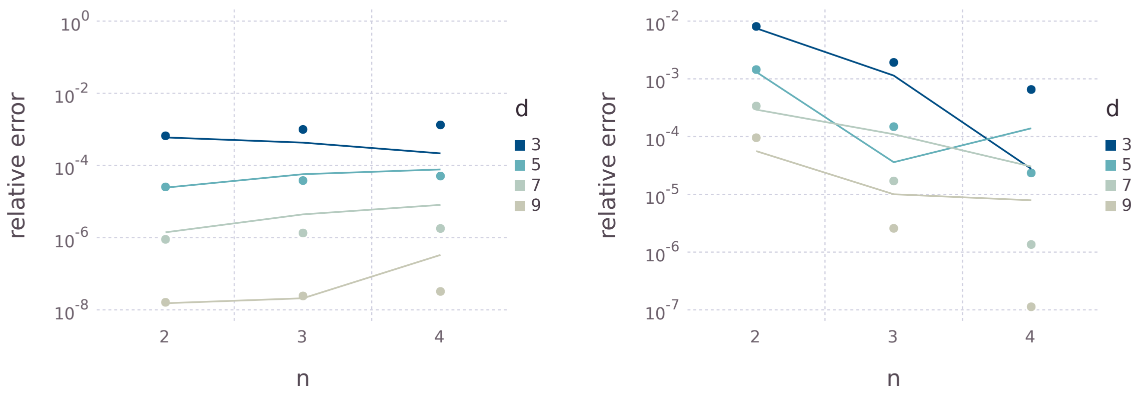

Figure 2. *Relative error of g_3 (left) and g_4 (right) as a function of dimension n compared to tensor-product quadrature formulas indicated by circles. The degree of the formulas used is denoted by d.*

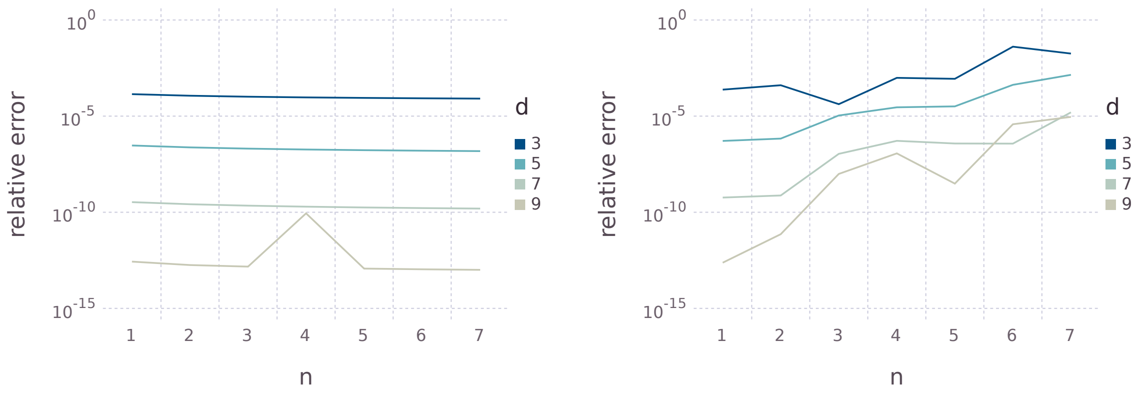

Figure 3. *Relative error of g_1 (left) and g_2 (right) as a function of dimension n. The degree of the formulas used is denoted by d.*

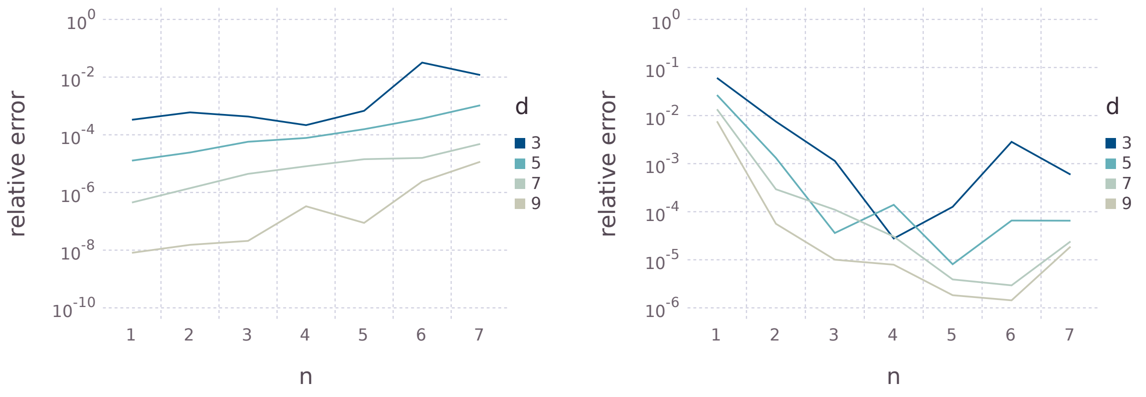

Figure 4. *Relative error of g_3 (left) and g_4 (right) as a function of dimension n. The degree of the formulas used is denoted by d.*

Table 2

Comparison of the relative error of a quasi-Monte Carlo method of N samples and the multisymmetric cubature formula of degree d. The test integrand is g_1.

	Sym. cubature		Quasi-Monte Carlo		
n	$d = 5$	$d = 9$	$N = 10^2$	$N = 10^3$	$N = 10^4$
1	$2.9 \cdot 10^{-7}$	$2.7 \cdot 10^{-13}$	$4.3 \cdot 10^{-4}$	$1.6 \cdot 10^{-4}$	$4.8 \cdot 10^{-5}$
2	$2.3 \cdot 10^{-7}$	$1.8 \cdot 10^{-13}$	$2.9 \cdot 10^{-4}$	$4.0 \cdot 10^{-4}$	$4.3 \cdot 10^{-5}$
3	$2.0 \cdot 10^{-7}$	$1.5 \cdot 10^{-13}$	$1.2 \cdot 10^{-3}$	$1.4 \cdot 10^{-4}$	$3.0 \cdot 10^{-5}$
4	$1.8 \cdot 10^{-7}$	$8.8 \cdot 10^{-11}$	$8.6 \cdot 10^{-4}$	$1.4 \cdot 10^{-4}$	$4.2 \cdot 10^{-6}$
5	$1.7 \cdot 10^{-7}$	$1.2 \cdot 10^{-13}$	$1.5 \cdot 10^{-3}$	$1.5 \cdot 10^{-4}$	$4.7 \cdot 10^{-6}$

Table 3

Comparison of the relative error of a quasi-Monte Carlo method of N samples and the multisymmetric cubature formula of degree d. The test integrand is g_2.

	Sym. cubature		Quasi-Monte Carlo		
n	$d = 5$	$d = 9$	$N = 10^2$	$N = 10^3$	$N = 10^4$
1	$5.1 \cdot 10^{-7}$	$2.4 \cdot 10^{-13}$	$1.3 \cdot 10^{-3}$	$3.1 \cdot 10^{-4}$	$1.0 \cdot 10^{-4}$
2	$6.7 \cdot 10^{-7}$	$7.2 \cdot 10^{-12}$	$3.9 \cdot 10^{-3}$	$5.3 \cdot 10^{-4}$	$1.2 \cdot 10^{-5}$
3	$1.1 \cdot 10^{-5}$	$9.8 \cdot 10^{-9}$	$3.9 \cdot 10^{-3}$	$1.6 \cdot 10^{-3}$	$1.3 \cdot 10^{-4}$
4	$2.8 \cdot 10^{-5}$	$1.1 \cdot 10^{-7}$	$2.5 \cdot 10^{-3}$	$3.9 \cdot 10^{-3}$	$2.4 \cdot 10^{-4}$
5	$3.2 \cdot 10^{-5}$	$3.1 \cdot 10^{-9}$	$3.4 \cdot 10^{-3}$	$9.6 \cdot 10^{-3}$	$9.5 \cdot 10^{-4}$

Table 4

Comparison of the relative error of a quasi-Monte Carlo method of N samples and the multisymmetric cubature formula of degree d. The test integrand is g_3.

	Sym. cubature		Quasi-Monte Carlo		
n	$d = 5$	$d = 9$	$N = 10^2$	$N = 10^3$	$N = 10^4$
1	$1.2 \cdot 10^{-5}$	$8.1 \cdot 10^{-9}$	$1.4 \cdot 10^{-3}$	$2.7 \cdot 10^{-4}$	$7.4 \cdot 10^{-5}$
2	$2.4 \cdot 10^{-5}$	$1.5 \cdot 10^{-8}$	$2.3 \cdot 10^{-3}$	$3.1 \cdot 10^{-4}$	$1.2 \cdot 10^{-4}$
3	$5.7 \cdot 10^{-5}$	$2.1 \cdot 10^{-8}$	$1.1 \cdot 10^{-4}$	$4.3 \cdot 10^{-4}$	$5.0 \cdot 10^{-5}$
4	$7.7 \cdot 10^{-5}$	$3.3 \cdot 10^{-7}$	$7.7 \cdot 10^{-4}$	$1.7 \cdot 10^{-3}$	$9.7 \cdot 10^{-4}$
5	$1.5 \cdot 10^{-4}$	$8.8 \cdot 10^{-8}$	$6.8 \cdot 10^{-3}$	$2.0 \cdot 10^{-3}$	$1.7 \cdot 10^{-4}$

are exactly integrated. When comparing to the tensor-product rule, one has to note that the active dimension ranges from 4 (in the case of $n = 2$) to 12 (in the case of $n = 6$), which leads to a required number of evaluations of 3^4 up to 3^{12} for the tensor-product formula of degree $d = 5$, which is significantly more than the number of evaluations needed for the multisymmetric cubature formula considering that the (worst-case) amount of necessary evaluations remains constant for $n \geqslant d$ as shown in Table 1.

Tables 2, 3, 4, and 5 show a comparison of multisymmetric cubature formulas to a quasi-Monte Carlo method. Considering error and number of evaluations, the multisymmetric cubature formula seems to be superior to the quasi-Monte Carlo method in both aspects. Tables 6, 7, 8, and 9 show a comparison of multisymmetric cubature formulas to a Clenshaw–Curtis sparse grid. The sparse grid was constructed adaptively [12], where the algorithm stopped after the first iteration step when N evaluations are exceeded. This is not a very natural way of applying an adaptive sparse grid method, but otherwise the runtime and

Table 5

Comparison of the relative error of a quasi-Monte Carlo method of N samples and the multisymmetric cubature formula of degree d. The test integrand is g_4.

	Sym. cubature		Quasi-Monte Carlo		
n	$d = 5$	$d = 9$	$N = 10^2$	$N = 10^3$	$N = 10^4$
1	$2.6 \cdot 10^{-2}$	$7.6 \cdot 10^{-3}$	$2.7 \cdot 10^{-2}$	$5.6 \cdot 10^{-3}$	$6.8 \cdot 10^{-4}$
2	$1.3 \cdot 10^{-3}$	$5.6 \cdot 10^{-5}$	$1.0 \cdot 10^{-2}$	$1.9 \cdot 10^{-3}$	$2.9 \cdot 10^{-4}$
3	$3.6 \cdot 10^{-5}$	$1.0 \cdot 10^{-5}$	$6.8 \cdot 10^{-3}$	$1.7 \cdot 10^{-3}$	$2.0 \cdot 10^{-4}$
4	$1.4 \cdot 10^{-4}$	$7.9 \cdot 10^{-6}$	$4.9 \cdot 10^{-3}$	$1.2 \cdot 10^{-3}$	$1.9 \cdot 10^{-4}$
5	$8.0 \cdot 10^{-6}$	$1.8 \cdot 10^{-6}$	$4.1 \cdot 10^{-3}$	$1.0 \cdot 10^{-3}$	$1.3 \cdot 10^{-4}$

Table 6

Comparison of the relative error of a classical sparse grid method of at least N evaluations and the multisymmetric cubature formula of degree d. The test integrand is g_1.

	Sym. cubature		Sparse grid		
n	$d = 5$	$d = 9$	$N = 10^2$	$N = 5 \cdot 10^2$	$N = 10^3$
1	$2.9 \cdot 10^{-7}$	$2.7 \cdot 10^{-13}$	$4.8 \cdot 10^{-16}$	$6.9 \cdot 10^{-13}$	$1.8 \cdot 10^{-13}$
2	$2.3 \cdot 10^{-7}$	$1.8 \cdot 10^{-13}$	$4.9 \cdot 10^{-6}$	$6.9 \cdot 10^{-8}$	$7.5 \cdot 10^{-13}$
3	$2.0 \cdot 10^{-7}$	$1.5 \cdot 10^{-13}$	$8.5 \cdot 10^{-4}$	$1.0 \cdot 10^{-7}$	$7.2 \cdot 10^{-8}$
4	$1.8 \cdot 10^{-7}$	$8.8 \cdot 10^{-11}$	$1.3 \cdot 10^{-3}$	$6.2 \cdot 10^{-6}$	$2.4 \cdot 10^{-6}$
5	$1.7 \cdot 10^{-7}$	$1.2 \cdot 10^{-13}$	$1.6 \cdot 10^{-3}$	$9.4 \cdot 10^{-6}$	$6.3 \cdot 10^{-6}$

Table 7

Comparison of the relative error of a classical sparse grid method of at least N evaluations and the multisymmetric cubature formula of degree d. The test integrand is g_2.

	Sym. cubature		Sparse grid		
n	$d = 5$	$d = 9$	$N = 10^2$	$N = 5 \cdot 10^2$	$N = 10^3$
1	$5.1 \cdot 10^{-7}$	$2.4 \cdot 10^{-13}$	$1.9 \cdot 10^{-15}$	$2.8 \cdot 10^{-14}$	$9.6 \cdot 10^{-13}$
2	$6.7 \cdot 10^{-7}$	$7.2 \cdot 10^{-12}$	$9.2 \cdot 10^{-6}$	$1.3 \cdot 10^{-8}$	$8.2 \cdot 10^{-12}$
3	$1.1 \cdot 10^{-5}$	$9.8 \cdot 10^{-9}$	$1.1 \cdot 10^{-4}$	$6.5 \cdot 10^{-7}$	$6.3 \cdot 10^{-7}$
4	$2.8 \cdot 10^{-5}$	$1.1 \cdot 10^{-7}$	$5.9 \cdot 10^{-3}$	$2.9 \cdot 10^{-4}$	$9.1 \cdot 10^{-6}$
5	$3.2 \cdot 10^{-5}$	$3.1 \cdot 10^{-9}$	$1.4 \cdot 10^{-2}$	$9.6 \cdot 10^{-4}$	$7.4 \cdot 10^{-4}$

amount of function evaluations would not have been comparable to that of our formulas or a quasi-Monte Carlo method. Increasing the dimensionality of the problem seems to drastically decrease the accuracy of the sparse grid, whereas the multisymmetric cubature formula does not expose this behavior as much. In terms of efficiency, the multisymmetric cubature rule seems to be superior to both the quasi-Monte Carlo and sparse-grid methods.

6.2. High-dimensional test cases. For the high-dimensional comparison, we use Genz functions [11]. In the Genz functions, u is a location parameter and a is an effective parameter, which is normed according to Table 10. The norming ensures that the difficulty of the integration problems remains more or less constant with respect to the number of dimensions [11, 22]. The parameters are chosen randomly under the conditions that the functions remain multisymmetric and that the norm of a satisfies the value prescribed in Table 10. Since these parameters are chosen randomly, we use the standard Monte Carlo method for computing the

Table 8

Comparison of the relative error of a classical sparse grid method of at least N evaluations and the multisymmetric cubature formula of degree d. The test integrand is g_3.

	Sym. cubature		Sparse grid		
n	$d = 5$	$d = 9$	$N = 10^2$	$N = 5 \cdot 10^2$	$N = 10^3$
1	$1.2 \cdot 10^{-5}$	$8.1 \cdot 10^{-9}$	$6.9 \cdot 10^{-13}$	$3.6 \cdot 10^{-13}$	$1.3 \cdot 10^{-12}$
2	$2.4 \cdot 10^{-5}$	$1.5 \cdot 10^{-8}$	$4.6 \cdot 10^{-5}$	$3.0 \cdot 10^{-7}$	$1.6 \cdot 10^{-9}$
3	$5.7 \cdot 10^{-5}$	$2.1 \cdot 10^{-8}$	$2.9 \cdot 10^{-4}$	$4.9 \cdot 10^{-6}$	$4.5 \cdot 10^{-6}$
4	$7.7 \cdot 10^{-5}$	$3.3 \cdot 10^{-7}$	$8.4 \cdot 10^{-3}$	$6.3 \cdot 10^{-5}$	$5.6 \cdot 10^{-5}$
5	$1.5 \cdot 10^{-4}$	$8.8 \cdot 10^{-8}$	$2.1 \cdot 10^{-2}$	$1.9 \cdot 10^{-3}$	$1.3 \cdot 10^{-3}$

Table 9

Comparison of the relative error of a classical sparse grid method of at least N evaluations and the multisymmetric cubature formula of degree d. The test integrand is g_4.

	Sym. cubature		Sparse grid		
n	$d = 5$	$d = 9$	$N = 10^2$	$N = 5 \cdot 10^2$	$N = 10^3$
1	$2.6 \cdot 10^{-2}$	$7.6 \cdot 10^{-3}$	$1.8 \cdot 10^{-3}$	$9.9 \cdot 10^{-3}$	$1.2 \cdot 10^{-3}$
2	$1.3 \cdot 10^{-3}$	$5.6 \cdot 10^{-5}$	$3.6 \cdot 10^{-3}$	$8.4 \cdot 10^{-4}$	$2.2 \cdot 10^{-4}$
3	$3.6 \cdot 10^{-5}$	$1.0 \cdot 10^{-5}$	$1.8 \cdot 10^{-3}$	$3.8 \cdot 10^{-4}$	$3.8 \cdot 10^{-4}$
4	$1.4 \cdot 10^{-4}$	$7.9 \cdot 10^{-6}$	$2.5 \cdot 10^{-3}$	$7.0 \cdot 10^{-4}$	$2.3 \cdot 10^{-4}$
5	$8.0 \cdot 10^{-6}$	$1.8 \cdot 10^{-6}$	$2.0 \cdot 10^{-3}$	$4.7 \cdot 10^{-4}$	$3.9 \cdot 10^{-4}$

relative root mean square error (rRMSE),

$$\text{rRMSE} := \sqrt{\frac{\mathbb{E}\left[\left(\int_{[0,1]^{nm}} f(x)\mathrm{d}x - Q(f)\right)^2\right]}{\mathbb{E}\left[\left(\int_{[0,1]^{nm}} f(x)\mathrm{d}x\right)^2\right]}}. \tag{18}$$

The number of samples is chosen sufficiently large such that we obtain a 99% confidence interval for a computation error of less than 1%.

The calculation of the exact integral for f_3, Genz's corner-peak function, demands a significant amount of computational work, and is numerically unstable for $n > 20$. Therefore, we chose to omit this function.

In this test case, we chose $m := 1$. Figures 5, 6, 7, 8, and 9 compare the fully symmetric cubature rule to standard Monte Carlo and quasi-Monte Carlo with 10^4 samples each and a sparse grid with more than 10^3 evaluations. We would like to point out that the worst-case bound on the number of evaluations for the multisymmetric cubature rules is 19 in the case $d = 5$ and 45 in the case $d = 7$, a comparably small amount. The Genz functions are used as test integrands. These results should be taken with a grain of salt, since even though the free parameters are chosen in a way such that the difficulty to integrate remains constant, the integrands essentially converge to a constant as n grows. It is notable that the standard and quasi-Monte Carlo methods show better results for smaller dimensions. For smaller dimensions, the test integrands are less regular (e.g., the oscillatory function oscillates very quickly for small n and the Gaussian function has a very small variance), which favors Monte Carlo methods. This explanation is consistent with the observation that sparse grids

Table 10
Genz functions.

Integrand family	$\|a\|_1$
$f_1(x) := \cos\left(2\pi u_1 + \sum_i a_i x_i\right)$	$\frac{110}{\sqrt{(nm)^3}}$
$f_2(x) := \prod_i \frac{1}{a_i^{-2} + (x_i - u_i)^2}$	$\frac{600}{(nm)^2}$
$f_3(x) := \left(1 + \sum_i a_i x_i\right)^{(n \cdot m+1)}$	$\frac{600}{(nm)^2}$
$f_4(x) := \exp\left(-\sum_i a_i^2 (x_i - u_i)^2\right)$	$\frac{100}{nm}$
$f_5(x) := \exp\left(-\sum_i a_i \lvert x_i - u_i \rvert\right)$	$\frac{150}{(nm)^2}$
$f_6(x) := \begin{cases} 0, & x_1 > u_1 \text{ or } x_2 > u_2, \\ \exp\left(\sum_i a_i x_i\right), & \text{otherwise} \end{cases}$	$\frac{100}{(nm)^2}$

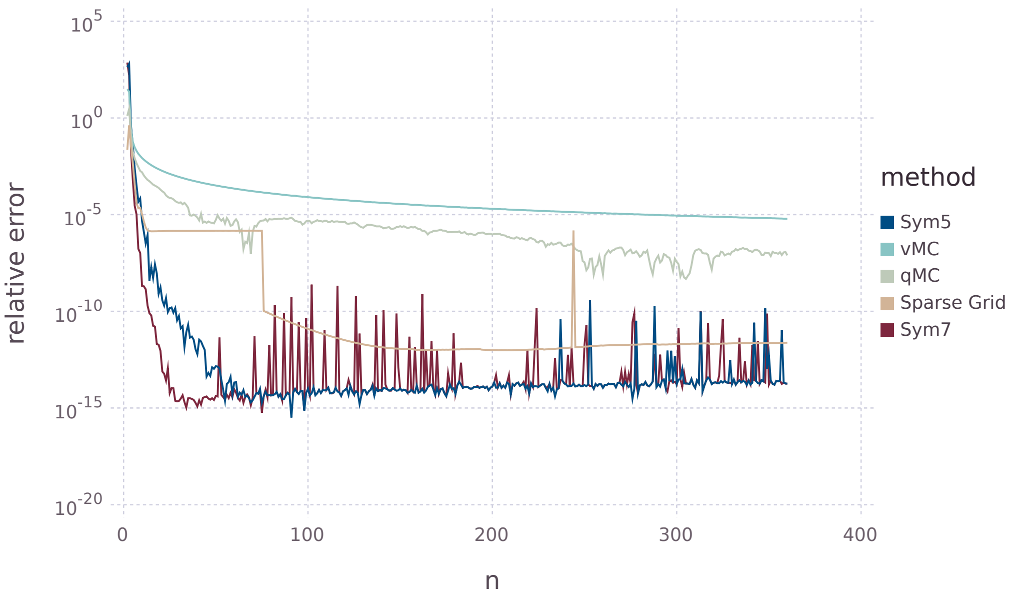

Figure 5. *Relative error of different integration methods for Genz's n-dimensional oscillatory function f_1.*

and multisymmetric cubature formulas outperform Monte Carlo methods for very regular integrands, while less regular integrand families (see Figures 8 and 9) seem to favor Monte Carlo methods.

6.3. A stochastic partial differential equation. In this section, we compute the expectation of the solution of a stochastic elliptic partial differential equation using our proposed formulas as well as a quasi-Monte Carlo method and compare the accuracy of the results obtained this way.

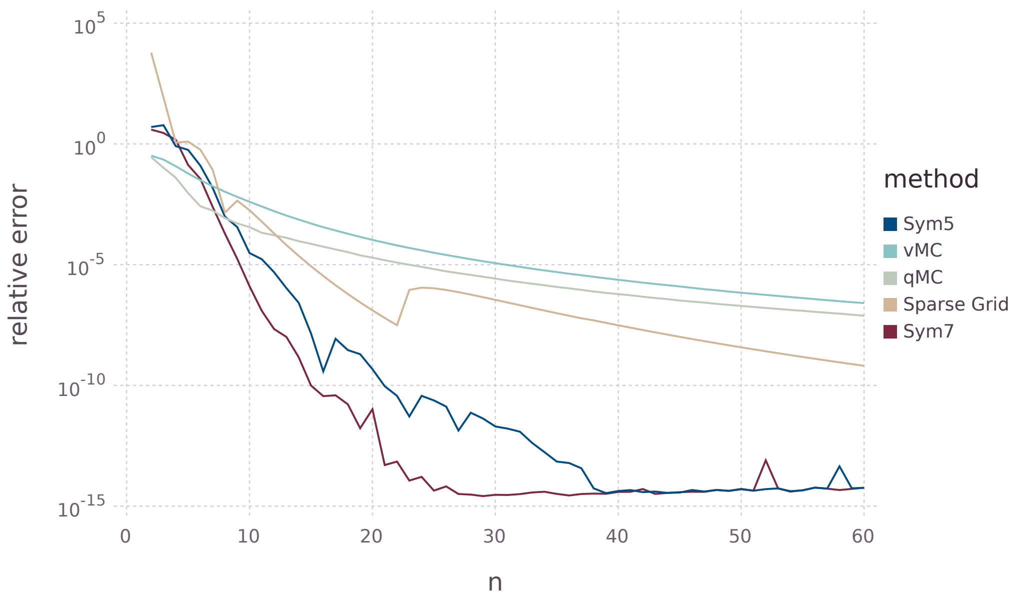

Figure 6. *Relative error of different integration methods for Genz's n-dimensional product-peak function f_2.*

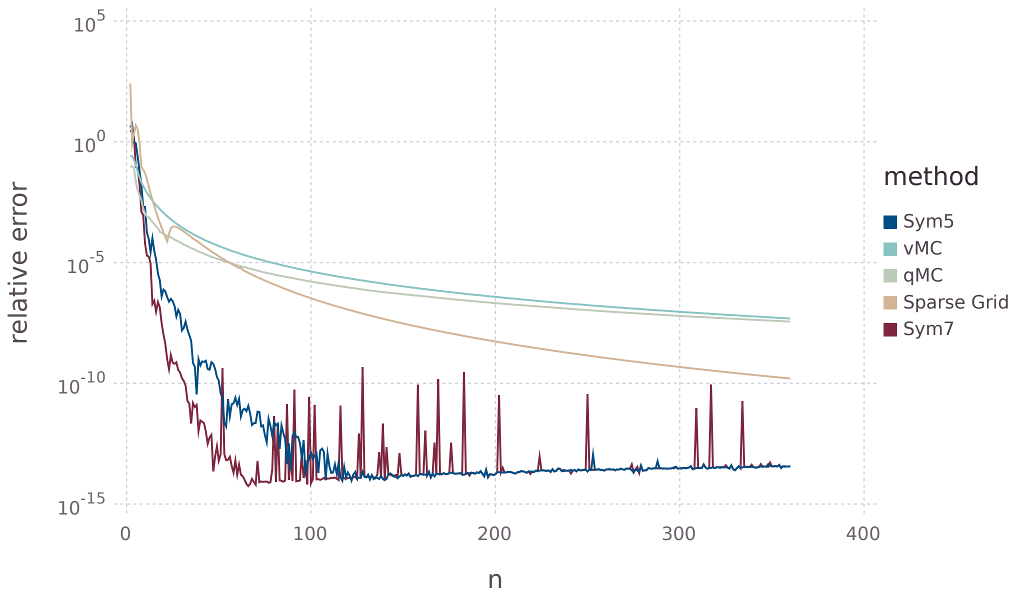

Figure 7. *Relative error of different integration methods for Genz's n-dimensional Gaussian function f_4.*

Again, we let $m := 1$ and define $D := [0,1]^2$ and a probability space Ω. We consider the problem

$$-\Delta u(x,y,\omega) = f(x,y,\omega) \quad \text{in } D \times \Omega, \tag{19a}$$

$$u(x,y,\omega) = g(x,y) \quad \text{on } \partial D \times \Omega, \tag{19b}$$

where we assume that f and g are sufficiently smooth in (x,y) such that the solution is classic.

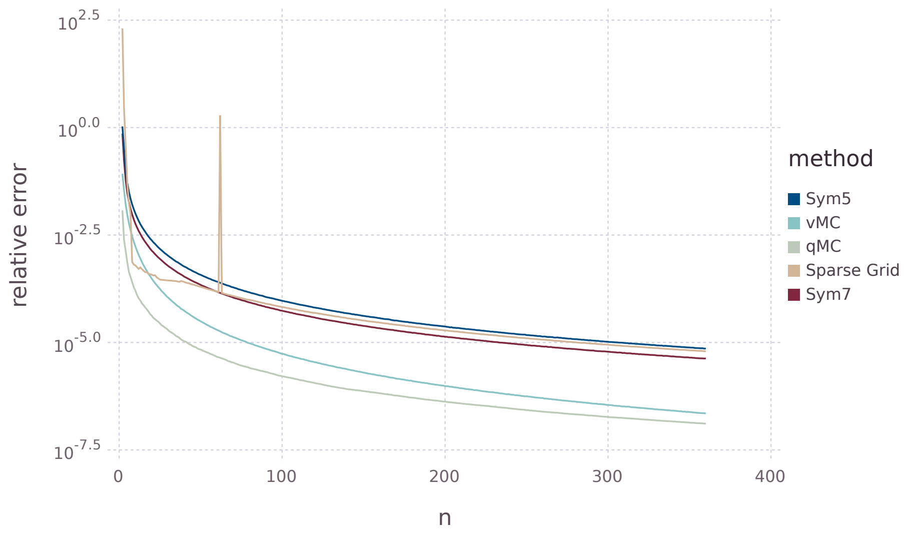

Figure 8. *Relative error of different integration methods for Genz's n-dimensional continuous function f_5.*

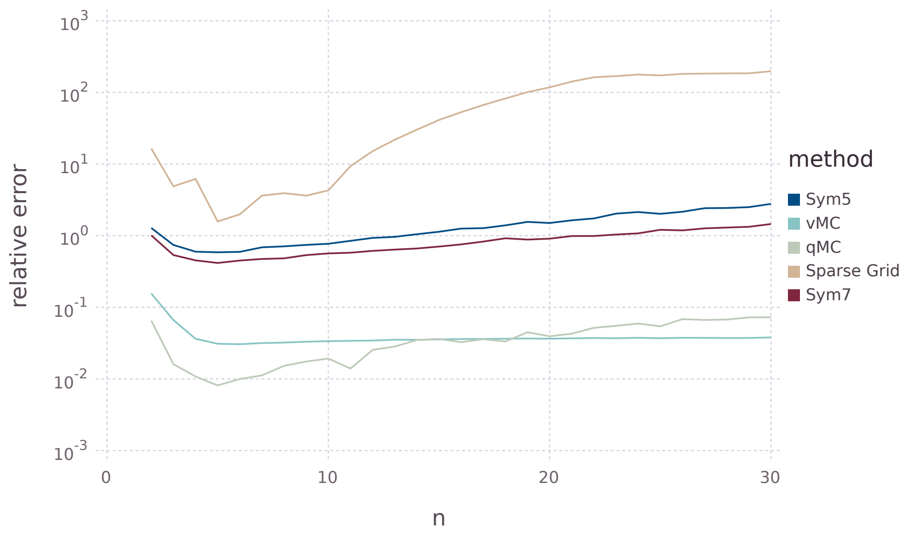

Figure 9. *Relative error of different integration methods for Genz's n-dimensional discontinuous function f_6.*

Integrating with respect to $dP(\omega)$, we obtain

$$\begin{aligned} -\int_\Omega \Delta u(x,y,\omega)dP(\omega) &= \int_\Omega f(x,y,\omega)dP(\omega) &&\text{in } D,\\ \int_\Omega u(x,y,\omega)dP(\omega) &= g(x,y) &&\text{on } \partial D. \end{aligned}$$

Since we assumed u to be sufficiently smooth, we may exchange the order of integration with respect to $dP(\omega)$ and taking derivatives with respect to the spatial variables, yielding

$$-\Delta\mathbb{E}[u] = \mathbb{E}[f] \quad \text{in } D, \tag{20}$$
$$\mathbb{E}[u] = g \quad \text{on } \partial D, \tag{21}$$

which allows us to compute the expectation of the solution of the original linear problem by solving the deterministic problem for the expectation above.

In order to numerically solve the deterministic elliptic problems for fixed ω in (19), we use the open-source Julia programming language [10]. For testing purposes, we consider the right-hand side

$$f(x, y, \omega) := \frac{1}{n}\sum_{i=1}^{n} \exp\left(-U_i(\omega)\left(x^2 + y^2\right)\right),$$

where $U_i \sim U(0, 1)$, i.e., we choose $(U_i)_{i=1}^n$ to be an independent and identically distributed sequence of uniformly distributed random variables. We choose $g := 1$ such that the expectation of (19) satisfies the equation

$$-\Delta\mathbb{E}[u] = \frac{1 - \exp\left(-\left(x^2 + y^2\right)\right)}{x^2 + y^2} \quad \text{in } D, \tag{22}$$
$$\mathbb{E}[u] = 1 \quad \text{on } \partial D. \tag{23}$$

In this numerical example, we set $n := 15$. Figure 10 shows the absolute error of the exact expectation obtained by solving (20) compared to the approximation obtained by using a multisymmetric cubature formula. The accuracy of the approximation of the expectation increases with the degree d of the cubature formula, where $d \in \{3, 5, 7, 9, 11\}$. For $d = 11$, the full accuracy of the floating-point numbers is reached and one can observe the numerical error of the finite-element solver.

Figure 11 shows the absolute error of the exact expectation compared to the approximation obtained by using the Sobol sequence, a quasi-Monte Carlo method. Since the integrand is highly regular, the error converges much more slowly for the quasi-Monte Carlo method compared to the multisymmetric cubature formula. For $d = 3$, a total amount of four evaluations is needed, resulting in an L^2-error of $5 \cdot 10^{-5}$, whereas the L^2-error for the quasi-Monte Carlo method with 10^2 samples is $8.7 \cdot 10^{-4}$. By taking 10^3 samples, the L^2-error improves by two orders of magnitude to $6.9 \cdot 10^{-6}$, whereas the multisymmetric cubature formula for $d = 11$ requires only 48 solver calls to reach an L^2-error of $5 \cdot 10^{-15}$, i.e., the computational accuracy.

7. Conclusions.

By making use of a priori knowledge of the integrand, we have developed a general setting for creating cubature formulas for the broad class of G-invariant functions in Definition 2. These cubature formulas are of immediate importance for the numerical approximation of solutions of stochastic partial differential equations. Theoretical results for spaces

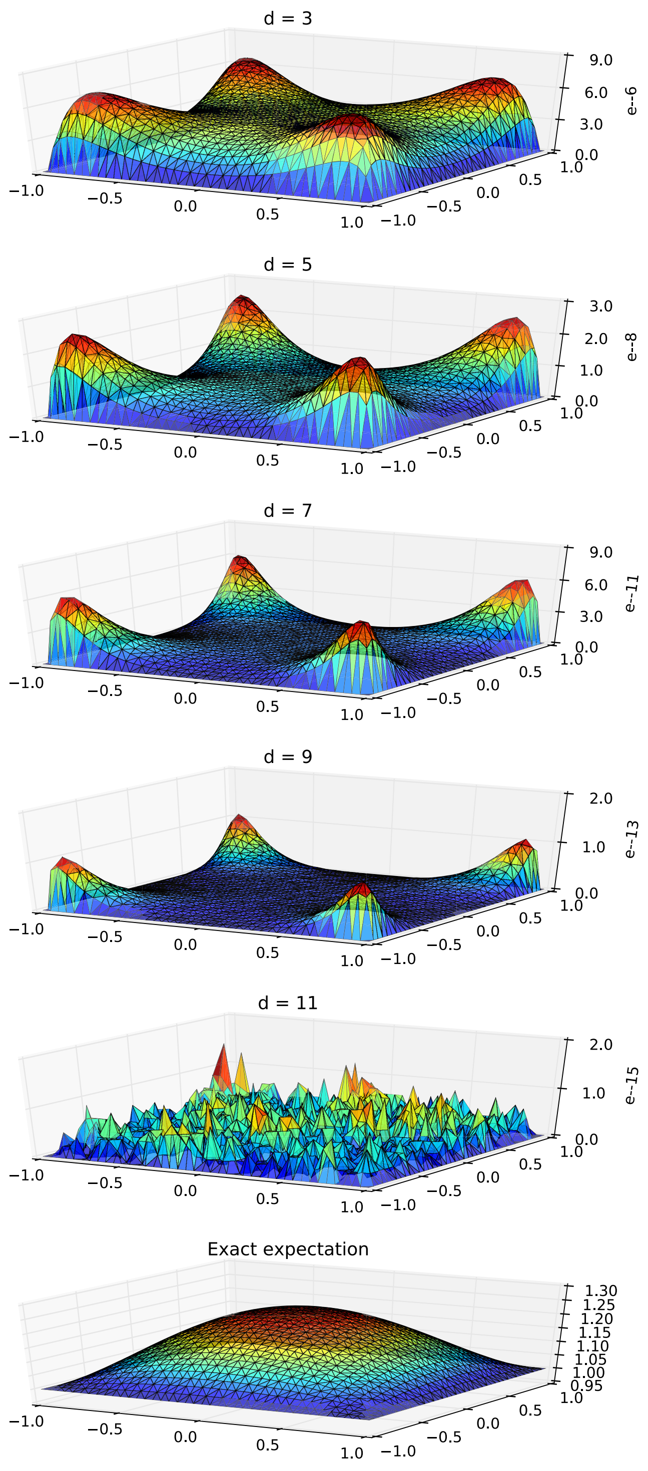

Figure 10. *The first plots show the absolute error of the exact expectation obtained by solving* (20) *compared to the approximation obtained by using a multisymmetric cubature rule of degree d. The exact expectation is shown in the last plot.*

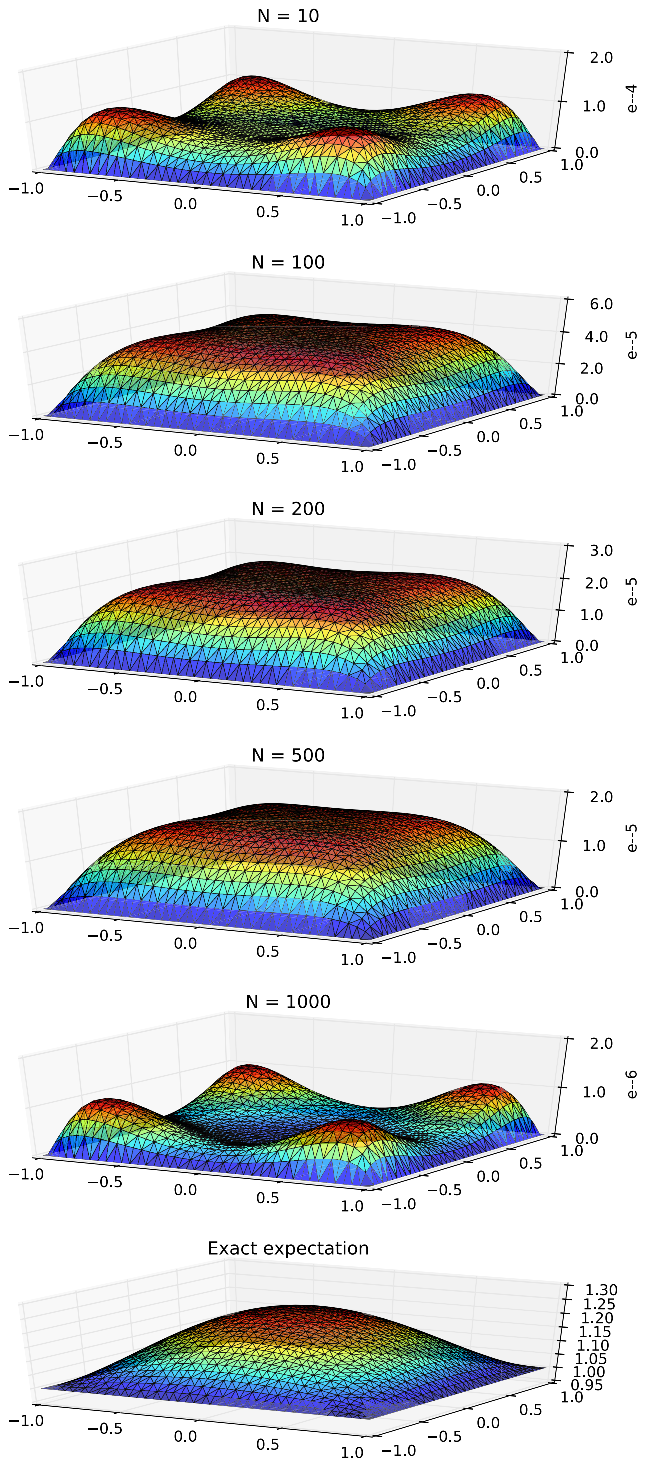

Figure 11. *The first plots show the absolute error of the exact expectation obtained by solving* (20) *compared to the approximation obtained by using a quasi-Monte Carlo method with N samples. The exact expectation is shown in the last plot.*

of G-invariant functions were shown in section 2 as well as standard error bounds in section 3. In the following, a general scheme for computing cubature formulas of G-invariant functions was developed in section 4. Based on that, a special kind of G-invariance, the notion of multisymmetry (see Definition 1) and the corresponding polynomial spaces were further examined in section 5. Finally, the algorithms were implemented and numerical results were shown in section 6, comparing the obtained multisymmetric cubature formulas to other, conventional multivariate integration techniques such as tensor-product Gauss–Legendre quadrature, quasi-Monte Carlo (the Sobol sequence), and Clenshaw–Curtis sparse grid. In the last part of section 6, the expectation of a stochastic elliptic partial differential equation was computed by using the proposed cubature formula and a quasi-Monte Carlo method.

The numerical results show that this newly developed integration method can prevail even against the computationally expensive tensor-product rule in terms of relative error to a certain extent. In both cases examined, namely, the fully symmetric and the multisymmetric one for $m = 2$, it was found that the proposed multisymmetric cubature formulas require far fewer evaluations for comparable accuracy than common methods such as quasi-Monte Carlo and Clenshaw–Curtis sparse grid. The effectiveness of our approach seems to increase with the regularity of the integrand. The results for the stochastic partial differential equation reinforce our conviction that the proposed formulas perform well for smooth integrands, beating the accuracy of the quasi-Monte Carlo method by orders of magnitude, again with far fewer function evaluations. This result suggests that multisymmetric cubature formulas can successfully be applied to more complex high-dimensional problems, e.g., in a stochastic discrete projection method or for the computation of the posterior density function in Bayesian parameter estimation.

In particular, we want to point out that the required number of evaluations scales very well with the number of dimensions for multisymmetric cubature formulas, being constant for fixed m, d, and $n \geqslant d$. This can be interpreted as actually overcoming the curse of dimensionality in the case of multisymmetry. Following the scheme of section 4, it may be possible to develop efficient algorithms for a multitude of groups G to compute dimensionally well-scaling cubature formulas. A logical next step might be to consider Cartesian products of multisymmetry groups, representing the case where there are several types of particles that are not mutually interchangeable.

Nonetheless, we encountered a limitation in the multisymmetric case. From a numerical perspective, the system to be solved becomes numerically unstable and thus the formulas obtained may lose precision as the dimension n increases.

Finally, we want to mention that natural applications of this low-cost integration method arise, e.g., in computational physics and, in particular, in computational quantum physics as well as in uncertainty quantification, when function evaluations are computationally expensive such as when solving stochastic partial differential equations. Whenever one has multisymmetric, smooth integrands and efficiency is a priority, the formulas presented here seem to be the integration technique of choice.

Acknowledgments. We would like to thank N. Levenberg, R. Schürer, and D. Toneian for helpful discussions. The computational results presented have been achieved in part using the Vienna Scientific Cluster (VSC).

REFERENCES

[1] T. Bagby, L. Bos, and N. Levenberg, *Multivariate simultaneous approximation*, Constr. Approx., 18 (2002), pp. 569–577, https://doi.org/10.1007/s00365-001-0024-6.

[2] I. Bogaert, *Iteration-free computation of Gauss–Legendre quadrature nodes and weights*, SIAM J. Sci. Comput., 36 (2014), pp. A1008–A1026, https://doi.org/10.1137/140954969.

[3] P. Bratley and B. L. Fox, *Algorithm 659: Implementing Sobol's quasirandom sequence generator*, ACM Trans. Math. Software, 14 (1988), pp. 88–100.

[4] E. Briand, *Polynômes Multisymétriques*, thesis, Université Rennes, Rennes, France, 2002, https://tel.archives-ouvertes.fr/tel-00002085.

[5] R. Cools, *An encyclopaedia of cubature formulas*, J. Complexity, 19 (2003), pp. 445–453.

[6] P. J. Davis and P. Rabinowitz, *Methods of Numerical Integration*, Dover, Mineola, NY, corrected 2nd ed., 2007.

[7] E. A. Devuyst and P. V. Preckel, *Gaussian cubature: a practitioner's guide*, Math. Comput. Model., 45 (2007), pp. 787–794, https://doi.org/10.1016/j.mcm.2006.07.021.

[8] J. Dick, *Walsh spaces containing smooth functions and quasi–Monte Carlo rules of arbitrary high order*, SIAM J. Numer. Anal., 46 (2008), pp. 1519–1553, https://doi.org/10.1137/060666639.

[9] J. Dick, F. Y. Kuo, and I. H. Sloan, *High-dimensional integration: The quasi-Monte Carlo way*, Acta Numer., 22 (2013), pp. 133–288, https://doi.org/10.1017/S0962492913000044.

[10] C. Geiersbach, C. Heitzinger, G. Pammer, S. Rigger, and G. Tulzer, *A 2D Finite Element Method Solver for Drift-Diffusion-Poisson Systems and Semilinear Poisson Equations*, https://github.com/Stivanification/DriftDiffusionPoissonSystems.jl (2016).

[11] A. Genz, *A package for testing multiple integration subroutines*, in Numerical Integration, Dordrecht, the Netherlands, 1987, pp. 337–340.

[12] T. Gerstner and M. Griebel, *Dimension-adaptive tensor-product quadrature*, Computing, 71 (2003), pp. 65–87.

[13] A. Hinrichs, E. Novak, M. Ullrich, and H. Woźniakowski, *The curse of dimensionality for numerical integration of smooth functions*, Math. Comput., 83 (2014), pp. 2853–2863, https://doi.org/10.1090/S0025-5718-2014-02855-X.

[14] A. Hinrichs, E. Novak, M. Ullrich, and H. Woźniakowski, *Product rules are optimal for numerical integration in classical smoothness spaces*, J. Complexity, 38 (2017), pp. 39–49.

[15] S. Joe and F. Y. Kuo, *Remark on algorithm 659: Implementing Sobol's quasirandom sequence generator*, ACM Trans. Math. Software, 29 (2003), pp. 49–57.

[16] D. Nuyens, G. Suryanarayana, and M. Weimar, *Rank-*1 *lattice rules for multivariate integration in spaces of permutation-invariant functions. Error bounds and tractability*, Adv. Comput. Math., 42 (2016), pp. 55–84, https://dx.doi.org/10.1007/s10444-015-9411-6.

[17] D. Nuyens, G. Suryanarayana, and M. Weimar, *Construction of quasi-Monte Carlo rules for multivariate integration in spaces of permutation-invariant functions*, Constr. Approx., 45 (2017), pp. 311–344, https://doi.org/10.1007/s00365-016-9362-2.

[18] A. B. Owen, *Latin supercube sampling for very high-dimensional simulations*, ACM Trans. Model. Comput. Simul., 8 (1998), pp. 71–102, https://doi.org/10.1145/272991.273010.

[19] G. Pammer and S. Rigger, *Multisymmetry*, https://github.com/Stivanification/DriftDiffusionPoissonSystems.jl (2017).

[20] D. Rydh, *A minimal set of generators for the ring of multisymmetric functions*, Ann. Inst. Fourier (Grenoble), 57 (2007), pp. 1741–1769, https://eudml.org/doc/10276.

[21] L. Schläfli, *Gesammelte Mathematische Abhandlungen*, Springer Basel, 1953, pp. 9–112.

[22] R. Schürer, *A comparison between (quasi-)Monte Carlo and cubature rule based methods for solving high-dimensional integration problems*, Math. Comput. Simul., 62 (2003), pp. 509–517.

[23] A. Stroud, *Approximate calculation of multiple integrals.* Prentice-Hall Ser. Automat. Comput., Prentice-Hall, Englewood Cliffs, NJ, 1971.

[24] F. Vaccarino, *The ring of multisymmetric functions*, Ann. Inst. Fourier (Grenoble), 55 (2005), pp. 717–731, http://eudml.org/doc/116205.

[25] M. Weimar, *The complexity of linear tensor product problems in (anti)symmetric Hilbert spaces*, J. Approx. Theory, 164 (2012), pp. 1345–1368, https://doi.org/10.1016/j.jat.2012.05.016.

[26] M. Weimar, *On lower bounds for integration of multivariate permutation-invariant functions*, J. Complexity, 30 (2014), pp. 87–97, https://doi.org/10.1016/j.jco.2013.10.003.