TU WIEN

MASTER THESIS

# High-dimensional Integration: Cubature Formulas for Multisymmetric Functions

*Author:*
Stefan RIGGER

*Supervisor:*
Assoz. Prof. Dipl. Ing. Dr.
Clemens HEITZINGER

August 31, 2017

TECHNISCHE
UNIVERSITÄT
WIEN
Vienna University of Technology

# Abstract

This thesis is focused on the problem of numerical multivariate integration in high dimensions. In particular, we introduce the idea of interpolatory cubature formulas for multsymmetric functions. The numerical calculation of integrands with a large number of arguments is a challenging problem that arises frequently in Uncertainty Quantification, e.g. in Bayesian Estimation or the solution of stochastic partial differential equations. While high-dimensional integration is very difficult in general, there are certain types of integrand families that are more tractable than others, such as integrands satisfying permutation-invariance properties related to multisymmetry. The goal of this thesis is to provide a short overview over the most popular techniques in the field of high-dimensional numerical integration and then explain the idea of cubature formulas for multisymmetric functions in more detail, an idea that was developed in joint work with my supervisor C. Heitzinger and my colleague G. Pammer. This work closely follows a paper on this topic that is currently under review.

# Zusammenfassung

Die vorliegende Arbeit beschäftigt sich mit der Problemstellung der numerischen Integration multivariater Funktionen in hohen Dimensionen. Insbesondere soll eine neue Methode diskutiert werden, nämlich die der Kubaturformeln für multisymmetrische Funktionen. Die numerische Berechnung von Integranden, die eine große Zahl an Argumenten aufweisen, ist eine schwierige Herausforderung, die in natürlicher Weise im Gebiet "Uncertainty Quantification" auftritt, zum Beispiel beim (numerischen) Lösen von stochastischen partiellen Differentialgleichungen. Hochdimensionale Integration ist im Allgemeinen schwierig, jedoch gibt es bestimmte Klassen von Integranden für die sich das Problem der hochdimensionalen Integration als leichter herausstellt. Wir interessieren uns daher für Klassen von Funktionen, die bestimmte Permutationsinvarianzeigenschaften erfüllen wie etwa multisymmetrische Funktionen. Ziel dieser Arbeit ist zunächst einen kurzen Überblick über die beliebtesten Techniken zu geben, die bei hochdimensionaler Integration zum Einsatz kommen, und anschließend näher auf Kubaturformeln für multisymmetrische Funktionen einzugehen. Die Idee, Kubaturformeln für solche Familien von Funktionen zu betrachten wurde gemeinsam mit meinem Betreuer C. Heitzinger und meinem Kollegen G. Pammer entwickelt. Eine Publikation zu diesem Thema, die inhaltlich in weiten Teilen mit der vorliegenden Arbeit übereinstimmt, wird derzeit rezensiert.

# Acknowledgements

I would like to thank my supervisor, Prof. Clemens Heitzinger, for his continuous encouragement and for providing me with the opportunity to work on this topic.

I would also like to thank my friends and colleagues at TU Wien for all the interesting and entertaining conversations we had together. Without them the experience of studying mathematics would not have been as pleasant and as fascinating as for me as it proved to be. In particular, I want to thank my colleague and friend Gudmund Pammer for the countless hours we spent together developing our idea of cubature formulas for multisymmetric functions, the core idea behind this work.

Finally, I would like to thank my parents for their unwavering and unconditional support throughout these years.

# Danksagung

Zuerst möchte ich mich bei meinem Betreuer, Prof. Clemens Heitzinger, für die Gelegenheit an diesem Thema zu arbeiten sowie für seine stets ermutigenden Anmerkungen bedanken.

Ich möchte mich außerdem bei meinen Kollegen und Freunden an der TU Wien für all die interessanten und unterhaltsamen Gespräche der letzten Jahre bedanken, durch die mein Studium zu einer angenehmen und faszinierenden Erfahrung geworden ist. Insbesondere möchte ich mich bei meinem Kollegen und Freund Gudmund Pammer für die unzähligen Stunden bedanken, die wir zusammmen damit verbracht haben, an unserer Idee der Kubaturformeln für multisymmetrische Funktionen zu arbeiten, die Grundidee hinter dieser Arbeit.

Schließlich möchte ich mich bei meinen Eltern für ihre bedingungslose Unterstützung bedanken.

# Contents

# CHAPTER 1

# Introduction

The ideas discussed in this work were motivated by problems arising in the field of Uncertainty Quantification, more precisely the numerical solution of stochastic partial differential equations. In general, the numerical solution of stochastic partial differential equations is 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 $\epsilon$ for all functions of a class (i.e., in the worst case) grows exponentially in dimension $N$ (i.e., the number of arguments) [13]. 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 equations are to be solved. Recently, the idea to exploit permutation-invariance conditions as another kind of a-priori knowledge has been enunciated [26, 27], 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 [17]. We propose to develop interpolatory cubature formulas for permutation-invariance conditions related to *multisymmetry groups,* motivated by the following example.

Let $u(x_1, \ldots, x_N)$ be a real-valued function of $N = nm$ variables defined on $\mathbb{R}^N$ that represents a physical attribute of a multi-particle 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 $(x_1, \ldots, x_N)$, 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 $u(x_1, \ldots, x_N) = u(\mathbf{x}_1, \ldots, \mathbf{x}_n) = u(\mathbf{x}_{\pi(1)}, .., \mathbf{x}_{\pi(n)}))$ holds for any transposition $\pi \in S_n$. Examples are $u(x_1, \ldots, x_N)$ 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 $i$-th 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 1.1.** Let $m$ and $n$ be natural numbers and $N := mn$. Let $a, b, \ldots, z$ be an alphabet of $n$ letters. We organize the arguments of a function $u\colon [0,1]^N \to \mathbb{R}$ 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 $\mathbb{R}^N$ by $S_{n,m}$.

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

**Definition 1.2.** Let $(G, \circ)$ be a subgroup of $(S_N, \circ)$. A function $u\colon [0,1]^N \to \mathbb{R}$ is called $G$*-invariant* if

$$u(\sigma(x_1, \ldots, x_N)) = u(x_1, \ldots, x_N) \quad \forall \sigma \in G \quad \forall (x_1, \ldots, x_N) \in [0,1]^N. \tag{1.1}$$

## Outline of Thesis

In the second chapter, we give a brief summary of the most popular techniques used for high-dimensional integration problems. The third chapter contains some simple results about analytical properties of spaces of $G$-invariant functions as well as error bounds for cubature formulas for multisymmetric functions. In the fourth chapter, we develop a numerical strategy to compute cubature formulas for multisymmetric functions. In the fifth chapter, we discuss some elementary results about bases and generating systems of spaces of multisymmetric polynomials. Numerical results are presented in the sixth chapter, comparing the proposed formulas to other integration techniques such as product rules, quasi-Monte Carlo rules and sparse grids. Finally, we draw some conclusions in the seventh chapter.

CHAPTER 2

# Numerical Integration Methods

Of the many challenges posed by multivariate numerical integration, the most difficult one seems to be overcoming the so-called curse of dimensionality, which means that the worst-case difficulty of numerical multivariate integration increases with the number of integration variables. Another challenge is the variety of possible geometries, because often a strategy that works for one geometry might not work for another, e.g. a strategy that works for hyperrectangles does not necessarily work for hyperspheres. We will confine ourselves to the case of the unit hypercube in this work (from which the case of general hyperrectangles can be deduced easily) and focus on the difficulties introduced by considering large dimensions. Also, we only consider somewhat smooth functions.

## Formulation of the Problem

We wish to approximate the integral

$$I_N(f) = \int_{[0,1]^N} f(\boldsymbol{x})\ \mathrm{d}\boldsymbol{x} := \int_0^1 \cdots \int_0^1 f(x_1, \ldots, x_N)\ \mathrm{d}x_1 \ldots \mathrm{d}x_N, \tag{2.1}$$

for some continuous function $f$, where $N \geqslant 1$ is possibly large, by a $k$-point integration rule of the form

$$Q(f) = \sum_{i=1}^{k} \omega_i f(\boldsymbol{x}_i), \tag{2.2}$$

with *weights* $\omega_1, \ldots, \omega_k \in \mathbb{R}$ and $f$ is evaluated at the (usually prescribed) *nodes* $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k \in [0,1]^N$.

## Monte Carlo Methods

The *Standard Monte Carlo method (SMC) or Monte Carlo method (MC)* is an equal-weight cubature rule of the form

$$Q_k(f) = \frac{1}{k}\sum_{i=1}^{k} f(\boldsymbol{u}_i), \tag{2.3}$$

where $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k$ are i.i.d (independent and identically distributed) uniform *random variables* on $[0,1]^N$. The fact that the nodes are not prescribed, but sampled from a uniform distribution on $[0,1]^N$ distinguishes Standard Monte Carlo from other methods discussed in this work. In particular, this makes it possible to find probabilistic error bounds. The following result is well-known, see for example [9].

**Theorem 2.1** (Error bounds for Standard Monte Carlo). *For all $f \in L^2([0,1]^N, \mathbb{R})$, we have*

$$\sqrt{\mathbb{E}[|I_N(f) - Q_k(f)|^2]} = \frac{\sigma(f)}{\sqrt{k}} \tag{2.4}$$

*where the expectation is taken with respect to the joint distribution of $(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k)$, and*

$$\sigma^2(f) := I_N(f^2) - (I_N(f))^2 \tag{2.5}$$

*is the* variance *of $f$.*

It is easy to see that the random variable $Q_k(f)$ has mean

$$\mathbb{E}[Q_k(f)] = I_N(f)$$

and variance

$$\mathbb{V}[Q_k(f)] = \frac{\sigma^2(f)}{k}.$$

By the central limit theorem, if $f \in L^2([0,1]^N)$ and $f \neq 0$, then

$$\lim_{k\to\infty} \mathbb{P}\left(|I_N(f) - Q_k(f)| \leqslant c\frac{\sigma(f)}{\sqrt{k}}\right) = \Phi(c) - \Phi(-c).$$

where $\Phi$ is the distribution function of the standard normal distribution. This means that we have a probabilistic error bound with a convergence rate of $O(k^{-1/2})$ *independently of the dimension $N$*. Even though the independence of the dimension is a pleasing property, the convergence rate of $O(k^{-1/2})$ can be very slow in practice, especially when dealing with very smooth functions. This motivates the use of *quasi-Monte Carlo methods.*

## Quasi-Monte Carlo

*Quasi-Monte Carlo (QMC) methods* are equal-weight cubature rules of the form

$$Q_k(f) := \frac{1}{k}\sum_{i=1}^{k} f(\boldsymbol{x}_i) \tag{2.6}$$

where the nodes $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k \in [0,1]^N$ are chosen *deterministically.* By doing this, one hopes to find determinstic error bounds and to improve the MC convergence rate of $O(k^{-1/2})$ for smooth functions. See [9] for a survey on quasi-Monte Carlo methods.

**Definition 2.2** (radical inverse function)**.** For integers $i \geqslant 0$ and $b \geqslant 2$, we define the *radical inverse function* $\phi_b(i)$ as follows.

$$\text{If} \quad i = \sum_{a=1}^{\infty} i_a b^{a-1}, \quad \text{where} \quad i_a \in \{0,1,\ldots,b-1\}, \quad \text{then } \phi_b(i) := \sum_{a=1}^{\infty} \frac{i_a}{b^a}.$$

In other words, if $i = (\cdots, i_2 i_1)_b$ denotes the base $b$ representation of $i$, then $\phi_b(i) := (0.i_1 i_2 \cdots)_b$.

**Example 2.3** (van der Corput sequence)**.** The *van der Corput sequence in base b* is the one-dimensional sequence

$$\phi_b(0), \phi_b(1), \phi_b(2), \ldots$$

For example, take $b = 2$. We write down the natural numbers $0, 1, 2, \ldots$ in base 2

$$0, 1_2, 10_2, 11_2, 100_2, 101_2, 110_2, \ldots$$

Applying $\phi_2$ to each number yields

$$0, 0.1_2, 0.01_2, 0.11_2, 0.001_2, 0.101_2, 0.011_2, \ldots$$

which in decimal form is the sequence

$$0, 0.5, 0.25, 0.75, 0.125, 0.625, 0.375, \ldots$$

**Example 2.4** (Halton sequence)**.** Let $p_1, \ldots, p_N$ be the first $N$ prime numbers. The *Halton sequence* $\boldsymbol{x}_1, \boldsymbol{x_2}, \ldots$ *in* $N$ *dimensions* is given by

$$x_i = (\phi_{p_1}(i), \phi_{p_2}(i), \ldots, \phi_{p_N}(i)), \quad i = 0, 1, \ldots$$

that is, the $j$th components of nodes in the Halton sequence form the van der Corput sequence in base $p_j$, where $p_j$ is the $j$th prime. Explicitly, we have

$$\begin{aligned}
\boldsymbol{x}_1 &= (0, 0, 0, \ldots, 0),\\
\boldsymbol{x}_2 &= (0.1_2, 0.1_3, 0.1_5, \ldots, 0.1_{p_N})\\
\boldsymbol{x}_3 &= (0.01_2, 0.2_3, 0.2_5, \ldots, 0.2_{p_N}),\\
&\ \vdots
\end{aligned}$$

**Definition 2.5** (star discrepancy)**.** Let $P = \{\mathbf{x}_1, \dots, \mathbf{x}_k\}$ be a point set. We define the star discrepancy $D_N^*(P)$ as

$$D_k^*(P) = \sup_{y_i \in [0,1]} \left| \prod_{i=1}^{s} y_i - \frac{\#\left\{P \cap \times_{i=1}^{s}[0, y_i]\right\}}{N} \right|.$$

The star discrepancy of a point set $P$ can be seen as a measure of how much $P$ deviates from a uniformly distributed point set.

**Theorem 2.6** (Koksma-Hlawka inequality)**.** *If $f$ has bounded variation on $[0,1]^N$ in the sense of Hardy and Krause, then for any point set $P = \{\mathbf{x}_1, \dots, \mathbf{x}_k\}$ the inequality*

$$\left| \frac{1}{k} \sum_{i=1}^{k} f(\mathbf{x}_i) - \int_{[0,1]^N} f(\mathbf{u}) \, \mathrm{d}\mathbf{u} \right| \leqslant V(f) D_k^*(P) \tag{2.7}$$

*holds, where $V(f)$ designates the variation in the sense of Hardy and Krause.*

The definition of variation in the sense of Hardy and Krause is given in [15]. The Koksma-Hlawka inequality is sharp in the sense that for any $\mathbf{x}_1, \dots, \mathbf{x}_k \in [0,1]^N$ and $\epsilon > 0$, one can find $f \in C^\infty([0,1]^N)$ with $V(f) = 1$ such that

$$\left| \frac{1}{k} \sum_{i=1}^{k} f(\mathbf{x}_i) - \int_{[0,1]^N} f(\mathbf{u}) \, \mathrm{d}\mathbf{u} \right| > D_k^*(\{\mathbf{x}_1, \dots \mathbf{x}_k\}) - \epsilon$$

For proofs of the above statements see Chapter 2, Theorem 2.11 and Theorem 2.12 in [15]. The Koksma-Hlawka inequality implies error bounds for various kinds of sequences. Van der Corput and Halton sequences are known to have discrepancies satisfying $D(\{\mathbf{x}_1, \dots, \mathbf{x}_k\}) = O(\log(k)^N/k)$, (see [9]) which by the Koksma-Hlawka inequality implies convergence rates of order $\log(k)^N/k$ for functions of bounded variation, which beats the Monte Carlo convergence rate of $O(k^{1/2})$ asymptotically.

## Sparse Grids

Let $f : [0,1] \to \mathbb{R}$ be a continuous function and $(m_k)_{k \in \mathbb{N}}$ be a non-decreasing sequence of natural numbers. Let

$$Q_{m_k}(f) := \sum_{i=1}^{m_k} w_{ik} f(x_{ik}) \tag{2.8}$$

denote a sequence of quadrature rules with positive weights $w_{ik} \in (0, \infty)$ which satisfies

$$\lim_{k \to \infty} Q_{m_k}(f) = \int_0^1 f(x) \, \mathrm{d}x, \quad f \in C[0,1]. \tag{2.9}$$

Define $Q_{m_0} := 0$ and

$$\Delta_k := Q_{m_k} - Q_{m_{k-1}}, \quad k \geqslant 1.$$

For a multiindex $(k_1, \ldots, k_N) = \mathbf{k} \in \mathbb{N}^N$, let

$$\Delta_{\mathbf{k}} = \Delta_{k_1} \otimes \cdots \otimes \Delta_{k_N}.$$

Now, if $f : [0,1]^N \to \mathbb{R}$ is a multivariate continuous function, we find

$$\int_{[0,1]^N} f(\mathbf{x})\ \mathrm{d}\mathbf{x} = \sum_{\mathbf{k}\in\mathbb{N}^N} \Delta_{\mathbf{k}}(f).$$

For a given $l \in \mathbb{N}$, the *classical sparse grid method*, sometimes called Smolyak method, is then defined by

$$Q_l^{(N)}(f) := \sum_{|\mathbf{k}|\leqslant l+N-1} \Delta_{\mathbf{k}}(f) \tag{2.10}$$

where $|\mathbf{k}| := \sum_{i=1}^N k_i$. Note that the tensor product rule is recovered if one chooses $|\cdot|_\infty := \max(k_1, \ldots, k_N)$ instead of the $|\cdot|$-norm. It is also possible to choose the multiindices $\mathbf{k}$ appearing in (2.10) adaptively depending on the function $f$, see [12]. For a survey on sparse grids see [5]. The number of points in a sparse grid can be determined as

$$n_l^N = \sum_{|k|\leqslant l+N-1} m_{k_1} \cdots m_{k_N}. \tag{2.11}$$

If $n_l^1 = O(2^l)$ the order of $n_l^N$ is (see [16])

$$n_l^N = O(2^l l^{N-1}).$$

Define the space of functions with bounded mixed derivatives or order $r$, that is,

$$\mathcal{W}_N^r := \left\{ f : [0,1]^N \to \mathbb{R}, \left\| \frac{\partial^{|\mathbf{s}|} f}{\partial^{s_1} x_1 \cdots \partial^{s_N} x_N} \right\| < \infty,\ s_i \leqslant r \right\}$$

Now assume that the one-dimensional quadrature formulas satisfy the error bound

$$\left| Q_{m_l}(f) - \int_0^1 f(x)\ \mathrm{d}x \right| = O((n_l^1)^{-r}), \quad f \in C([0,1]^N)^r.$$

Which holds, for example, for interpolatory quadrature formulas with positive weights, such as the Clenshaw-Curtis, Gauss-Patterson and Gauss-Legendre formulas. Taking one of these rules as a one-dimensional basis, if $f \in \mathcal{W}_N^r$ and $n_l^1 = O(2^l)$, we have an error bound of order (see [25])

$$\left| Q_l^{(N)}(f) - \int_{[0,1]^N} f(\mathbf{x})\ \mathrm{d}\mathbf{x} \right| = O(2^{-lr} l^{(N-1)(r+1)}). \tag{2.12}$$

CHAPTER 3

# Spaces of $G$-invariant Functions

## Continuous and $p$-integrable $G$-invariant functions

Here and in the following chapters, let $G$ denote a (fixed) subgroup of $S_N$, the symmetric group in $N$ letters. We denote the $N$-dimensional unit cube by $I := [0,1]^N$. If $X$ is a vector space of real-valued functions defined on a subset of $\mathbb{R}^N$, 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 $[0,1]^N$ 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 3.1 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 [26, 27, 17], although not for the precise setting we are interested in.

**Proposition 3.1.** *Let $X := (C(I), \|.\|_\infty)$ or $X := (L^p(I), \|.\|_p)$ for $1 \leqslant p \leqslant \infty$. Then the linear averaging operator*

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

*satisfies*

$$S(1) = 1, \tag{3.2a}$$
$$S(S(f)g) = S(f)\cdot S(g) \quad \forall\forall f,g \in X. \tag{3.2b}$$

*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 $\sigma \in S_N$ we have $\lambda(M) = \lambda(\sigma^{-1}(M))$. We now prove formula (3.2). 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 (3.2) 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 3.2** ($G$-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 3.1. 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 3.3** ($L^p$-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 3.2, making use of the fact that $\mathcal{P}(I)$ is dense in $L^p(I)$. □

## Taylor Expansion of $G$-invariant functions

**Theorem 3.4.** *The Taylor polynomials of a (sufficiently smooth) G-invariant function $f : [0,1]^N \to \mathbb{R}$ 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 $\mathbf{a} = (a_1, \ldots, a_N)$ has the form

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

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

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

Thus,

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

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

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

inductively on the order of the multiindex $\alpha$: 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 $\mathbf{e}_i \in \mathbb{R}^N$ 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 3.5.** Without the assumption that the center of the Taylor polynomials be $G$-invariant, Theorem 3.4 does not hold in general: Consider for example the first-order Taylor polynomial of $f(x,y) := xy$ centered at $\mathbf{a} = (1,0)$.

## Error Bounds

**Definition 3.6** (cubature formula and error functional)**.** A $k$-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

$N$-cube or $N$-sphere have been developed in the past, see [23, 6] for compilations. In the monograph [7, p. 376], it is stated that "[t]here appears to be no systematic theoretical approach to monomial rules nor any systematic evaluation of their practical effectiveness. Two properties of approximate integration rules are considered particularly desirable: the abscissas should lie in the region and the weights should be positive." Furthermore [7, p. 378], "[f]inally almost all work on error estimation has been confined to the two-dimensional case." 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 3.7.** *Let $K$ be a connected compact subset of $\mathbb{R}^N$ 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 $\sigma$ 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 $\mathbb{R}^N$ with the following property: for each multiindex $\alpha$ with $|\alpha| \leqslant \min(m,n)$, we have*

$$\|D^\alpha(f-p_n)\|_\infty \leqslant \frac{C}{n^{m-|\alpha|}} \sum_{|\gamma|\leqslant m} \|D^\gamma f\|_\infty, \tag{3.3}$$

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

*Proof.* See Theorem 2 in [1]. □

The following error bounds are a simple consequence of the properties of the symmetrization operator $S$ and the above Jackson Theorem. We remark that the same reasoning given in line (3.5) shows that for $X = L^p(I)$ or $X = C(I)$ and $f \in X^G$, one has

$$\operatorname{dist}(\mathcal{P}_{\leqslant d}, f) = \operatorname{dist}(\mathcal{P}^G_{\leqslant d}, f) \tag{3.4}$$

**Theorem 3.8** (Error bounds)**.** *Let $f\colon I \to \mathbb{R}$ be a $G$-invariant function of class $C^m$. Let $Q$ be a $k$-point cubature formula with positive weights $\omega_i$ 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 $N$ and $m$ such that*

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

*holds.*

*Proof.* Let $f \in C(I)^G$. Consider again the linear operator $S$ introduced in Proposition 3.1. Let $p_n$ denote a polynomial approximation of $f$ satisfying the error bound in Theorem 3.7. Setting $s_n := S(p_n)$, we obtain

$$\|s_n - f\|_\infty = \|S(p_n - f)\|_\infty \leqslant \|S\| \cdot \|p_n - f\|_\infty \leqslant \frac{C}{n^m} \sum_{|\gamma| \leqslant m} \|D^\gamma f\|_\infty. \tag{3.5}$$

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 3.9.** Setting $G := \{\mathrm{id}\}$ in Theorem 3.7, one obtains an error bound for the classical case. Thus, the error bounds formulated above can be viewed as an answer to an open question stated in [8].

CHAPTER 4

# Cubature Formulas for $G$-invariant Functions

In this chapter, an approach for computing cubature formulas with positive weights for $G$-invariant functions on $[0,1]^N$ is proposed. Let $B := \{p_1, \dots, p_{n^*}\}$ be a basis of $\mathcal{P}^G_{\leqslant d}$ and define $n^* := \dim \mathcal{P}^G_{\leqslant d}$. Let $\mathcal{N} := \{\mathbf{x}_1, \dots, \mathbf{x}_k\}$ be a collection of points in $[0,1]^N$. 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 $\omega_1, \dots, \omega_k$ satisfy the system

$$\sum_{i=1}^{k} p_l(\mathbf{x}_i)\omega_i = \int_{[0,1]^N} p_l(\mathbf{x})d\mathbf{x} \quad \forall l \in \{1, \dots, n^*\}. \tag{4.1}$$

In the univariate case, system (4.1) has a unique solution for any choice of distinct $\mathbf{x}_1, ..., \mathbf{x}_d$. In the multivariate case, system (4.1) is not always solvable and if a solution exists, it is not unique in general. In our case, the number of nodes will typically exceed the number of basis polynomials by far.

## Basic Scheme

We suggest the following algorithm, which is a variation of the approach presented in [8].

1. Generate a basis $p_1, \dots, p_{n^*}$ of $\mathcal{P}^G_{\leqslant d}$.
2. Calculate the integrals $\int_I p_l(\mathbf{x})d\mathbf{x}$ for all $l \in \{1, \dots, n^*\}$.
3. Calculate the nodes of a univariate degree $d$ Gaussian Quadrature formula on $[0,1]$, denoted by $\mathcal{N}_0$. Let $C := \prod_{i=1}^{N} \mathcal{N}_0$. Consider the natural action of $G$ on $C$. Choose elements $\mathbf{x}_1, \dots, \mathbf{x}_k$ 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 $(A)_{ij} := (p_i(\mathbf{x}_j))$. Let $(\mathbf{b})_i := \int_I p_i(\mathbf{x}) d\mathbf{x}$. 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 $N$-dimensional tensor product of a univariate degree-$d$ Gaussian Quadrature Rule solves system (4.1) with $\mathcal{N} = C$. Let $\mathbf{y}_j$ denote the nodes and $w_j$ denote the weights of the tensor product formula for $j \in \{1, \ldots, (\frac{d+1}{2})^N\}$. 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 $\omega$ 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 0-th 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 algorithmical 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 $S_N$, they are not computationally efficient. We will present an optimized version of the basic scheme for the multisymmetric case in Chapter 5.

## Generating a Basis of $\mathcal{P}^G_{\leqslant d}$

Let $\mathcal{I}$ be the set of all multiindices $\alpha$ satisfying $|\alpha| \leqslant d$, i.e., $\mathcal{I} = \{(\alpha_1, \ldots, \alpha_N) \mid \alpha_i \in \mathbb{N} \cup \{0\},\ \sum_{i=1}^{N} \alpha_i \leqslant d\}$. 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 $\beta_i$ 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 4.1.** *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 3.1.*

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

$$D^{\beta}(\mathbf{x}^{\alpha}) = \alpha! \delta_{\alpha\beta} \quad \forall \beta \text{ with } |\beta| = |\alpha| \tag{4.2}$$

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

$$\sum_{i=1}^{n^*} c_i S(\mathbf{x}^{\beta_i}) = \frac{1}{|G|} \sum_{i=1}^{n^*} c_i \sum_{\sigma \in G} \mathbf{x}^{\sigma(\beta_i)} = 0. \tag{4.3}$$

Let $\beta_j$ be a multiindex of order $d$. 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

$$D^{\beta_j} \left( \sum_{i=1}^{n^*} c_i \sum_{\sigma \in G} \mathbf{x}^{\sigma(\beta_i)} \right) = c_j m \beta_j! = 0. \tag{4.4}$$

Repeating this reasoning inductively yields $c_1 = \cdots = c_{n^*} = 0$ 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 $S(\mathbf{x}^\alpha) = S(\mathbf{x}^\beta)$. Thus, the equality $\bigcup_{i=1}^{n^*} O_{\beta_i} = \mathcal{I}$ implies that

$$\operatorname{span}\{S(\mathbf{x}^\alpha) \mid \alpha \in \mathcal{I}\} = \operatorname{span}\{S(\mathbf{x}^{\beta_i}) \mid i \in \{1, \ldots, n^*\}\}, \tag{4.5}$$

which proves the claim. □

Using Lemma 4.1, a basis of $\mathcal{P}^G_{\leqslant d}$ can be generated as follows.

**Algorithm 1** Basis Generation

```
Set $\mathcal{I} := \{(\alpha_1, \ldots, \alpha_N) \mid \alpha_i \in \mathbb{N} \cup \{0\},\ \sum_{i=1}^N \alpha_i \leqslant d\}$
Set $B := \varnothing$
while $\mathcal{I} \neq \varnothing$ do
    Pick $\alpha \in \mathcal{I}$
    Update $B := B \cup \{S(\mathbf{x}^\alpha)\}$
    Update $\mathcal{I} := \mathcal{I} \setminus \{\sigma(\alpha) : \sigma \in G\}$
end while
return $B$
```

## Calculating Integrals of Basis Polynomials

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

$$\int_{[0,1]^N} S(\mathbf{x}^{\beta_i}) d\mathbf{x} = \int_{[0,1]^N} \mathbf{x}^{\beta_i} d\mathbf{x}. \tag{4.6}$$

## 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 $C := \prod_{i=1}^{N} \mathcal{N}_0$ modulo $G$ is given in Algorithm 2.

**Algorithm 2** Node Generation

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

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 chapter.

**CHAPTER 5**

# Multisymmetry

In a purely algebraic context, the notion of *multisymmetric functions* usually refers to elements of an abstract, category-theoretical construction that is not canonically related to real-valued functions. In this work, whenever we refer to *multisymmetric functions*, we consider real-valued functions defined on $[0,1]^N$ that satisfy permutation-invariance properties related to a multisymmetry group as defined in the introduction.

## Algebraical Theory

From now on, let $n$ and $m$ be positive integers, and $N := nm$. 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 5.1.** We are only interested in $\mathcal{P}_{\leqslant d}^{S_{m,n}}$ as an $\mathbb{R}$-vector space, therefore we will not strictly distinguish between polynomials in the algebraical sense and polynomial functions defined on $\mathbb{R}^n$.

**Definition 5.2.** A *vector partition* is a finite multiset of elements of $\mathbb{N}^m \backslash \{\mathbf{0}\}$.

Let $\boldsymbol{\alpha} = (\alpha^{(1)}, \ldots, \alpha^{(k)})$ be a vector of $k$ elements of $\mathbb{N}^m$. 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 $\gamma \in \mathbb{N}^m \backslash \{\mathbf{0}\}$, we call $\mathfrak{p}$ a *partition* of $\gamma$. Let $\Pi_m$ denote the set of vector partitions with parts in $\mathbb{N}^m \backslash \{\mathbf{0}\}$. For an integer vector $\gamma$, let $|\gamma|$ denote the sum of its components.

**Example 5.3.** 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 $a, b, \ldots, z$ 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 5.4.** Let $\mathfrak{p}$ be a vector partition with parts in $\mathbb{N}^m$ of length at most $n$. The *monomial multisymmetric function with index* $\mathfrak{p}$ is defined as

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

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

**Example 5.5.** Let $m := 2$, $n := 3$ and $\mathfrak{p} := [(1,0),(1,0),(1,1)]$. Then we have

$$\begin{aligned} m_{[(1,0),(1,0),(1,1)]} &= \boldsymbol{a}^{(1,0)}\boldsymbol{b}^{(1,0)}\boldsymbol{c}^{(1,1)} + \boldsymbol{a}^{(1,0)}\boldsymbol{b}^{(1,1)}\boldsymbol{c}^{(1,0)} + \boldsymbol{a}^{(1,1)}\boldsymbol{b}^{(1,0)}\boldsymbol{c}^{(1,0)} \\ &= a_1 b_1 c_1 c_2 + a_1 b_1 b_2 c_1 + a_1 a_2 b_1 c_1. \end{aligned}$$

**Theorem 5.6.** *The monomial multisymmetric functions $m_{\mathfrak{p}}$, where $\mathfrak{p}$ is a vector partition with parts in $\mathbb{N}^m$ of length at most $n$, together with the constant function* 1 *form a basis of $\mathcal{P}^{S_{m,n}}$.*

Similarly to Lemma 4.1, Theorem 5.6 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 $[\boldsymbol{\alpha}] = [\alpha^{(a)}, \alpha^{(b)}, \ldots, \alpha^{(z)}]$ with the multiindex $\boldsymbol{\alpha} \in \mathbb{N}^N$. Letting $S_{m,n}$ act naturally on $\mathbb{N}^N$, we see that the orbit of an element $\boldsymbol{\alpha} \in \mathbb{N}^N$ can be described by $\mathfrak{I}([\boldsymbol{\alpha}])$, i.e., all the sequences $\boldsymbol{\beta} = (\beta^{(a)}, \ldots, \beta^{(z)})$ 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 3.1. As $\mathfrak{p}$ runs through the vector partitions of length at most $n$ with parts in $\mathbb{N}^m \backslash \{\mathbf{0}\}$, $\mathfrak{I}(\mathfrak{p})$ runs through the orbits of $\mathbb{N}^N \backslash \{\mathbf{0}\}$ under $S_{m,n}$. Thus, the claim follows from Lemma 4.1. $\square$

As a corollary of Theorem 5.6, we find that

$$\dim \mathcal{P}(I)^{S_{m,n}}_{\leqslant d} = 1 + |\{\mathfrak{p} \in \Pi_m \mid |s(\mathfrak{p})| \leqslant d, \ell_{\mathfrak{p}} \leqslant n\}|. \tag{5.1}$$

Formula (5.1) 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 $\mathcal{P}^{S_{m,n}}_{\leqslant d}$ is constant in $n$ for $n \geqslant d$. Recall that dim $\mathcal{P}(I)^{S_{m,n}}_{\leqslant d}$ gives an upper bound on the amount of weights needed to integrate all polynomials of $\mathcal{P}(I)^{S_{m,n}}_{\leqslant d}$ 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 5.7.** Let $P \in \mathbb{R}[X_1, \dots, X_m]$ be of positive degree. We define the *elementary multisymmetric functions associated to $P$* over the following generating function:

$$\sum_{k=0}^{n} t^k e_k(P) := (1 + tP(\boldsymbol{a})) \cdot (1 + tP(\boldsymbol{b})) \cdots (1 + tP(\boldsymbol{z})). \tag{5.2}$$

It follows easily from this definition that

$$e_1(P) = P(\boldsymbol{a}) + P(\boldsymbol{b}) + \cdots + P(\boldsymbol{z}) \tag{5.3}$$

holds.

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

**Example 5.8.** 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 5.9.** *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 $\mu$ smaller or equal to $n$.*

*Proof.* See Theorem 1 of [24]. □

We introduce a multigrading with values in $\mathbb{N}^m$ on $\mathcal{P}^{S_{m,n}}$: For $x$ in the alphabet $a, b \dots, z$, we give the variable $x_i$ the multidegree $\xi_i$, where $\xi_i$ is the $i$-th 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 5.10.** 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 5.11.** *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 a $\mathbb{R}$-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 a $\mathbb{R}$-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

$$\operatorname{mdeg}\left(\prod_{i=1}^{k} e_1(\mu_i)\right) = \sum_{i=1}^{k} \operatorname{mdeg}(e_1(\mu_i)) = \sum_{i=1}^{k} \operatorname{mdeg}(\mu_i). \tag{5.4}$$

We establish the mapping

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

where we take all $\alpha_i$ in the definition to be nonzero. It is straightforward to see that $\varphi$ 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 a $\mathbb{R}$-vector space is only a rewording of Proposition 5.9, 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

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

which shows that $\mathcal{B}^{n,m}_{\leqslant d}$ is in fact a basis by formula 5.1. $\square$

**Remark 5.12.** 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 5.11.

**Definition 5.13.** 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 $i \in \mathbb{N}$. Define

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

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

**Corollary 5.14.** *Define the set $\mathcal{C}_{\leqslant d}^{n,m}$ as*

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

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

*Proof.* From the definition of $\mathcal{T}_m$, it is easy to see that $|\mathcal{C}_{\leqslant d}^{n,m}| = |\mathcal{B}_{\leqslant d}^{n,m}|$, so by Theorem 5.11 it is sufficient to show that $\mathcal{C}_{\leqslant d}^{n,m}$ is a generating system for $P(I)_{\leqslant d}^{S_{n,m}}$. Let $\mu_1, \ldots, \mu_{k*}$ $(P_1, \ldots, P_{k*})$ be an enumeration of all polynomials $\mu \in \mathcal{M}_m^+$ $(P \in \mathcal{T}_m^+)$ such that $|\operatorname{mdeg}(\mu)| \leqslant \min(n, d)$ $(|\operatorname{mdeg}(P)| \leqslant \min(n, d))$. We show that $\{e_1(P_1), \ldots, e_1(P_{k*})\}$ generates $\mathcal{P}_{\leqslant d}^{S_{m,n}}$ as an $\mathbb{R}$-algebra. Let $M \in \mathcal{P}(I)_{\leqslant d}^{S_{n,m}}$. By Theorem 5.11, there is a polynomial $Q \in \mathbb{R}[X_1, \ldots, X_{k*}]$ such that

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

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

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

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

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

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

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

which implies

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

which concludes the proof. □

## Implementation of the Basic Scheme

### Basis Generation

As indicated earlier, we will work with a "basis" of the form $\mathcal{C}^{n,m}_{\leqslant d}$ (as in Corollary 5.14) 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 $\mathcal{C}^{n,m}_{\leqslant d}$-polynomials to alleviate a flaw of the $\mathcal{B}^{n,m}_{\leqslant d}$-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 5.13): Let $y_j$ denote the nodes of the univariate degree-$d$ Gaussian quadrature formula on $[0, 1]$ for $j \in \{1, \dots, \frac{d+1}{2}\}$. Define

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

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.

### Calculating Integrals of Basis Polynomials

The usage of a "basis" like $\mathcal{C}^{n,m}_{\leqslant d}$ 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 5.15.** Let $l$ be a positive integer. Define the set $\mathfrak{P}(\{1, \dots, l\})$ to be the set of all (set-)partitions of $\{1, \dots, l\}$.

**Lemma 5.16.** *Let $P_i \in \mathcal{T}^+_m$ for all $i \in \{1, \dots, l\}$. Then the equation*

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

*holds.*

*Proof.* We calculate

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

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 $\{1, \ldots, l\}$ into at most $n$ parts, there are $n(n-1)\cdots(n-|\mathcal{D}|+1)$ terms associated to 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

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

□

Using Lemma 5.16, 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 $\{1, \ldots, l\}$, but as $l \leqslant d$ and $d$ typically is small, this is usually inexpensive.

### 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 $\prod_{i=1}^{m} \mathcal{N}_0$ with repetitions gives a full representative system of the nodes modulo $S_{m,n}$. Therefore, the number of orbits $k$ of $\prod_{i=1}^{N} \mathcal{N}_0$ under the natural action of $S_{m,n}$ is equal to

$$k = \binom{n + \left(\frac{d+1}{2}\right)^m - 1}{n}. \tag{5.5}$$

### 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 (5.5)), 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,

**Algorithm 3** 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
```

as $k \gg n^*$, we should be able to leave out a lot of columns and still obtain results. This motivates Algorithm 3.

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

# CHAPTER 6

# Numerical Results

All the formulas shown here as well as the code used to calculate them can be found in our GitHub repository [19].

We would like to preface this chapter 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 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 chapter, we compare the cubature formula for multisymmetric functions obtained with the method presented in Chapter 5 to other numerical integration methods such quasi-Monte-Carlo methods, e.g., the Sobol sequence [3, 14], sparse-grid methods, and tensor-product formulas.

In the implementation it became apparent that the numerically most costly parts of the algorithm proposed in Chapter 4 were finding a feasible solution to the possibly large system of linear equations (4.1), e.g., via linear programming and the computation of the integrals of the basis polynomials as described in Lemma 5.16. In Section 5.2.2, an efficient work-around for the former problem was proposed using that the system is greatly over-determined. 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 6.1. However, in our implementation, the system tends to become nummerically unstable as $n$ or $d$ grows large, e.g., for $n \geqslant 100$ resp. $d \geqslant 11$ and $m = 1$, since the value range of (4.1) becomes wider and wider.

| | 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 |

Table 6.1: 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 $d$.

## 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

$$
\begin{aligned}
g_1(x_1, y_1, \ldots, x_n, y_n) &:= \sum_i^n \left( \exp\left(\frac{x_i}{10}\right) + \exp(y_i) + \frac{1}{2}\sum_{j\neq i}^n \exp\left(\frac{x_i x_j}{10}\right) + \exp(y_i y_j) \right), \\
g_2(x_1, y_1, \ldots, x_n, y_n) &:= \sin\left(\sum_i^n \frac{x_i}{10} + y_i\right), \\
g_3(x_1, y_1, \ldots, x_n, y_n) &:= \exp\left(\sum_i^n -\frac{x_i^2}{10} - y_i^2\right), \\
g_4(x_1, y_1 \ldots, x_n, y_n) &:= \frac{1}{\sqrt{\sum_i^n \frac{x_i}{10} + y_i}}
\end{aligned}
$$

were examined, whose integrals can be derived easily analytically.

Figures 6.1 and 6.2 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 3.4, 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 6.3 and 6.4 to a numerical instability of our approach and the choice of polynomials which are exactly integrated. When comparing to the tensor-product

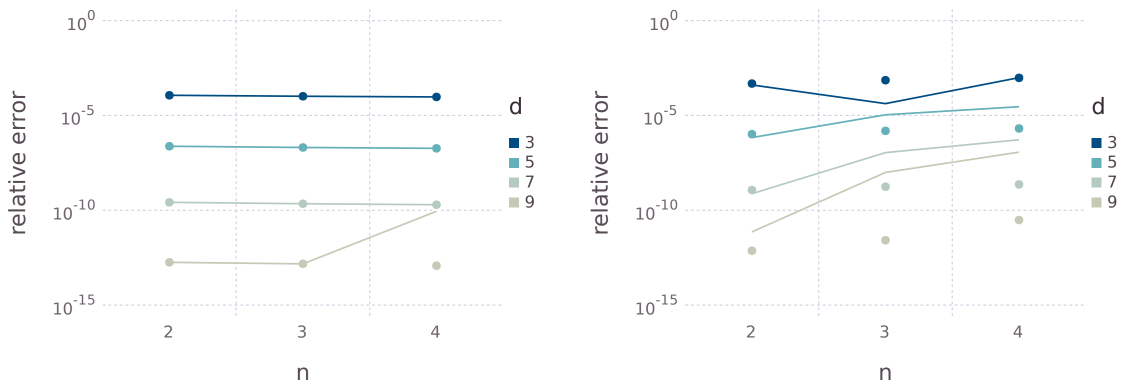

Figure 6.1: Relative error of $g_1$ (left) and $g_2$ (right) as a function of dimension $n$ compared to tensor-product quadrature formulas indicated by circles. The degree of the formulas used is denoted by $d$.

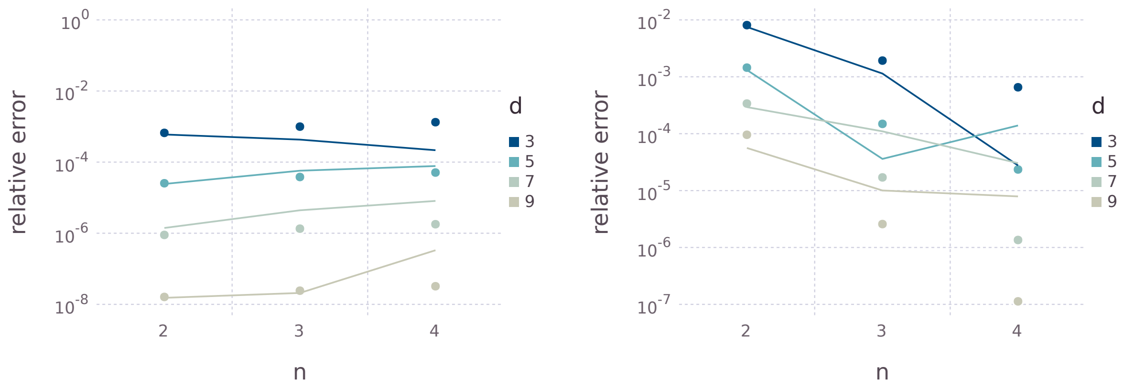

Figure 6.2: Relative error of $g_3$ (left) and $g_4$ (right) as a function of dimension $n$ compared to tensor-product quadrature formulas indicated by circles.The degree of the formulas used is denoted by $d$.

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 6.1.

Tables 6.2, 6.3, 6.4, and 6.5 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 6.6, 6.7, 6.8, and 6.9 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 where $N$ evaluations are exceeded. This is not a very natural way of applying an adaptive sparse

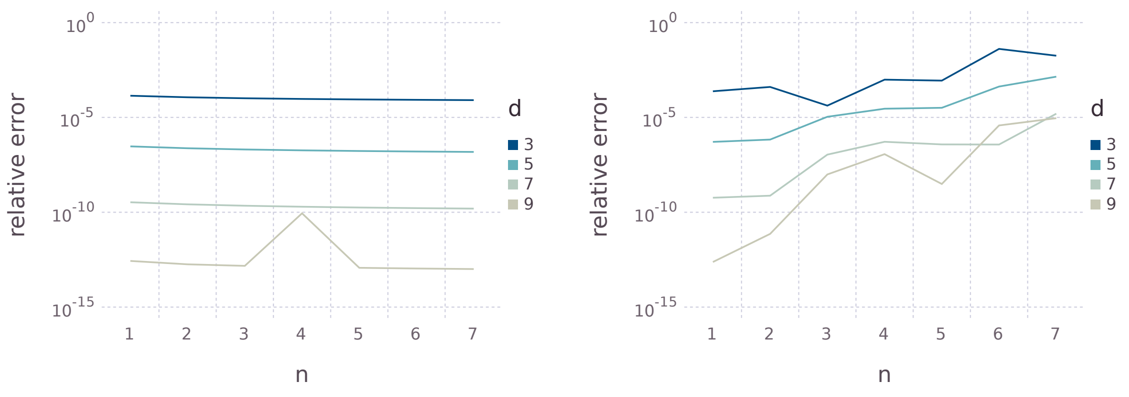

Figure 6.3: Relative error of $g_1$ (left) and $g_2$ (right) as a function of dimension $n$. The degree of the formulas used is denoted by $d$.

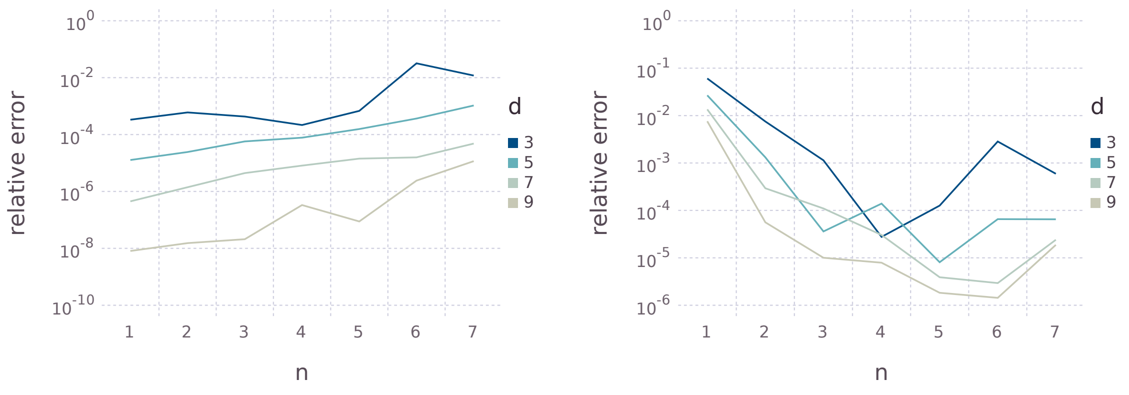

Figure 6.4: Relative error of $g_3$ (left) and $g_4$ (right)g as a function of dimension $n$. The degree of the formulas used is denoted by $d$.

grid method, but otherwise the runtime and 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 behaviour as much. In terms of efficiency, the multisymmetric cubature rule seems to be superior to both the quasi-Monte-Carlo and sparse-grid methods.

## 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 6.10. 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 func-

| | Sym. Cubature | | Quasi Monte Carlo | | |
|---|---|---|---|---|---|
| $n$ | $d=5$ | $d=9$ | $N=10^2$ | $N=10^3$ | $N=10^4$ |
| 1 | $2.9\cdot10^{-7}$ | $2.7\cdot10^{-13}$ | $4.3\cdot10^{-4}$ | $1.6\cdot10^{-4}$ | $4.8\cdot10^{-5}$ |
| 2 | $2.3\cdot10^{-7}$ | $1.8\cdot10^{-13}$ | $2.9\cdot10^{-4}$ | $4.0\cdot10^{-4}$ | $4.3\cdot10^{-5}$ |
| 3 | $2.0\cdot10^{-7}$ | $1.5\cdot10^{-13}$ | $1.2\cdot10^{-3}$ | $1.4\cdot10^{-4}$ | $3.0\cdot10^{-5}$ |
| 4 | $1.8\cdot10^{-7}$ | $8.8\cdot10^{-11}$ | $8.6\cdot10^{-4}$ | $1.4\cdot10^{-4}$ | $4.2\cdot10^{-6}$ |
| 5 | $1.7\cdot10^{-7}$ | $1.2\cdot10^{-13}$ | $1.5\cdot10^{-3}$ | $1.5\cdot10^{-4}$ | $4.7\cdot10^{-6}$ |

Table 6.2: 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 | $5.1\cdot10^{-7}$ | $2.4\cdot10^{-13}$ | $1.3\cdot10^{-3}$ | $3.1\cdot10^{-4}$ | $1.0\cdot10^{-4}$ |
| 2 | $6.7\cdot10^{-7}$ | $7.2\cdot10^{-12}$ | $3.9\cdot10^{-3}$ | $5.3\cdot10^{-4}$ | $1.2\cdot10^{-5}$ |
| 3 | $1.1\cdot10^{-5}$ | $9.8\cdot10^{-9}$ | $3.9\cdot10^{-3}$ | $1.6\cdot10^{-3}$ | $1.3\cdot10^{-4}$ |
| 4 | $2.8\cdot10^{-5}$ | $1.1\cdot10^{-7}$ | $2.5\cdot10^{-3}$ | $3.9\cdot10^{-3}$ | $2.4\cdot10^{-4}$ |
| 5 | $3.2\cdot10^{-5}$ | $3.1\cdot10^{-9}$ | $3.4\cdot10^{-3}$ | $9.6\cdot10^{-3}$ | $9.5\cdot10^{-4}$ |

Table 6.3: 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 | $1.2\cdot10^{-5}$ | $8.1\cdot10^{-9}$ | $1.4\cdot10^{-3}$ | $2.7\cdot10^{-4}$ | $7.4\cdot10^{-5}$ |
| 2 | $2.4\cdot10^{-5}$ | $1.5\cdot10^{-8}$ | $2.3\cdot10^{-3}$ | $3.1\cdot10^{-4}$ | $1.2\cdot10^{-4}$ |
| 3 | $5.7\cdot10^{-5}$ | $2.1\cdot10^{-8}$ | $1.1\cdot10^{-4}$ | $4.3\cdot10^{-4}$ | $5.0\cdot10^{-5}$ |
| 4 | $7.7\cdot10^{-5}$ | $3.3\cdot10^{-7}$ | $7.7\cdot10^{-4}$ | $1.7\cdot10^{-3}$ | $9.7\cdot10^{-4}$ |
| 5 | $1.5\cdot10^{-4}$ | $8.8\cdot10^{-8}$ | $6.8\cdot10^{-3}$ | $2.0\cdot10^{-3}$ | $1.7\cdot10^{-4}$ |

Table 6.4: 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 $g_3$.

tions remain multisymmetric and that the norm of $a$ satisfies the value prescribed in Table 6.10. Since these parameters are chosen randomly, we use the standard Monte-Carlo method for computing the relative root mean square error (rRMSE)

$$\mathrm{rRMSE} := \sqrt{\frac{\mathbb{E}\left[\left(\int_{[0,1]^{nm}} f(x)\mathrm{d}x - Q(f)\right)^2\right]}{\mathbb{E}\left[\left(\int_{[0,1]^{nm}} f(x)\mathrm{d}x\right)^2\right]}}. \tag{6.1}$$

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

| | Sym. Cubature | | Quasi Monte Carlo | | |
|---|---|---|---|---|---|
| $n$ | $d=5$ | $d=9$ | $N=10^2$ | $N=10^3$ | $N=10^4$ |
| 1 | $2.6\cdot10^{-2}$ | $7.6\cdot10^{-3}$ | $2.7\cdot10^{-2}$ | $5.6\cdot10^{-3}$ | $6.8\cdot10^{-4}$ |
| 2 | $1.3\cdot10^{-3}$ | $5.6\cdot10^{-5}$ | $1.0\cdot10^{-2}$ | $1.9\cdot10^{-3}$ | $2.9\cdot10^{-4}$ |
| 3 | $3.6\cdot10^{-5}$ | $1.0\cdot10^{-5}$ | $6.8\cdot10^{-3}$ | $1.7\cdot10^{-3}$ | $2.0\cdot10^{-4}$ |
| 4 | $1.4\cdot10^{-4}$ | $7.9\cdot10^{-6}$ | $4.9\cdot10^{-3}$ | $1.2\cdot10^{-3}$ | $1.9\cdot10^{-4}$ |
| 5 | $8.0\cdot10^{-6}$ | $1.8\cdot10^{-6}$ | $4.1\cdot10^{-3}$ | $1.0\cdot10^{-3}$ | $1.3\cdot10^{-4}$ |

Table 6.5: 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 | | Sparse Grid | | |
|---|---|---|---|---|---|
| $n$ | $d=5$ | $d=9$ | $N=10^2$ | $N=5\cdot10^2$ | $N=10^3$ |
| 1 | $2.9\cdot10^{-7}$ | $2.7\cdot10^{-13}$ | $4.8\cdot10^{-16}$ | $6.9\cdot10^{-13}$ | $1.8\cdot10^{-13}$ |
| 2 | $2.3\cdot10^{-7}$ | $1.8\cdot10^{-13}$ | $4.9\cdot10^{-6}$ | $6.9\cdot10^{-8}$ | $7.5\cdot10^{-13}$ |
| 3 | $2.0\cdot10^{-7}$ | $1.5\cdot10^{-13}$ | $8.5\cdot10^{-4}$ | $1.0\cdot10^{-7}$ | $7.2\cdot10^{-8}$ |
| 4 | $1.8\cdot10^{-7}$ | $8.8\cdot10^{-11}$ | $1.3\cdot10^{-3}$ | $6.2\cdot10^{-6}$ | $2.4\cdot10^{-6}$ |
| 5 | $1.7\cdot10^{-7}$ | $1.2\cdot10^{-13}$ | $1.6\cdot10^{-3}$ | $9.4\cdot10^{-6}$ | $6.3\cdot10^{-6}$ |

Table 6.6: 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 | | |
|---|---|---|---|---|---|
| $n$ | $d=5$ | $d=9$ | $N=10^2$ | $N=5\cdot10^2$ | $N=10^3$ |
| 1 | $5.1\cdot10^{-7}$ | $2.4\cdot10^{-13}$ | $1.9\cdot10^{-15}$ | $2.8\cdot10^{-14}$ | $9.6\cdot10^{-13}$ |
| 2 | $6.7\cdot10^{-7}$ | $7.2\cdot10^{-12}$ | $9.2\cdot10^{-6}$ | $1.3\cdot10^{-8}$ | $8.2\cdot10^{-12}$ |
| 3 | $1.1\cdot10^{-5}$ | $9.8\cdot10^{-9}$ | $1.1\cdot10^{-4}$ | $6.5\cdot10^{-7}$ | $6.3\cdot10^{-7}$ |
| 4 | $2.8\cdot10^{-5}$ | $1.1\cdot10^{-7}$ | $5.9\cdot10^{-3}$ | $2.9\cdot10^{-4}$ | $9.1\cdot10^{-6}$ |
| 5 | $3.2\cdot10^{-5}$ | $3.1\cdot10^{-9}$ | $1.4\cdot10^{-2}$ | $9.6\cdot10^{-4}$ | $7.4\cdot10^{-4}$ |

Table 6.7: 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 $g_2$.

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 6.5, 6.6, 6.7, 6.8, and 6.9 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

| | 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 6.8: 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 | $2.6 \cdot 10^{-2}$ | $7.6 \cdot 10^{-3}$ | $1.8 \cdot 10^{-3}$ | $9.9 \cdot 10^{-3}$ | $1.2 \cdot 10^{-3}$ |
| 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 | $8.0 \cdot 10^{-6}$ | $1.8 \cdot 10^{-6}$ | $2.0 \cdot 10^{-3}$ | $4.7 \cdot 10^{-4}$ | $3.9 \cdot 10^{-4}$ |

Table 6.9: 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 $g_4$.

Table 6.10: Genz Functions

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

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 favours Monte-Carlo methods. This explanation is consistent with the observation that sparse grids and multisymmetric cubature formulas outperform Monte-Carlo methods for very regular integrands, while less regular integrand families (see Figure 6.8 and Figure 6.9) seem to favor Monte-Carlo methods.

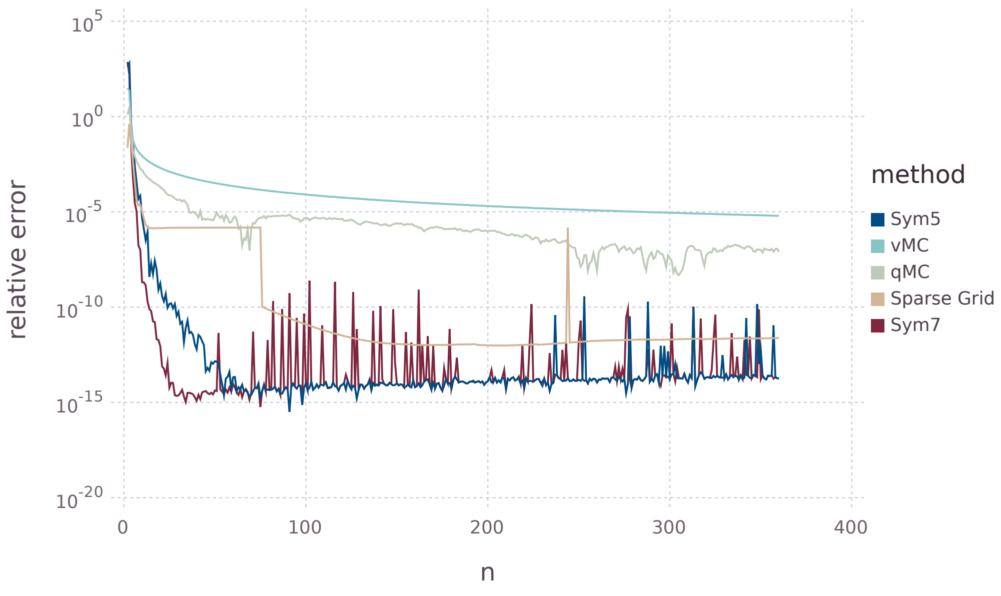

Figure 6.5: Relative error of different integration methods for Genz's $n$-dimensional oscillatory function $f_1$.

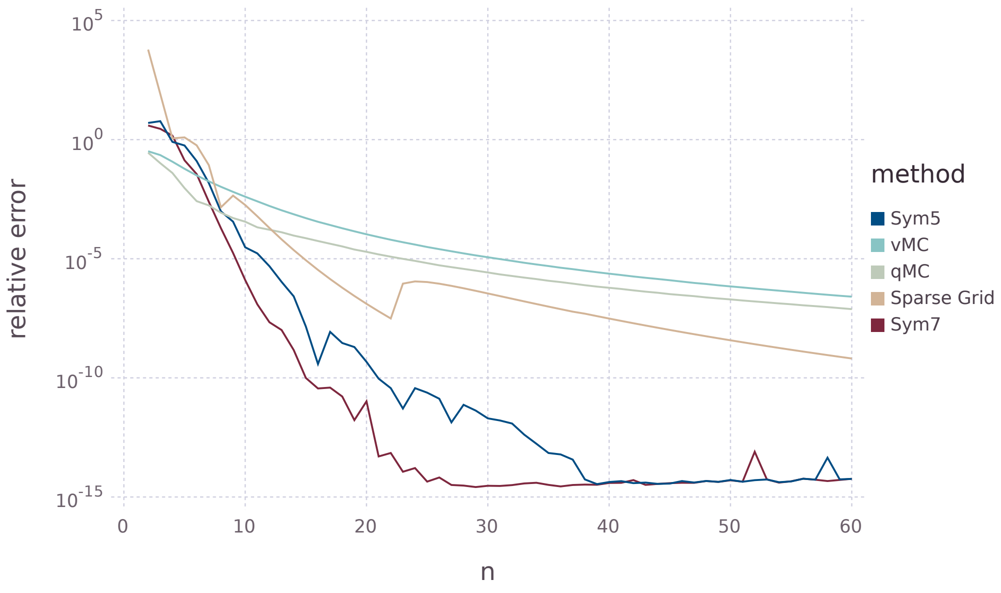

Figure 6.6: Relative error of different integration methods for Genz's $n$-dimensional product-peak function $f_2$.

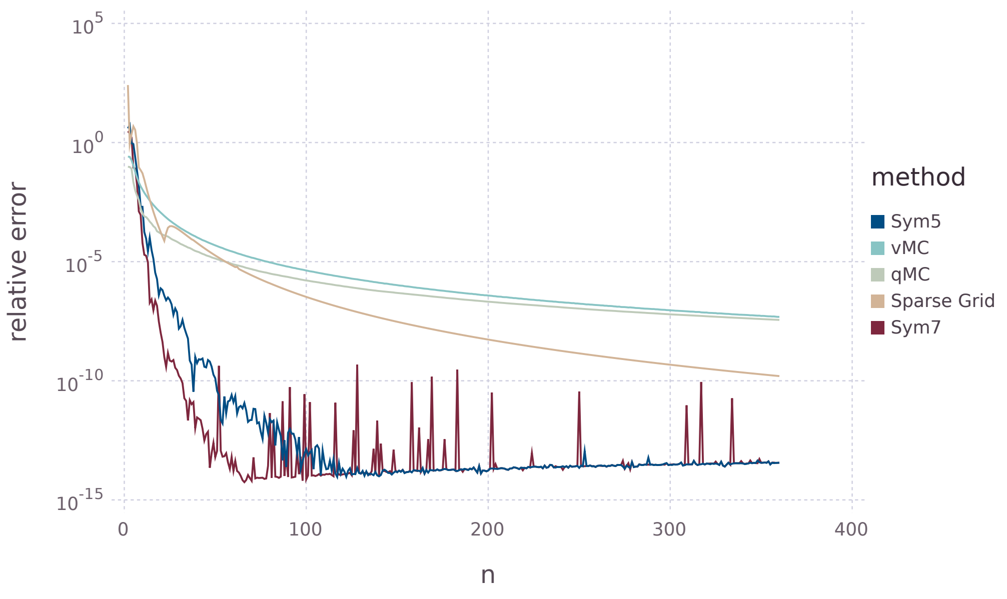

Figure 6.7: Relative error of different integration methods for Genz's $n$-dimensional gaussian function $f_4$.

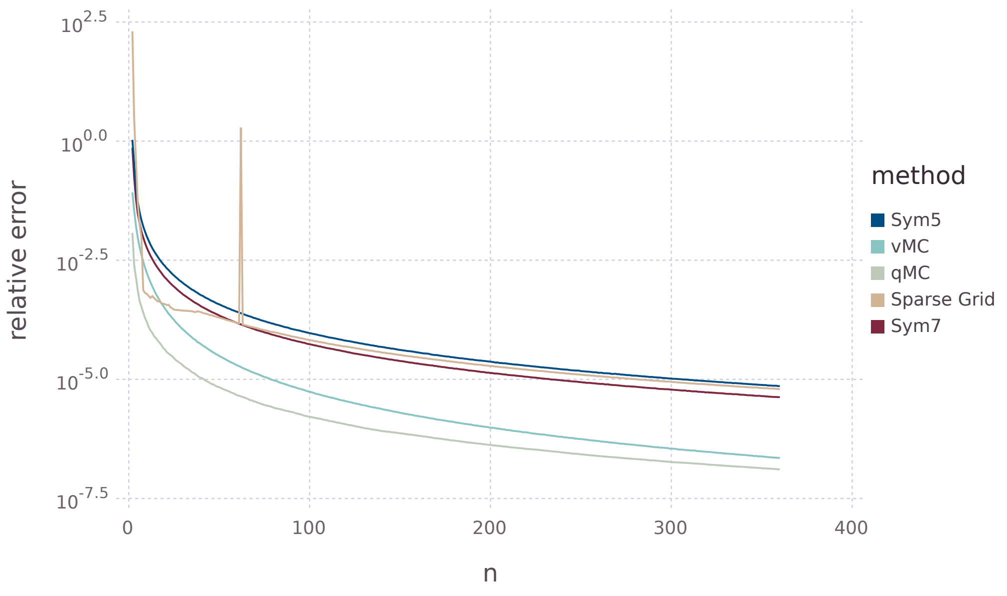

Figure 6.8: Relative error of different integration methods for Genz's $n$-dimensional continuous function $f_5$.

## 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

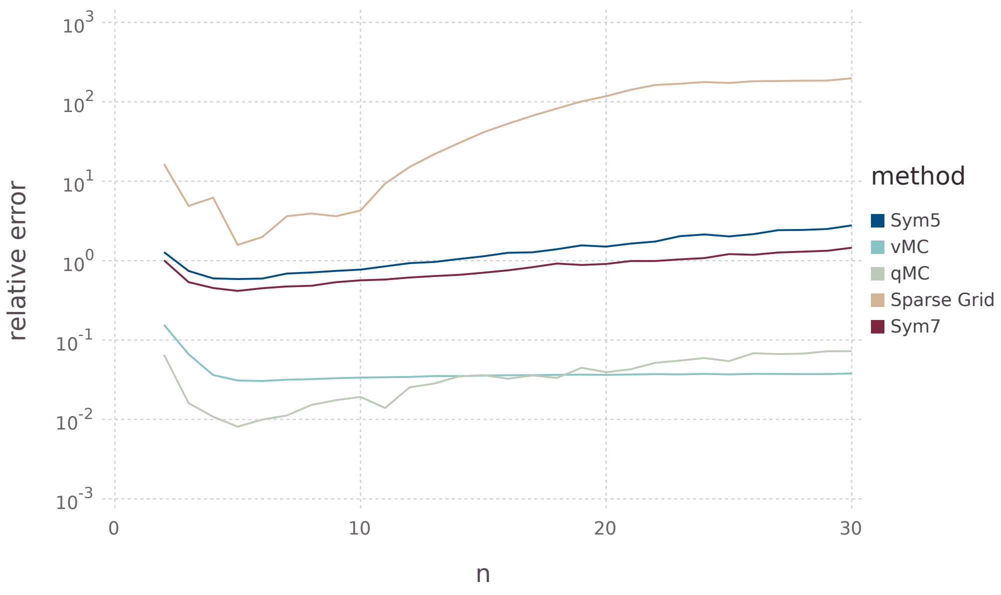

Figure 6.9: Relative error of different integration methods for Genz's $n$-dimensional discontinuous function $f_6$.

Carlo method and compare the accuracy of the results obtained this way.

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

$$-\Delta u(x,y,\omega) = f(x,y,\omega) \quad \text{in } D \times \Omega, \tag{6.2a}$$
$$u(x,y,\omega) = g(x,y) \quad \text{on } \partial D \times \Omega, \tag{6.2b}$$

where we assume that $f$ and $g$ are sufficiently smooth in $(x,y)$ such that the solution is classic. Integrating with respect to $dP(\omega)$, we obtain

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

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

$$-\Delta \mathbb{E}[u] = \mathbb{E}[f] \quad \text{in } D, \tag{6.3}$$
$$\mathbb{E}[u] = g \quad \text{on } \partial D, \tag{6.4}$$

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 $\omega$ in (6.2), we use the open-source Julia programming language [10]. For testing purposes, we consider the right-hand side

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

where $U_i \sim U(0, 1)$, i.e., we choose $(U_i)_{i=1}^n$ to be an i.i.d. sequence of uniformly distributed random variables. We choose $g := 1$ such that the expectation of equation (6.2) satisfies the equation

$$-\Delta\mathbb{E}[u] = \frac{1 - \exp\left(-(x^2 + y^2)\right)}{x^2 + y^2} \quad \text{in } D, \tag{6.5}$$

$$\mathbb{E}[u] = 1 \quad \text{on } \partial D. \tag{6.6}$$

In this numerical example, we set $n := 15$. Figure 6.10 shows the absolute error of the exact expectation obtained by solving (6.3) 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 6.11 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 slower for the quasi-Monte Carlo method compared to the multisymmetric cubature formula. For $d = 3$, a total amount of four evaluation is needed, resulting in an $L^2$-error of $5 \cdot 10^{-5}$, whereby 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 is improving by two orders of magnitude to $6.9 \cdot 10^{-6}$, whereby 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.

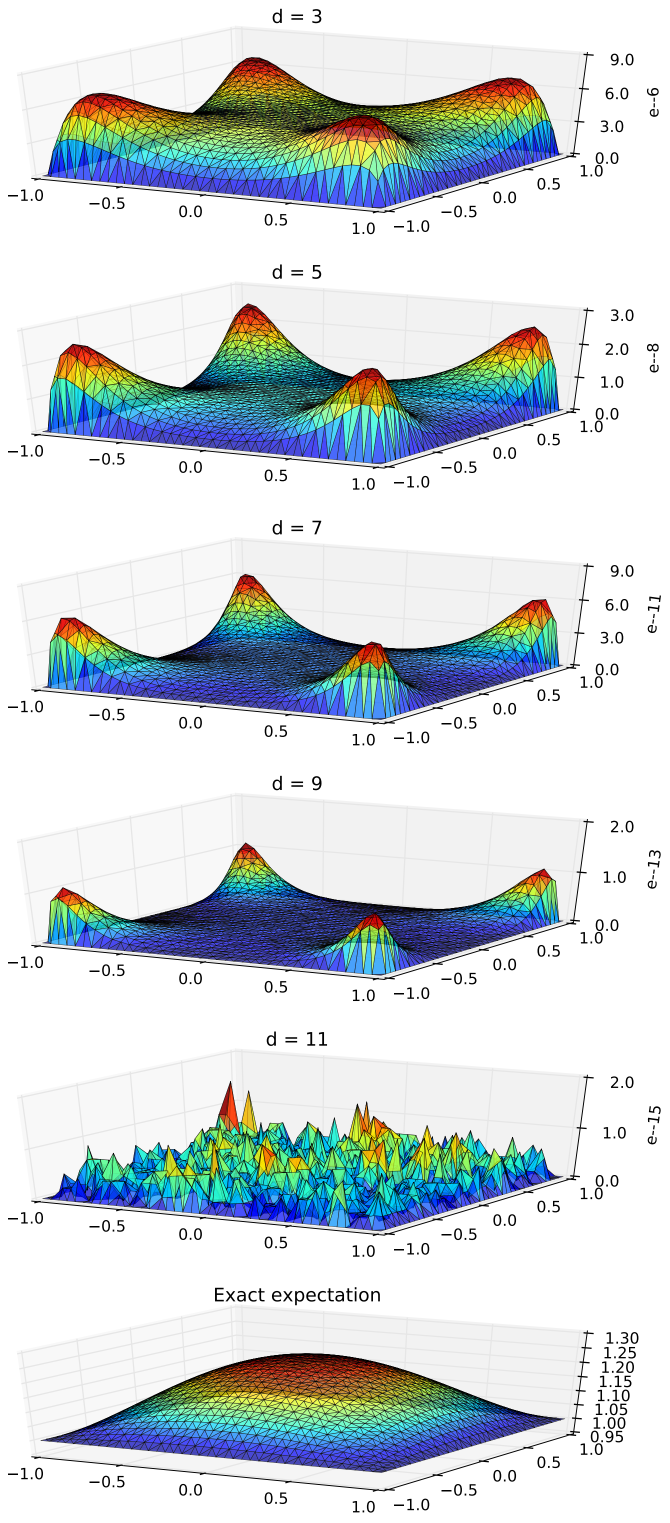

Figure 6.10: The first plots show the absolute error of the exact expectation obtained by solving (6.3) compared to the approximation obtained by using a multisymmetric cubature rule of degree $d$. The exact expectation is shown in the last plot.

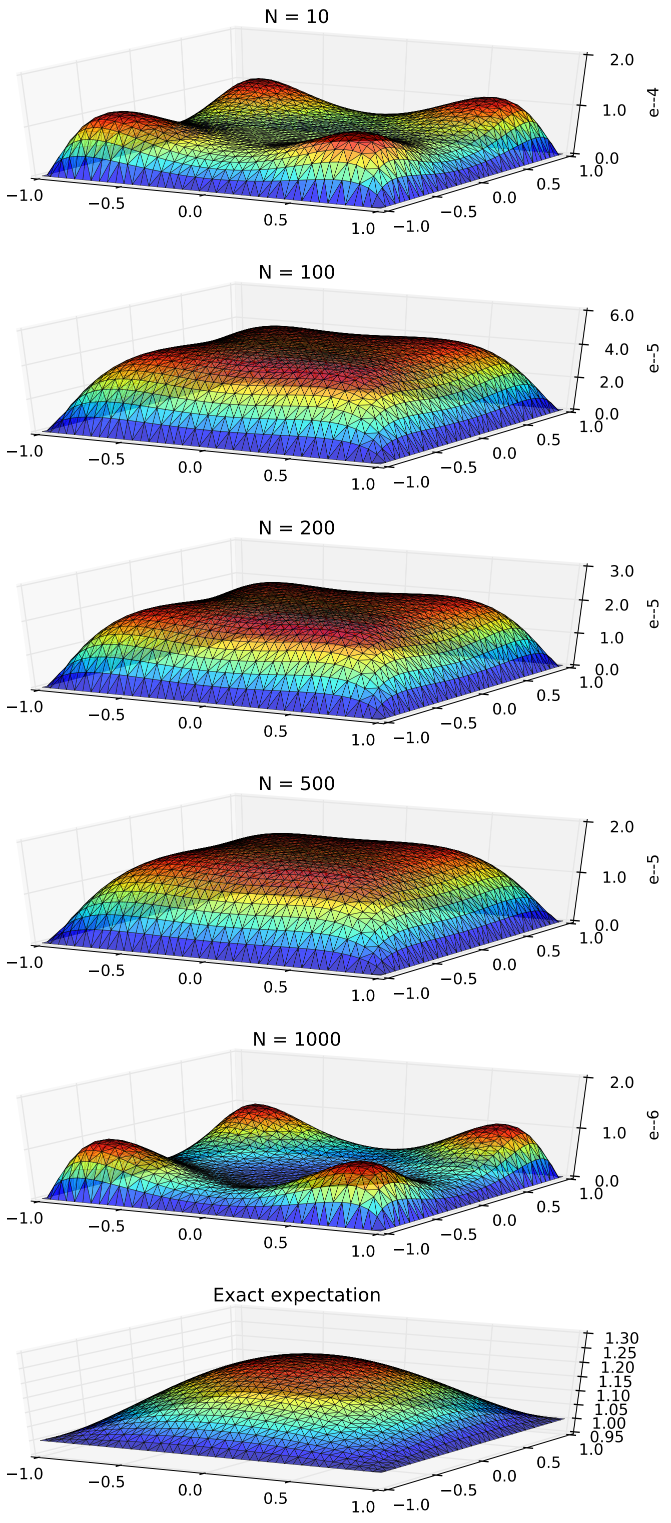

Figure 6.11: The first plots show the absolute error of the exact expectation obtained by solving (6.3) compared to the approximation obtained by using a quasi-Monte Carlo method with $N$ samples. The exact expectation is shown in the last plot.

# CHAPTER 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 1.2. These cubature formulas are of immediate importance for the numerical approximation of solutions of stochastic partial differential equations. Theoretical results for spaces of $G$-invariant functions were shown in Chapter 3.1 as well as standard error bounds in Section 3.3. In the following, a general scheme for computing cubature formulas of $G$-invariant functions was developed in Chapter 4. Based on that, a special kind of $G$-invariance, the notion of multisymmetry (see Definition 1.1) and the corresponding polynomial spaces were further examined in Chapter 5. Finally, the algorithms were implemented and numerical results were shown in Chapter 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 Chapter 6, the expectation of a simple stochastic 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 computational expensive tensor-product rule in terms of relative error to a certain extent. In both examined cases, namely the fully symmetric and the multisymmetric one for $m = 2$, it was found that the proposed multisymmetric cubature formulas require far less 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 respective integrand.The results for the stochastic partial differential equation reenforce 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 less function evaluations. This result suggests that multisymmetric cubature formulas could 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 a Bayesian parameter estimation.

In particular, we want to point out that the required amount 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 Chapter 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 couple of limitations 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.

# List of Figures

# Bibliography

[1] T. Bagby, L. Bos, and N. Levenberg. Multivariate simultaneous approximation. *Constr. Approx.*, 18(4):569–577, 2002.

[2] I. Bogaert. Iteration-free computation of Gauss-Legendre quadrature nodes and weights. *SIAM J. Sci. Comput.*, 36(3):a1008–a1026, 2014.

[3] Paul Bratley and Bennett L Fox. Algorithm 659: Implementing sobol's quasirandom sequence generator. *ACM Transactions on Mathematical Software (TOMS)*, 14(1):88–100, 1988.

[4] Emmanuel Briand. *Polynômes multisymétriques.* Theses, October 2002. Thèse en cotutelle avec l'Universidad de Cantabria. Co-directeur : Laureano Gonzalez-Vega. Rapporteurs : Michel Brion, Bernard Mourrain, Jean-Yves Thibon. Autres membres du jury : Consuelo Martinez Lopez, Tomas Recio Muniz, Felix Ulmer.

[5] Hans-Joachim Bungartz and Michael Griebel. Sparse grids. *Acta numerica*, 13:147–269, 2004.

[6] Ronald Cools and Philip Rabinowitz. Monomial cubature rules since "Stroud": A compilation. *J. Comput. Appl. Math.*, 48(3):309–326, 1993.

[7] Philip J. Davis and Philip Rabinowitz. *Methods of numerical integration. Corrected reprint of the 1984 2nd ed.* Mineola, NY: Dover Publications, corrected reprint of the 1984 2nd ed. edition, 2007.

[8] Eric A. Devuyst and Paul V. Preckel. Gaussian cubature: a practitioner's guide. *Math. Comput. Modelling*, 45(7-8):787–794, 2007.

[9] Josef Dick, Frances Y. Kuo, and Ian H. Sloan. High-dimensional integration: The quasi-Monte Carlo way. *Acta Numerica*, 22:133–288, 2013.

[10] Caroline Geiersbach, Clemens Heitzinger, Gudmund Pammer, Stefan Rigger, and Gerhard Tulzer. A 2d finite element method solver for drift-diffusion-poisson systems and semilinear poisson equations written in Julia. `https://github.com/Stivanification/DriftDiffusionPoissonSystems.jl`, 2016.

[11] Alan Genz. A package for testing multiple integration subroutines. In *Numerical Integration*, pages 337–340. Springer, 1987.

[12] Thomas Gerstner and Michael Griebel. Dimension–adaptive tensor–product quadrature. *Computing*, 71(1):65–87, 2003.

[13] A. Hinrichs, E. Novak, M. Ullrich, and H. Woźniakowski. The curse of dimensionality for numerical integration of smooth functions. *Math. Comput.*, 83(290):2853–2863, 2014.

[14] Stephen Joe and Frances Y Kuo. Remark on algorithm 659: Implementing sobol's quasirandom sequence generator. *ACM Transactions on Mathematical Software (TOMS)*, 29(1):49–57, 2003.

[15] Harald Niederreiter. *Random number generation and quasi-Monte Carlo methods.* SIAM, 1992.

[16] E Novak and K Ritter. Simple cubature formulas for d-dimensional integrals with high polynomial exactness and small error. *Report, Institut für Mathematik, Universität Erlangen–Nürnberg*, 1997.

[17] Dirk Nuyens, Gowri Suryanarayana, and Markus Weimar. Rank-1 lattice rules for multivariate integration in spaces of permutation-invariant functions. Error bounds and tractability. *Adv. Comput. Math.*, 42(1):55–84, 2016.

[18] Art B. Owen. Latin supercube sampling for very high-dimensional simulations. *ACM Trans. Model. Comput. Simul.*, 8(1):71–102, January 1998.

[19] Gudmund Pammer and Stefan Rigger. Multisymmetry. `https://github.com/Stivanification/Multisymmetry`, 2017.

[20] David Rydh. A minimal set of generators for the ring of multisymmetric functions. *Annales de l'institut Fourier*, 57(6):1741–1769, 2007.

[21] L Schläfli. Uber die resultante eines systems mehrerer algebraischen gleichungen, denk. der keiser. akad. der wiss., math-naturwiss. klasse, 4 band, 1852; gesammelte abhandlungen, 1953.

[22] Rudolf Michael Schürer. *High-dimensional numerical integration on parallel computers.* Citeseer, 2001.

[23] A.H. Stroud. Approximate calculation of multiple integrals. Prentice-Hall Series in Automatic Computation. Englewood Cliffs, New Jersey: Prentice-Hall, Inc. XIII, 431 p. $25.65 (1971)., 1971.

[24] Francesco Vaccarino. The ring of multisymmetric functions. *Annales de l'institut Fourier*, 55(3):717–731, 2005.

[25] Grzegorz W Wasilkowski and Henryk Wozniakowski. Explicit cost bounds of algorithms for multivariate tensor product problems. *Journal of Complexity*, 11(1):1–56, 1995.

[26] Markus Weimar. The complexity of linear tensor product problems in (anti)symmetric Hilbert spaces. *J. Approx. Theory*, 164(10):1345–1368, 2012.

[27] Markus Weimar. On lower bounds for integration of multivariate permutation-invariant functions. *J. Complexity*, 30(1):87–97, 2014.