Smolyak Quadrature

Vesa Kaarnioja
University of Helsinki
Department of Mathematics and Statistics

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta/Osasto — Fakultet/Sektion — Faculty Faculty of Science	Laitos — Institution — Department Department of Mathematics and Statistics

Tekijä — Författare — Author
Vesa Kaarnioja

Työn nimi — Arbetets titel — Title

Smolyak Quadrature

Oppiaine — Läroämne — Subject
Applied mathematics

Työn laji — Arbetets art — Level Master's thesis	Aika — Datum — Month and year May 2013	Sivumäärä — Sidoantal — Number of pages 75 pages

Tiivistelmä — Referat — Abstract

This thesis is an introduction to the theoretical foundation and practical usage of the Smolyak quadrature rule, which is used to evaluate high-dimensional integrals over regions of Euclidean spaces. Given a sequence of univariate quadrature rules, the Smolyak construction is defined in terms of tensor products taken over the univariate rules' consecutive differences. The evaluation points of the resulting multivariate quadrature rule are distributed more sparsely than those of e.g. tensor product quadrature. It can be shown that a multivariate quadrature rule formulated in this way inherits many useful properties of the underlying sequence of univariate quadrature rules, such as the polynomial exactness.

The original formulation of the Smolyak rule is prone to a copious amount of cancellation of terms in practice. This issue can be circumvented by isolating the occurrence of duplicates to a separate term, which can be computed a priori. The resulting combination method forms the basis for a numerical implementation of the Smolyak quadrature rule, which we have provided using the MATLAB scripting language.

Our tests suggest that the Smolyak rule provides a competitive alternative in the realm of multidimensional integration routines saturated by the stochastic Monte Carlo method and the deterministic Quasi-Monte Carlo method. This statement is especially valid in the case of smooth integrands and it is backed by the error analysis developed in the second chapter of this thesis. The classical convergence rate is also derived for integrands of sufficient smoothness in the case of a bounded integration region.

The third chapter serves as a qualitative approach to generalized sparse grid quadrature. Especially of interest is the dimension-adaptive construction. While it lacks the theoretical foundation of the Smolyak quadrature rule, it has the added benefit of adapting to the spatial structure of the integrand. A MATLAB implementation of this routine is presented vis-à-vis the Smolyak quadrature rule.

Avainsanat — Nyckelord — Keywords
Smolyak quadrature, curse of dimensionality, sparse grids, dimension-adaptive quadrature

Säilytyspaikka — Förvaringsställe — Where deposited
Kumpula Science Library

Muita tietoja — Övriga uppgifter — Additional information

Contents

Introduction

This thesis is an introduction to the Smolyak quadrature rule, which is used to evaluate high-dimensional integrals of the form

$$\mathcal{I}_W^d f = \int\limits_{I_1\times\cdots\times I_d} W(x_1,...,x_d)f(x_1,...,x_d)\,\mathrm{d}x_d\cdots\mathrm{d}x_1, \quad \varnothing \neq I_j \subseteq \mathbb{R} \text{ an interval,}$$

where f is integrable in the hyperrectangle $I_1 \times \cdots \times I_d$ and the weight function $W(x_1,...,x_d) = W_1(x_1)\cdots W_d(x_d)$ is a product of nonnegative integrable functions in I_j for $j = 1,...,d$. We interpret $\mathcal{I}_W^d f$ simply as an iterated integral with d nested integral signs.

A straightforward method to evaluate multidimensional integrals such as $\mathcal{I}_W^d f$ is to compound univariate rules coordinate-wise. The problem with this approach lies in the number of function evaluations: using univariate rules with n evaluation points results in a d-dimensional quadrature rule with a total of n^d points – an exponential correspondence with the dimension of the problem! This phenomenon has been dubbed the *curse of dimensionality*, which renders the naïve approach useless even on modern computers for moderately high values of d.

There have been various attempts to remedy the situation. The reader may be familiar with the stochastic *Monte Carlo method* (MC) or the deterministic *Quasi-Monte Carlo method* (QMC). They are variations on the concept of sparse grids and make use of the fact that accuracy of the quadrature rule increases as the discrepancy of the evaluation points decreases. MC uses computer-generated pseudo-random sequences to achieve this goal while QMC utilizes the crutch of number-theoretic sequences.

Given a sequence of univariate rules, the Smolyak quadrature rule is formulated as a sum of tensor products taken over the consecutive differences in the univariate sequence. The primary motive for the Smolyak construction is the fact that the evaluation points are generated more sparsely than those of the compounded approach. It can be shown that the asymptotic behaviour of the error term takes on the alternative methods with increasing order.

This result is known as the *fundamental theorem of Smolyak quadrature* in this thesis.

The first chapter opens with a discussion of the preliminary concepts leading to the definition of the Smolyak quadrature rule. We proceed to derive several properties of the Smolyak rule culminating in the celebrated combination method. The combination method forms the essential ingredient to construct the numerical implementation of the Smolyak rule. A MATLAB implementation of the algorithm is given in appendix C.

In the second chapter we concern ourselves with the approximation properties of the Smolyak rule. It can be shown that Smolyak's construction inherits the polynomial exactness of the underlying univariate rules. The role of polynomial exactness is remarkable since practically no restrictions need to be placed on the univariate sequence to derive strong results. In contrast the explicit convergence rate for Smolyak quadrature is derived only in the case of a bounded integration region.

The third and final chapter of this thesis serves as a qualitative approach to the generalized sparse grid quadrature rule and its relation to the Smolyak rule. The dimension-adaptive construction is presented as well since it provides a more dynamic alternative to high-dimensional integration problems than the relatively static Smolyak rule. MATLAB implementations of these methods are given in appendix C.

The results presented in this thesis fall into two categories: auxiliary results and main results, which are the tensor product theorem, the combination method, polynomial exactness, lemma 2.6 and theorem 2.11. By their nature, auxiliary results are properties which have not been proven in the references of this thesis. Some results such as dimension recursion and proposition 1.7 are known to the authors of selected works in the bibliography, but I have constructed their proofs for the sake of completeness.

The proof of the combination method and lemma 2.6 are referenced from the original paper by Wasilkowski and Woźniakowski [28]. The proof of the tensor product theorem is inspired by [27] but the proof is ultimately carried out using a different strategy. The results regarding polynomial exactness follow the lines of [20] and [16] but have been adapted to fit the tone and scope of this thesis. The inference leading to theorem 2.11 differs from the general approach taken in the source material as it does not exploit nested univariate sequences.

The main results have been generalized in two regards: the integration region is allowed to be a hyperrectangle as opposed to a hypercube and the assumption that the underlying univariate rules are nested has been omitted. This omission is salvaged by a result in approximation theory due to Brass in [3].

0.1 Prerequisites and notation

Knowledge of basic uni- and multivariate calculus should be sufficient for understanding the theoretical portion of this thesis. We shall additionally construct a numerical implementation of the Smolyak quadrature rule and its dimension-adaptive counterpart using MATLAB scripting language. This thesis is not a textbook on scientific computing so knowledge of the aforementioned language is assumed on the part of the reader.

We go through some of the necessary terminology used in the course of this thesis. Let E be an arbitrary normed space. The set of *linear mappings* from E to $\mathbb{R}$ is called the *algebraic dual space* of E, which we denote by E^*. Elements of the set E^* are called *linear functionals.* On the other hand, the set of continuous, linear functionals from E to $\mathbb{R}$ is called the *topological dual space* of E and it is denoted by E'. The vector space E' forms a complete normed space when it is accompanied by the *operator norm*

$$||T|| = \sup\{|Tx|;\ x \in E \text{ and } ||x||_E \leq 1\}, \quad T \in E',$$

where we understand $|| \cdot ||_E$ as the norm associated with the vector space E. It is a simple exercise to show that the operator norm fulfills the norm postulates.

Let $T \in E^*$. We call T *bounded* if there exists a constant $C \geq 0$ such that $|Tx| \leq C||x||_E$ for all $x \in E$. It is well-known from elementary functional analysis that T is continuous if and only if T is bounded. The last requirement is in turn equivalent to saying that $||T|| < \infty$. Hence for continuous, linear functionals $T\colon E \to \mathbb{R}$ we have

$$|Tx| \leq ||T|| \cdot ||x||_E \quad \text{for all } x \in E.$$

We follow the convention $0 \in \mathbb{N}$. Throughout this thesis we employ so-called *multi-index* notation. If $\alpha \in \mathbb{N}^d$, then we refer to its j^{th} coordinate universally as α_j. Let $\beta \in \mathbb{N}^d$. We write $\alpha \geq \beta$ if $\alpha_j \geq \beta_j$ for all $j = 1, ..., d$. We define additionally the shorthand $\mathbf{1} = (1, ..., 1) \in \mathbb{N}^d$.

We define the following multi-index norms

$$|\alpha|_1 = \sum_{i=1}^{d} \alpha_i \quad \text{and} \quad |\alpha|_\infty = \max_{1 \leq i \leq d} \alpha_i$$

and introduce the following convention for the mixed derivative operator:

$$\frac{\partial^\alpha}{\partial x^\alpha} = \frac{\partial^{|\alpha|_1}}{\partial x_1^{\alpha_1} \cdots \partial x_d^{\alpha_d}}, \quad x = (x_1, ..., x_d) \in \mathbb{R}^d.$$

The derivative operator has the *order* $|\alpha|_1$ and the *mixed order* $|\alpha|_\infty$.

Let $\varnothing \neq \Omega \subseteq \mathbb{R}^{d_1}$ and $\varnothing \neq \Xi \subseteq \mathbb{R}^{d_2}$. Suppose that $f : \Omega \times \Xi \to \mathbb{R}$ has continuous mixed derivatives up to mixed order r. Let $\alpha \in \mathbb{N}^{d_1}$ and $\beta \in \mathbb{N}^{d_2}$ such that $|\alpha|_\infty \leq r$ and $|\beta|_\infty \leq r$. In the course of this thesis, we occasionally employ the following nonstandard notation for the derivative operator:

$$\frac{\partial^\alpha f(x,y)}{\partial x^\alpha} = \frac{\partial^{\tilde{\alpha}} f(z)}{\partial z^{\tilde{\alpha}}} \quad \text{and} \quad \frac{\partial^\beta f(x,y)}{\partial y^\beta} = \frac{\partial^{\tilde{\beta}} f(z)}{\partial z^{\tilde{\beta}}},$$

where $x \in \Omega$, $y \in \Xi$, $z \in \Omega \times \Xi$ and we have set $\tilde{\alpha} = (\alpha, 0, ..., 0) \in \mathbb{N}^{d_1+d_2}$ and $\tilde{\beta} = (0, ..., 0, \beta) \in \mathbb{N}^{d_1+d_2}$.

The terminology and theory behind quadrature rules of one variable is briefly discussed in appendix A and the relevant combinatorial results are presented in appendix B. Especially of interest are the cardinalities of multi-index sets, which have been collected to theorem B.3 and the accompanying corollary B.4.

Chapter 1

The Smolyak method

In this chapter we introduce the Smolyak quadrature rule. In the original paper [25] by Smolyak, the method was developed for general tensor product spaces. Although we make use of the terminology concerning tensor product spaces, we shall not develop the theory to this extent. We instead limit ourselves to integrals over regions of $\mathbb{R}^d$, i.e. connected and non-empty subsets, which may contain some of their boundary points.

In addition we shall discuss the implications of the Smolyak quadrature formula and construct its numerical implementation based on the combination method. The convergence properties of this method are pursued in the subsequent chapter. Before we press onto these matters, we must cover some essential groundwork.

We turn our attention to the function spaces

$$H^r(\Omega) = \left\{ f : \Omega \to \mathbb{R};\ \frac{\partial^\alpha f(x)}{\partial x^\alpha} \text{ exists and is bounded in } \Omega \text{ for all } |\alpha|_\infty \leq r \right\}$$

for a fixed region $\varnothing \neq \Omega \subseteq \mathbb{R}^d$. We call r *regularity* of functions in $H^r(\Omega)$ and accompany these function classes with the respective norms

$$||f||_{H^r(\Omega)} = \max_{\substack{\alpha \in \mathbb{N}^d \\ |\alpha|_\infty \leq r}} \sup \left\{ \left| \frac{\partial^\alpha f(x)}{\partial x^\alpha} \right|;\ x \in \Omega \right\}.$$

1.1 Tensor product

Suitably defined tensor products are a convenient tool to factorize mathematical entities with product structure. Tensor product factorization can be applied to functions, linear operators or vector spaces among others. In this section we follow the outline presented in [27] and begin by limiting ourselves to tensor products defined for linear functionals of definite form – namely that of quadrature rules.

Definition 1.1. Suppose that $\varnothing \neq \Omega \subseteq \mathbb{R}^{d_1}$ and $\varnothing \neq \Xi \subseteq \mathbb{R}^{d_2}$ and let $S\colon H^r(\Omega) \to \mathbb{R}$ and $T\colon H^r(\Xi) \to \mathbb{R}$ be functionals. Suppose additionally that they admit to representations

$$Sf = \sum_{i=1}^{m} a_i f(x_i) \quad \text{and} \quad T\tilde{f} = \sum_{i=1}^{n} b_i \tilde{f}(y_i)$$

for a selection of positive weights $(a_i)_{i=1}^m$ and $(b_i)_{i=1}^n$ and vectors $(x_i)_{i=1}^m$ and $(y_i)_{i=1}^n$ in the domains Ω and Ξ. Now $\Omega \times \Xi \subseteq \mathbb{R}^{d_1+d_2}$ and the *tensor product* of S and T is the linear functional $S \otimes T\colon H^r(\Omega \times \Xi) \to \mathbb{R}$ defined by setting

$$S \otimes Tf = \sum_{i=1}^{m} \sum_{j=1}^{n} a_i b_j f(x_i, y_j).$$

Remark 1. The quadrature-like operators in the previous definition are linear and bounded. Consider for example the operator S. Let $f \in H^r(\Omega)$ and $\alpha \in \mathbb{N}^{d_1}$, $|\alpha|_\infty \leq r$. Then

$$\left| S \frac{\partial^\alpha f(x)}{\partial x^\alpha} \right| = \left| \sum_{i=1}^{m} a_i \frac{\partial^\alpha f(x)}{\partial x^\alpha} \bigg|_{x=x_i} \right| \leq m \max_{1\leq i\leq m} |a_i| \max_{\substack{\alpha\in\mathbb{N}^{d_1} \\ |\alpha|_\infty\leq r}} \sup_{x\in\Omega} \left| \frac{\partial^\alpha f(x)}{\partial x^\alpha} \right|.$$

Since α was arbitrary, taking the supremum over the set of functions $f \in H^r(\Omega)$ such that $||f||_{H^r(\Omega)} \leq 1$ yields $||S|| \leq m \max\limits_{1\leq i\leq m} |a_i| < \infty$.

Let $\Omega_j \neq \varnothing$ be regions in Euclidean spaces $\mathbb{R}^{d_j}$ and let $T_j\colon H^r(\Omega_j) \to \mathbb{R}$ be functionals for $j = 1, 2, 3, ...$ such that

$$T_j f = \sum_{i=1}^{m_j} w_i^{(j)} f(x_i^{(j)}),$$

where $(w_i^{(j)})_{i=1}^{m_j}$ are positive weights and $(x_i^{(j)})_{i=1}^{m_j}$ is a sequence of vectors in Ω_j. We define the following shorthand notation:

$$\bigotimes_{i=1}^{1} T_i = T_1 \quad \text{and} \quad \bigotimes_{i=1}^{n} T_i = \bigotimes_{i=1}^{n-1} T_i \otimes T_n \quad \text{for } n = 2, 3, 4, ... \tag{1.1}$$

By induction with respect to n it is easy to prove that $T_1 \otimes \cdots \otimes T_n$ defines a linear functional $H^r(\Omega_1 \times \cdots \times \Omega_n) \to \mathbb{R}$ such that

$$\bigotimes_{i=1}^{n} T_i f = \sum_{i_1=1}^{m_1} \cdots \sum_{i_n=1}^{m_n} w_{i_1}^{(1)} \cdots w_{i_n}^{(n)} f(x_{i_1}^{(1)}, ..., x_{i_n}^{(n)}) \quad \text{for } n = 1, 2, 3, ...$$

It is immediately obvious from definition 1.1 that the tensor product is not commutative: generally $S \otimes T \neq T \otimes S$ for quadrature-like operators S and T. From the extended definition, we see that the tensor product is associative: $(S \otimes T) \otimes R = S \otimes (T \otimes R)$ for quadrature-like operators S, T and R. Furthermore, the form of the operators permits the use of the distributive identity $(S + T) \otimes R = S \otimes R + T \otimes R$, when $+$ is taken to be the ordinary pointwise sum.

The role of the tensor product in the scope of this thesis is twofold. It is used directly in the formulation of the Smolyak quadrature rule as we shall see in the next section. On the other hand, it has an important isometric property in the dual of $H^r(\Omega)$ spaces which is essential in the derivation of the error term of the quadrature rule. This is a nontrivial property which we will formulate in the following theorem.

Theorem 1.2 (Tensor product theorem)**.** *Let $\Omega_j \neq \varnothing$ be regions in Euclidean spaces $\mathbb{R}^{d_j}$ and define the functionals $T_j \colon H^r(\Omega_j) \to \mathbb{R}$ for all $j = 1, 2, 3, ..., n$ by setting*

$$T_j f = \sum_{i=1}^{m_j} w_i^{(j)} f(x_i^{(j)}),$$

where $(w_i^{(j)})_{i=1}^{m_j}$ are positive weights and $(x_i^{(j)})_{i=1}^{m_j}$ is in Ω_j. Then

$$\left|\left| \bigotimes_{i=1}^{n} T_i \right|\right| = \prod_{i=1}^{n} ||T_i||$$

in the respective operator norms.

Proof. It is sufficient to prove the claim for $n = 2$. The general case follows via induction due to representation (1.1).

Let $S \colon H^r(\Omega) \to \mathbb{R}$ and $T \colon H^r(\Xi) \to \mathbb{R}$ be functionals such that

$$Sf = \sum_{i=1}^{m} a_i f(x_i) \quad \text{and} \quad T\tilde{f} = \sum_{i=1}^{n} b_i \tilde{f}(y_i).$$

Let $f \in H^r(\Omega \times \Xi)$. Fix $x_0 \in \Omega$ and $\alpha \in \mathbb{N}^d$, $|\alpha|_\infty \leq r$. Define the functions $g \colon \Omega \to \mathbb{R}$ and $h \colon \Xi \to \mathbb{R}$ by setting

$$g(x) = \sum_{i=1}^{n} b_i f(x, y_i) \quad \text{and} \quad h(x) = \left. \frac{\partial^\alpha f(z, x)}{\partial z^\alpha} \right|_{z=x_0}.$$

We immediately find that $g \in H^r(\Omega)$ and $h \in H^r(\Xi)$. Especially $||h||_{H^r(\Xi)} \leq ||f||_{H^r(\Omega\times\Xi)}$ regardless of x_0 and α. We compute

$$\left|\left.\frac{\partial^\alpha g(x)}{\partial x^\alpha}\right|_{x=x_0}\right| = \left|\sum_{i=1}^{n} b_i \left.\frac{\partial^\alpha f(x,y_i)}{\partial x^\alpha}\right|_{x=x_0}\right| = |Th| \leq ||T|| \cdot ||h||_{H^r(\Xi)}$$
$$\leq ||T|| \cdot ||f||_{H^r(\Omega\times\Xi)}.$$

Since x_0 and α were arbitrary, we obtain $||g||_{H^r(\Omega)} \leq ||T|| \cdot ||f||_{H^r(\Omega\times\Xi)}$.

On the other hand, we attain a useful identity:

$$Sg = \sum_{i=1}^{m} a_i g(x_i) = \sum_{i=1}^{m} a_i \sum_{j=1}^{n} b_j f(x_i,y_j) = \sum_{i=1}^{m}\sum_{j=1}^{n} a_i b_j f(x_i,y_j) = S\otimes Tf.$$

These facts yield the inequality

$$|S\otimes Tf| = |Sg| \leq ||S|| \cdot ||g||_{H^r(\Omega)} \leq ||S|| \cdot ||T|| \cdot ||f||_{H^r(\Omega\times\Xi)}.$$

Taking the supremum over the set $\{f \in H^r(\Omega\times\Xi);\ ||f||_{H^r(\Omega\times\Xi)} \leq 1\}$ implies $||S\otimes T|| \leq ||S|| \cdot ||T||$.

The other inequality follows in a trivial manner. Let $g \in H^r(\Omega)$ such that $||g||_{H^r(\Omega)} \leq 1$ and $h \in H^r(\Xi)$ such that $||h||_{H^r(\Xi)} \leq 1$. Define $f(x,y) = g(x)h(y)$. We observe that $||f||_{H^r(\Omega\times\Xi)} \leq 1$ and achieve

$$||S\otimes T|| \geq |S\otimes Tf| = \left|\sum_{i=1}^{m}\sum_{j=1}^{n} a_i b_j g(x_i) h(y_j)\right| = \left|\sum_{i=1}^{m} a_i g(x_i)\right| \cdot \left|\sum_{j=1}^{n} b_j h(y_j)\right|$$
$$= |Sg| \cdot |Th|.$$

Taking suprema over the right-hand side of the inequality above yields the desired result $||S\otimes T|| = ||S|| \cdot ||T||$. □

Example 1.3. Consider the problem of evaluating

$$\mathcal{I}_W^d f = \int\limits_{I_1} \cdots \int\limits_{I_d} W_1(x_1)\cdots W_d(x_d) f(x_1,...,x_d)\,\mathrm{d}x_d \cdots \mathrm{d}x_1$$

using a sequence of univariate quadrature rules $(U_k^{(j)})_{j=1}^d$, where the subscript denotes the number of evaluation points. Suppose that the univariate rules are chosen in such a way that $U_k^{(j)} p = \mathcal{I}_{W_j}^1 p$ holds for all polynomials p of degree at most m_k in I_j. Let $(w_i^{(j)})_{i=1}^k$ be the weights and $(x_i^{(j)})_{i=1}^k$ the evaluation points of the rule $U_k^{(j)}$.

Employing the quadrature rules $U_k^{(j)}$ to the integral $\mathcal{I}_W^d f$ coordinate-wise yields the approximation

$$\mathcal{I}_W^d f \approx \int\limits_{I_1} \cdots \int\limits_{I_{d-1}} W_1(z_1) \cdots W_{d-1}(z_{d-1}) \sum_{i_d=1}^{k} w_{i_d}^{(d)} f(z_1, ..., z_{d-1}, x_{i_d}^{(d)})\, \mathrm{d}z_{d-1} \cdots \mathrm{d}z_1$$
$$\approx \sum_{i_1=1}^{k} \cdots \sum_{i_d=1}^{k} w_{i_1}^{(1)} \cdots w_{i_d}^{(d)} f(x_{i_1}^{(1)}, ..., x_{i_d}^{(d)}).$$

The above can now be expressed in tensor product form

$$\mathcal{I}_W^d f = \bigotimes_{i=1}^{d} U_k^{(i)} f + \text{error}. \tag{1.2}$$

In literature this type of quadrature is typically referred to as *tensor product quadrature.* It turns out that more imaginative use of the tensor product reveals a multivariate quadrature rule that is perhaps less immediate but in many respects more efficient than the approach taken above – at least in terms of the number of function evaluations as the quadrature order is increased.

1.2 Smolyak quadrature rule

Using definition 1.1 of the tensor product, we are ready to present the Smolyak quadrature rule.

Definition 1.4 (Smolyak quadrature rule)**.** Let $(U_i^{(j)})_{i=1}^{\infty}$ be a sequence of univariate quadrature rules in the interval $\varnothing \neq I_j \subseteq \mathbb{R}$, $j = 1, ..., d$. We introduce the *difference operators* in I_j by setting

$$\Delta_0^{(j)} = 0, \quad \Delta_1^{(j)} = U_1^{(j)} \quad \text{and} \quad \Delta_{i+1}^{(j)} = U_{i+1}^{(j)} - U_i^{(j)} \quad \text{for } i = 1, 2, 3, \dots .$$

The *Smolyak quadrature rule* of order k in the hyperrectangle $I_1 \times \cdots \times I_d$ is the operator

$$\mathcal{Q}_k^d = \sum_{\substack{|\alpha|_1 \leq k \\ \alpha \in \mathbb{N}^d}} \bigotimes_{i=1}^{d} \Delta_{\alpha_i}^{(i)}. \tag{1.3}$$

Remark 2. The tensor product $\Delta_{\alpha_1}^{(1)} \otimes \cdots \otimes \Delta_{\alpha_d}^{(d)}$ in the summand of (1.3) vanishes whenever $\alpha_i = 0$ for some index i. In the sequel we always assume that $\alpha \geq \mathbf{1}$ and hence $k \geq d$.

Remark 3. In the case $d = 1$, we obtain

$$\mathcal{Q}_k^1 = \sum_{i=1}^{k} \Delta_i^{(1)} = U_1^{(1)} + (U_2^{(1)} - U_1^{(1)}) + ... + (U_k^{(1)} - U_{k-1}^{(1)}) = U_k^{(1)} \quad \text{for all } k \geq 1.$$

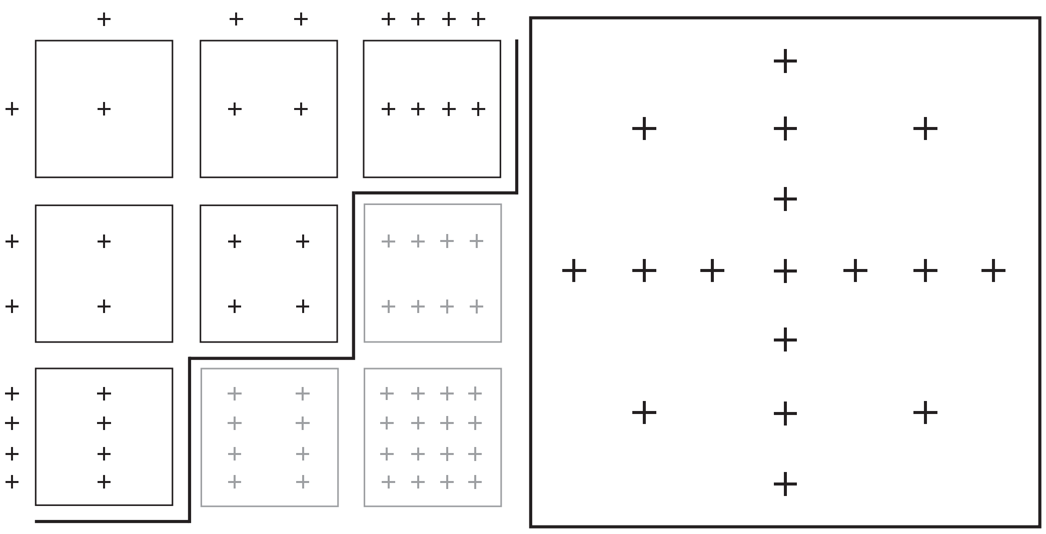
We can directly apply properties of univariate quadrature rules to the Smolyak rule in the one-dimensional case. This makes properties of the Smolyak rule easy to prove by dimension-wise induction.

Using the difference operators defined above, we can write the tensor product operator (1.2) of order k in the form

$$\bigotimes_{i=1}^{d} U_k^{(i)} = \left(\sum_{\alpha_1=0}^{k} \Delta_{\alpha_1}^{(1)} \right) \otimes \cdots \otimes \left(\sum_{\alpha_d=0}^{k} \Delta_{\alpha_d}^{(d)} \right) = \sum_{\alpha_1=0}^{k} \cdots \sum_{\alpha_d=0}^{k} \bigotimes_{i=1}^{d} \Delta_{\alpha_i}^{(i)}$$
$$= \sum_{\substack{|\alpha|_\infty \leq k \\ \alpha \in \mathbb{N}^d}} \bigotimes_{i=1}^{d} \Delta_{\alpha_i}^{(i)}.$$

The rule (1.3) can therefore be considered as a delayed sum of the ordinary tensor product operator (1.2).

Denote the evaluation point set of $U_i^{(j)}$ by $X_i^{(j)}$. Construction of the evaluation points used by $\mathcal{Q}_k^d$ is illustrated in the schematic below.



(a) Product grids $X_{i_1}^{(1)} \times X_{i_2}^{(2)}$ such that $\#X_k^{(j)} = 2^{k-1}$ and $|(i_1, i_2)|_\infty \leq 3$.

(b) The grid corresponding to rule $\mathcal{Q}_4^2$ is the set $\bigcup\{X_{i_1}^{(1)} \times X_{i_2}^{(2)};\ |(i_1, i_2)|_1 \leq 4\}$.

Figure 1.1: Construction of Smolyak quadrature evaluation points. These images were kindly produced for use in this thesis by Janika Kaarnioja.

We shall postpone more involved convergence inspections until the next chapter. We first proceed to explore several useful properties of the Smolyak rule.

Dimension recursion is a quintessential property of the Smolyak rule. In this thesis, we use this term to refer to two closely related results. The most important use of dimension recursion is to allow us to perform the dimension-wise induction step and prove properties of the Smolyak rule.

Proposition 1.5 (Dimension recursion). *Let $k \geq d \geq 2$. Then*

$$\mathcal{Q}_k^d = \sum_{\substack{|\alpha|_1 \leq k-1 \\ \alpha \in \mathbb{N}^{d-1},\ \alpha \geq \mathbf{1}}} \left(\bigotimes_{i=1}^{d-1} \Delta_{\alpha_i}^{(i)} \right) \otimes U_{k-|\alpha|_1}^{(d)} = \sum_{i=d-1}^{k-1} \mathcal{Q}_i^{d-1} \otimes \Delta_{k-i}^{(d)}.$$

Proof. We establish notation for the summation index set of rule (1.3) by setting

$$\mathscr{I}(k,d) = \{\alpha \in \mathbb{N}^d;\ |\alpha|_1 \leq k \text{ and } \alpha \geq \mathbf{1}\}.$$

It is easy to check that the following recursion relation is valid for $k \geq d \geq 2$:

$$\mathscr{I}(k,d) = \{(\alpha, j) \in \mathbb{N}^d;\ \alpha \in \mathscr{I}(k-1, d-1) \text{ and } 1 \leq j \leq k - |\alpha|_1\}.$$

The first equality can now be proven by writing the summation index set of (1.3) recursively and utilizing the distributive property of the tensor product. In this way we attain

$$\begin{aligned}
\mathcal{Q}_k^d &= \sum_{\substack{|\alpha|_1 \leq k-1 \\ \alpha \in \mathbb{N}^{d-1},\ \alpha \geq \mathbf{1}}} \sum_{j=1}^{k-|\alpha|_1} \left(\bigotimes_{i=1}^{d-1} \Delta_{\alpha_i}^{(i)} \right) \otimes \Delta_j^{(d)} \\
&= \sum_{\substack{|\alpha|_1 \leq k-1 \\ \alpha \in \mathbb{N}^{d-1},\ \alpha \geq \mathbf{1}}} \left(\bigotimes_{i=1}^{d-1} \Delta_{\alpha_i}^{(i)} \right) \otimes \sum_{j=1}^{k-|\alpha|_1} \Delta_j^{(d)} \\
&= \sum_{\substack{|\alpha|_1 \leq k-1 \\ \alpha \in \mathbb{N}^{d-1},\ \alpha \geq \mathbf{1}}} \left(\bigotimes_{i=1}^{d-1} \Delta_{\alpha_i}^{(i)} \right) \otimes U_{k-|\alpha|_1}^{(d)},
\end{aligned}$$

where the last equality follows from remark 3.

Continuing where we left off, we can further compute

$$\begin{aligned}
\mathcal{Q}_k^d &= \sum_{\substack{|\alpha|_1 \le k-1 \\ \alpha \in \mathbb{N}^{d-1},\ \alpha \ge \mathbf{1}}} \left(\bigotimes_{i=1}^{d-1} \Delta_{\alpha_i}^{(i)} \right) \otimes U_{k-|\alpha|_1}^{(d)} \\
&= \sum_{j=d-1}^{k-1} \sum_{\substack{|\alpha|_1 = j \\ \alpha \in \mathbb{N}^{d-1},\ \alpha \ge \mathbf{1}}} \left(\bigotimes_{i=1}^{d-1} \Delta_{\alpha_i}^{(i)} \right) \otimes U_{k-j}^{(d)} \\
&= \sum_{j=d-1}^{k-1} \sum_{\substack{|\alpha|_1 = j \\ \alpha \in \mathbb{N}^{d-1},\ \alpha \ge \mathbf{1}}} \left(\bigotimes_{i=1}^{d-1} \Delta_{\alpha_i}^{(i)} \right) \otimes \sum_{\ell=j}^{k-1} \Delta_{k-\ell}^{(d)} \\
&= \sum_{j=d-1}^{k-1} \sum_{\ell=j}^{k-1} \sum_{\substack{|\alpha|_1 = j \\ \alpha \in \mathbb{N}^{d-1},\ \alpha \ge \mathbf{1}}} \left(\bigotimes_{i=1}^{d-1} \Delta_{\alpha_i}^{(i)} \right) \otimes \Delta_{k-\ell}^{(d)}.
\end{aligned}$$

Changing the order of the first two summation signs nets us

$$\mathcal{Q}_k^d = \sum_{\ell=d-1}^{k-1} \left(\sum_{j=d-1}^{\ell} \sum_{\substack{|\alpha|_1 = j \\ \alpha \in \mathbb{N}^{d-1},\ \alpha \ge \mathbf{1}}} \bigotimes_{i=1}^{d} \Delta_{\alpha_i}^{(i)} \right) \otimes \Delta_{k-\ell}^{(d)} = \sum_{\ell=d-1}^{k-1} \mathcal{Q}_\ell^{d-1} \otimes \Delta_{k-\ell}^{(d)}$$

proving the claim. □

The form in which we presented the Smolyak rule seems to imply that in practical usage, a great amount of cancellation of terms is bound to occur due to the presence of difference operators. The following example sheds light on this issue

Example 1.6. Given univariate rules $U_1^{(j)}$, $U_2^{(j)}$, and $U_3^{(j)}$, determine $\mathcal{Q}_5^3$.

We suppress the spatial direction by writing $U_i = U_i^{(j)}$ for the univariate rules and $\Delta_i = \Delta_i^{(j)}$ for the respective difference operators. The collection of all multi-indices $\alpha \in \mathbb{N}^3$ with $\alpha \ge \mathbf{1}$ and $|\alpha|_1 \le 5$ is

$$\begin{aligned}
\{&(1,1,1),(2,1,1),(1,2,1),(1,1,2),(2,2,1),\\
&(2,1,2),(1,2,2),(3,1,1),(1,3,1),(1,1,3)\}.
\end{aligned}$$

Plugging these into the rule (1.3), we get

$$\begin{aligned}
\mathcal{Q}_5^3 = {}& \Delta_1 \otimes \Delta_1 \otimes \Delta_1 + \Delta_2 \otimes \Delta_1 \otimes \Delta_1 + \Delta_1 \otimes \Delta_2 \otimes \Delta_1 + \Delta_1 \otimes \Delta_1 \otimes \Delta_2 \\
&+ \Delta_2 \otimes \Delta_2 \otimes \Delta_1 + \Delta_2 \otimes \Delta_1 \otimes \Delta_2 + \Delta_1 \otimes \Delta_2 \otimes \Delta_2 + \Delta_3 \otimes \Delta_1 \otimes \Delta_1 \\
&+ \Delta_1 \otimes \Delta_3 \otimes \Delta_1 + \Delta_1 \otimes \Delta_1 \otimes \Delta_3
\end{aligned}$$

$$
\begin{aligned}
&= U_1 \otimes U_1 \otimes U_1 + U_2 \otimes U_1 \otimes U_1 - U_1 \otimes U_1 \otimes U_1 + U_1 \otimes U_2 \otimes U_1 \\
&\quad - U_1 \otimes U_1 \otimes U_1 + U_1 \otimes U_1 \otimes U_2 - U_1 \otimes U_1 \otimes U_1 + U_2 \otimes U_2 \otimes U_1 \\
&\quad - U_1 \otimes U_2 \otimes U_1 - U_2 \otimes U_1 \otimes U_1 + U_1 \otimes U_1 \otimes U_1 + U_2 \otimes U_1 \otimes U_2 \\
&\quad - U_1 \otimes U_1 \otimes U_2 - U_2 \otimes U_1 \otimes U_1 + U_1 \otimes U_1 \otimes U_1 + U_1 \otimes U_2 \otimes U_2 \\
&\quad - U_1 \otimes U_1 \otimes U_2 - U_1 \otimes U_2 \otimes U_1 + U_1 \otimes U_1 \otimes U_1 + U_3 \otimes U_1 \otimes U_1 \\
&\quad - U_2 \otimes U_1 \otimes U_1 + U_1 \otimes U_3 \otimes U_1 - U_1 \otimes U_2 \otimes U_1 + U_1 \otimes U_1 \otimes U_3 \\
&\quad - U_1 \otimes U_1 \otimes U_2 \\
&= U_1 \otimes U_1 \otimes U_1 - 2U_2 \otimes U_1 \otimes U_1 - 2U_1 \otimes U_2 \otimes U_1 - 2U_1 \otimes U_1 \otimes U_2 \\
&\quad + U_2 \otimes U_2 \otimes U_1 + U_2 \otimes U_1 \otimes U_2 + U_1 \otimes U_2 \otimes U_2 + U_3 \otimes U_1 \otimes U_1 \\
&\quad + U_1 \otimes U_3 \otimes U_1 + U_1 \otimes U_1 \otimes U_3.
\end{aligned}
$$

The example above poses the question, whether there exists an alternative expression of the rule (1.3) that does away with the difference operators. The answer to this question is positive and we shall derive the so-called *combination method* in the form presented by Wasilkowski and Woźniakowski in [28].

In order to achieve such a representation, we first need to understand the behaviour of difference operators in tensor product operations. This is the leeway that allows us to ultimately gain control over the cancellation of duplicate terms. To this end, we present the following result.

Proposition 1.7. *Let $\alpha \in \mathbb{N}^d$ and $\alpha \geq \mathbf{1}$. Then*

$$\bigotimes_{i=1}^{d} \Delta_{\alpha_i}^{(i)} = \sum_{\substack{\gamma \in \{0,1\}^d \\ \alpha - \gamma \geq \mathbf{1}}} (-1)^{|\gamma|_1} \bigotimes_{i=1}^{d} U_{\alpha_i - \gamma_i}^{(i)}.$$

Proof. A natural approach to show this proposition is to apply dimension-wise induction. In the elementary case we need only to verify the two possible cases:

$$
\begin{aligned}
\Delta_1^{(1)} &= U_1^{(1)} = (-1)^0 U_{1-0}^{(1)}; \\
\Delta_i^{(1)} &= U_i^{(1)} - U_{i-1}^{(1)} = (-1)^0 U_{i-0}^{(1)} + (-1)^1 U_{i-1}^{(1)}, \quad i \geq 2.
\end{aligned}
$$

Next we suppose that the claim holds for some $d \geq 1$. Let $\alpha \in \mathbb{N}^{d+1}$ and $\alpha \geq \mathbf{1}$. If we first assume that $\alpha_{d+1} \neq 1$, then we get by direct computation

$$\begin{aligned}\sum_{\substack{\gamma\in\{0,1\}^{d+1}\\ \alpha-\gamma\geq\mathbf{1}}}(-1)^{|\gamma|_1}\bigotimes_{i=1}^{d+1}U^{(i)}_{\alpha_i-\gamma_i} &= \sum_{\substack{\gamma\in\{0,1\}^{d}\\ \alpha-\gamma\geq\mathbf{1}}}(-1)^{|\gamma|_1+0}\left(\bigotimes_{i=1}^{d}U^{(i)}_{\alpha_i-\gamma_i}\right)\otimes U^{(d+1)}_{\alpha_{d+1}-0}\\ &\quad+\sum_{\substack{\gamma\in\{0,1\}^{d}\\ \alpha-\gamma\geq\mathbf{1}}}(-1)^{|\gamma|_1+1}\left(\bigotimes_{i=1}^{d}U^{(i)}_{\alpha_i-\gamma_i}\right)\otimes U^{(d+1)}_{\alpha_{d+1}-1}\\ &= \sum_{\substack{\gamma\in\{0,1\}^{d}\\ \alpha-\gamma\geq\mathbf{1}}}(-1)^{|\gamma|_1}\left(\bigotimes_{i=1}^{d}U^{(i)}_{\alpha_i-\gamma_i}\right)\otimes \Delta^{(d+1)}_{\alpha_{d+1}}\end{aligned}$$

since $\Delta^{(d+1)}_{\alpha_{d+1}} = U^{(d+1)}_{\alpha_{d+1}} - U^{(d+1)}_{\alpha_{d+1}-1}$. The induction hypothesis implies that

$$\sum_{\substack{\gamma\in\{0,1\}^{d+1}\\ \alpha-\gamma\geq\mathbf{1}}}(-1)^{|\gamma|_1}\bigotimes_{i=1}^{d+1}U^{(i)}_{\alpha_i-\gamma_i} = \left(\bigotimes_{i=1}^{d}\Delta^{(i)}_{\alpha_i}\right)\otimes\Delta^{(d+1)}_{\alpha_{d+1}} = \bigotimes_{i=1}^{d+1}\Delta^{(i)}_{\alpha_i}.$$

If $\alpha_{d+1} = 1$, then we substitute $U^{(d+1)}_{\alpha_{d+1}-1} = 0$ in the computations above and arrive at the same conclusion. This proves the claim. □

The previous proposition immediately yields a rudimentary representation of the rule (1.3) sans difference operators:

$$\mathcal{Q}^d_k = \sum_{\substack{|\alpha|_1\leq k\\ \alpha\in\mathbb{N}^d}}\ \sum_{\substack{\gamma\in\{0,1\}^{d}\\ \alpha-\gamma\geq\mathbf{1}}}(-1)^{|\gamma|_1}\bigotimes_{i=1}^{d}U^{(i)}_{\alpha_i-\gamma_i}.$$

We can freely change the order of summation since both summation sets in the expression above are finite:

$$\mathcal{Q}^d_k = \sum_{\gamma\in\{0,1\}^d}\ \sum_{\substack{|\alpha|_1\leq k\\ \alpha\in\mathbb{N}^d,\ \alpha-\gamma\geq\mathbf{1}}}(-1)^{|\gamma|_1}\bigotimes_{i=1}^{d}U^{(i)}_{\alpha_i-\gamma_i}.$$

This allows us to replace the summation variable α by $\beta = \alpha - \gamma$ with the conditions $\beta \geq \mathbf{1}$ and $|\beta|_1 \leq k - |\gamma|_1$. Since the latter condition also imposes the upper bound $|\beta|_1 \leq |\beta|_1 + |\gamma|_1 \leq k$ to the range of multi-indices β, we can change the order of summation one last time to arrive at the formula

$$\mathcal{Q}^d_k = \sum_{\substack{|\beta|_1\leq k\\ \beta\in\mathbb{N}^d,\ \beta\geq\mathbf{1}}}\ \sum_{\substack{\gamma\in\{0,1\}^d\\ |\gamma|_1\leq k-|\beta|_1}}(-1)^{|\gamma|_1}\bigotimes_{i=1}^{d}U^{(i)}_{\beta_i}.$$

In the expression above the second summation sign actually controls the cancellation of duplicate terms, which we are now free to manipulate. We refer to theorem B.3i) in the appendix to find that

$$\begin{aligned}\sum_{\substack{\gamma\in\{0,1\}^d\\|\gamma|_1\le k-|\beta|_1}}(-1)^{|\gamma|_1} &= \sum_{i=0}^{\min\{d,k-|\beta|_1\}}(-1)^i\sum_{\substack{\gamma\in\{0,1\}^d\\|\gamma|_1=i}}1\\ &= \sum_{i=0}^{\min\{d,k-|\beta|_1\}}(-1)^i\#\{\gamma\in\{0,1\}^d;\ |\gamma|_1=i\}\\ &= \sum_{i=0}^{\min\{d,k-|\beta|_1\}}(-1)^i\binom{d}{i}.\end{aligned}$$

The term above vanishes whenever $d\le k-|\beta|_1$ so we can discard these multi-indices. Recalling that $\beta\ge\mathbf{1}$ we can assume that $|\beta|_1\ge\max\{d,k-d+1\}$. Using equation (B.1) from appendix B yields

$$\sum_{\substack{\gamma\in\{0,1\}^d\\|\gamma|_1\le k-|\beta|_1}}(-1)^{|\gamma|_1}=(-1)^{k-|\beta|_1}\binom{d-1}{k-|\beta|_1}.$$

Explicitly writing down the discussion above yields the sought-after result.

Characterization 1.8 (Combination method)**.** Let $U_i^{(j)}$ be univariate quadrature rules in the interval $\varnothing\ne I_j\subseteq\mathbb{R}$ and suppose that $k\ge d$. Then

$$\mathcal{Q}_k^d=\sum_{\substack{\max\{d,k-d+1\}\le|\alpha|_1\le k\\ \alpha\in\mathbb{N}^d,\ \alpha\ge\mathbf{1}}}(-1)^{k-|\alpha|_1}\binom{d-1}{k-|\alpha|_1}\bigotimes_{i=1}^d U_{\alpha_i}^{(i)}. \tag{1.4}$$

Looking back at example 1.6 where we determined $\mathcal{Q}_5^3$ from the rules U_1, U_2 and U_3, observe that the solution according to (1.4) is

$$\begin{aligned}\mathcal{Q}_5^3=&\sum_{\substack{3\le|\alpha|_1\le5\\ \alpha\in\mathbb{N}^3,\ \alpha\ge\mathbf{1}}}(-1)^{5-|\alpha|_1}\binom{2}{5-|\alpha|_1}U_{\alpha_1}\otimes U_{\alpha_2}\otimes U_{\alpha_3}\\ &=(-1)^2\binom{2}{2}U_1\otimes U_1\otimes U_1\\ &\quad+(-1)^1\binom{2}{1}(U_2\otimes U_1\otimes U_1+U_1\otimes U_2\otimes U_1+U_1\otimes U_1\otimes U_2)\\ &\quad+(-1)^0\binom{2}{0}(U_2\otimes U_2\otimes U_1+U_2\otimes U_1\otimes U_2+U_1\otimes U_2\otimes U_2\\ &\qquad+U_3\otimes U_1\otimes U_1+U_1\otimes U_3\otimes U_1+U_1\otimes U_1\otimes U_3)\end{aligned}$$

$$\begin{aligned}&= U_1 \otimes U_1 \otimes U_1 - 2U_2 \otimes U_1 \otimes U_1 - 2U_1 \otimes U_2 \otimes U_1 - 2U_1 \otimes U_1 \otimes U_2 \\ &\quad + U_2 \otimes U_2 \otimes U_1 + U_2 \otimes U_1 \otimes U_2 + U_1 \otimes U_2 \otimes U_2 + U_3 \otimes U_1 \otimes U_1 \\ &\quad + U_1 \otimes U_3 \otimes U_1 + U_1 \otimes U_1 \otimes U_3,\end{aligned}$$

which is precisely the same result we achieved in example 1.6 with less tedium!

An immediate consequence of the formula (1.4) is the way it renders the evaluation point sets of the Smolyak rule explicitly known. Let $U_i^{(j)}$ be univariate quadrature rules in I_j and let $X_i^{(j)}$ be point sets containing the respective quadrature rules' evaluation points. Then by (1.4) the evaluation points of $\mathcal{Q}_k^d$ form the set

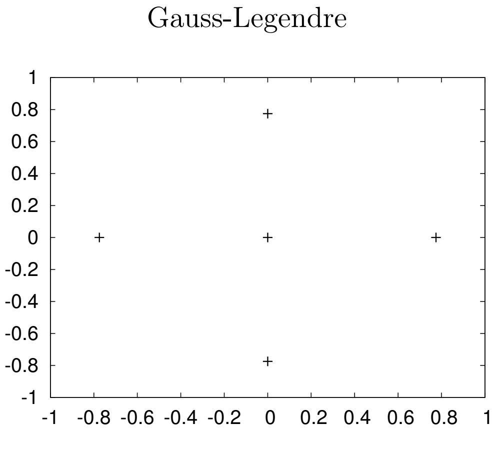
$$\eta(k,d) = \bigcup_{\substack{\max\{d,k-d+1\} \le |\alpha|_1 \le k \\ \alpha \in \mathbb{N}^d,\ \alpha \ge \mathbf{1}}} X_{\alpha_1}^{(1)} \times \cdots \times X_{\alpha_d}^{(d)} \quad \text{for all } k \ge d.$$

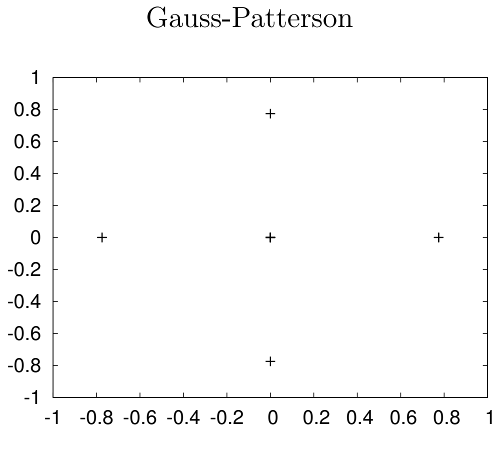
We call elements of the set $\eta(k,d)$ the *nodes* of $\mathcal{Q}_k^d$. The cardinality of the set $\eta(k,d)$ is also called the *cost* of $\mathcal{Q}_k^d$ since it determines the minimum number of function evaluations required to compute the Smolyak rule.

Denote the number of evaluation points of $U_i^{(j)}$ by $n_i^{(j)}$ and suppose that $n_i^{(j)} \le n_{i+1}^{(j)}$. If the univariate rules are nested, i.e. $X_i^{(j)} \subseteq X_{i+1}^{(j)}$, then the nodes of $\mathcal{Q}_k^d$ form the set

$$\eta(k,d) = \bigcup_{\substack{|\alpha|_1 = k \\ \alpha \in \mathbb{N}^d,\ \alpha \ge \mathbf{1}}} X_{\alpha_1}^{(1)} \times \cdots \times X_{\alpha_d}^{(d)} \quad \text{for all } k \ge d.$$

Example 1.9. We compare the evolution of $\eta(k,2)$, when the univariate rules are chosen to be the non-nested Gauss-Legendre and nested Gauss-Patterson quadrature rules of 1, 3, 7, 15 and 31 points in $[-1,1]$. The nested Gauss-Patterson rule manages to yield a more economical point set as hinted by the preceding discussion.





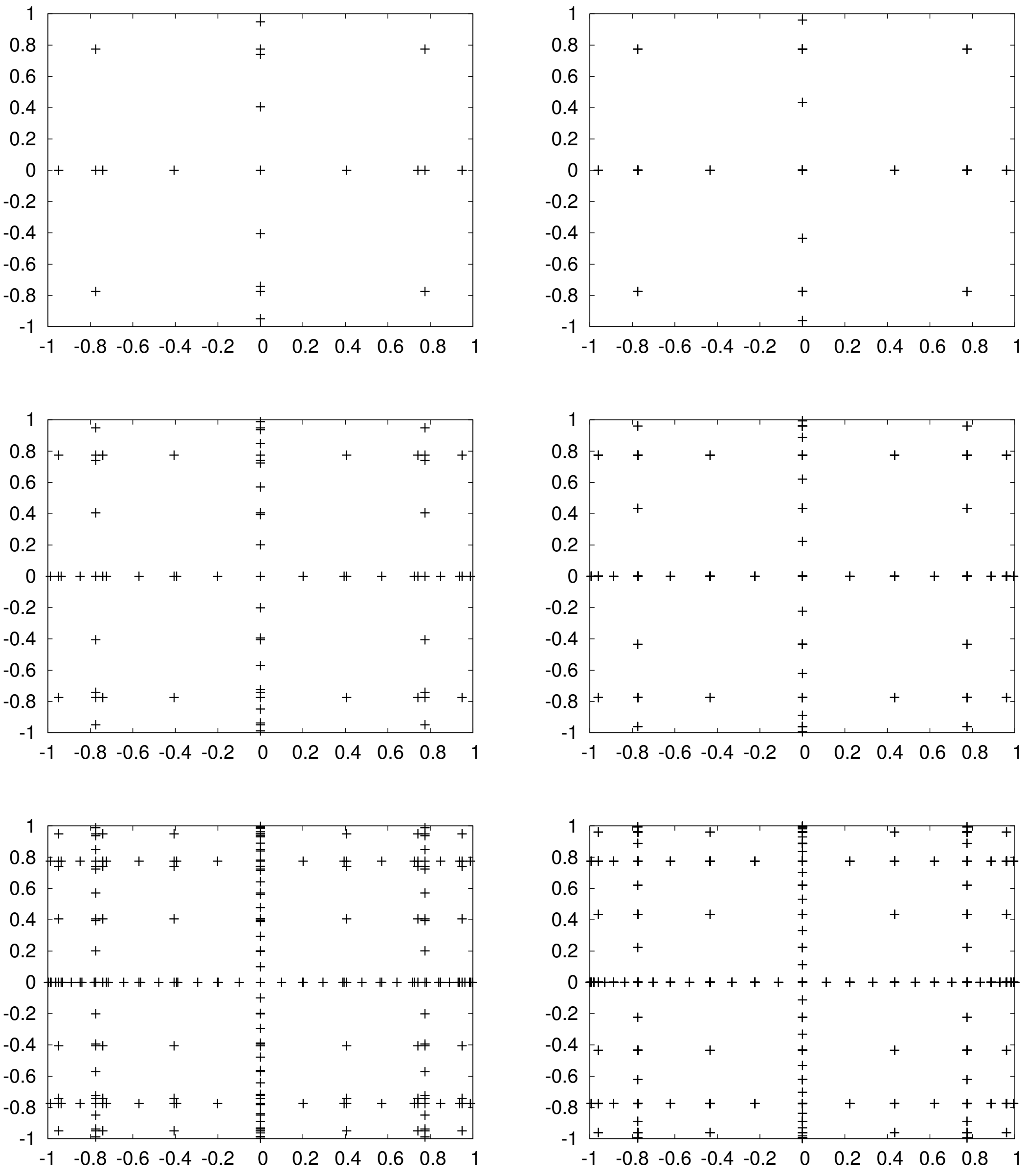


Figure 1.2: The evolution of $\eta(k, 2)$ for $k = 3, 4, 5, 6$.

The selection of basis sequence affects both the accuracy and cost of Smolyak quadrature. In the numerical sections of this thesis we shall inspect three kinds of choices for the basis sequence:

i) slowly increasing, non-nested Gauss-Legendre and Gauss-Hermite sequences $U_i^{(j)}$ with the cardinality $n_i^{(j)} = i$;

ii) rapidly increasing, nested Clenshaw-Curtis sequence $U_i^{(j)}$ with the cardinality $n_1^{(j)} = 1$ and $n_i^{(j)} = 2^{i-1} + 1$ for $i > 1$. This sequence is recommended by Novak and Ritter in [19];

iii) delayed, nested Gauss-Patterson and Genz-Keister rules. The construction of these basis sequences is described below.

Let $U_i = U_i^{(j)}$ be a slowly increasing Gauss-Patterson or Genz-Keister basis sequence with n_i evaluation points and polynomial exactness m_i. Holtz [17] recommends the following construction for a delayed basis sequence $\widetilde{U}_i$ with $\widetilde{n}_i$ points and polynomial exactness $\widetilde{m}_i$:

```
$i = 1$;
$\ell = 0$;
**repeat**
    $\ell = \ell + 1$;
    **while** $m_\ell \geq 2i - 1$ **do**
        $\widetilde{U}_i^{(j)} = U_\ell^{(j)}$;
        $i = i + 1$;
    **end**
**until** a basis sequence of desired length is achieved.
```

This procedure yields the following sequences:

i	1	2	3	4	5	6	7	$\cdots$	12	13	$\cdots$	24
$\widetilde{n}_i$	1	3	3	7	7	7	15	$\cdots$	15	31	$\cdots$	31
$\widetilde{m}_i$	1	5	5	11	11	11	23	$\cdots$	23	47	$\cdots$	47
$2i-1$	1	3	5	7	9	11	13	$\cdots$	23	25	$\cdots$	47

Table 1.1: The delayed Gauss-Patterson sequence.

i	1	2	3	4	$\cdots$	8	9	$\cdots$	15	16	$\cdots$	32
$\widetilde{n}_i$	1	3	3	9	$\cdots$	9	19	$\cdots$	19	41	$\cdots$	41
$\widetilde{m}_i$	1	5	5	15	$\cdots$	15	29	$\cdots$	29	63	$\cdots$	63
$2i-1$	1	3	5	7	$\cdots$	15	17	$\cdots$	29	31	$\cdots$	63

Table 1.2: The delayed Genz-Keister sequence.

The concept of delaying should be interpreted in the following sense: after a certain number of repetitions, the Smolyak rule with a delayed Gauss-Patterson or Genz-Keister basis sequence becomes equal in accuracy with their slowly increasing Gaussian counterpart and no further improvements are possible without passing higher order rules to the Smolyak construction. Observe that in example 1.9 we used a slowly increasing Gauss-Patterson basis sequence instead of a delayed one.

We end this section with an elementary result regarding the cost of $\mathcal{Q}_k^d$, when the basis sequence $U_i^{(j)}$ has $n_i^{(j)} = 2^{i-1}$ evaluation points. This sequence has special interest to us since it will be used in the error analysis of the Smolyak rule.

Proposition 1.10. *Let $U_i^{(j)}$ be univariate quadrature rules with $n_i^{(j)} = 2^{i-1}$ nodes and $k \geq d \geq 1$. The cost of $\mathcal{Q}_k^d$ is*

$$\#\eta(k,d) = \sum_{i=\max\{d,k-d+1\}}^{k} 2^{i-d}\binom{i-1}{d-1}.$$

Proof. Utilizing the combination method we evaluate

$$\begin{aligned}\#\eta(k,d) = \sum_{\substack{\max\{d,k-d+1\}\leq|\alpha|_1\leq k\\ \alpha\in\mathbb{N}^d,\ \alpha\geq\mathbf{1}}} n_{\alpha_1}^{(1)}\cdots n_{\alpha_d}^{(d)} &= \sum_{i=\max\{d,k-d+1\}}^{k}\ \sum_{\substack{|\alpha|_1=i\\ \alpha\in\mathbb{N}^d,\ \alpha\geq\mathbf{1}}} 2^{|\alpha|_1-d}\\ &= \sum_{i=\max\{d,k-d+1\}}^{k} 2^{i-d} \sum_{\substack{|\alpha|_1=i\\ \alpha\in\mathbb{N}^d,\ \alpha\geq\mathbf{1}}} 1.\end{aligned}$$

The latter sum is equal to $\#\{\alpha\in\mathbb{N}^d;\ |\alpha|_1 = i \text{ and } \alpha\geq\mathbf{1}\}$, which is precisely $\binom{i-1}{d-1}$ according to theorem B.3ii). This is the desired result. $\square$

Remark 4. This result is actually a modified version of a result concerning the topology of sparse grids, the theory of which is explored in detail by Bungartz and Griebel in [5].

1.3 Numerical implementation

In this section we formulate the numerical implementation of the Smolyak rule for evaluation of integrals of the form

$$\mathcal{I}_W^d f = \int_{I^d} W(x_1,...,x_d)f(x_1,...,x_d)\,\mathrm{d}x_d\cdots\mathrm{d}x_1.$$

For simplicity we have assumed the integration region to be the hypercube $I^d = I\times\cdots\times I$, where $\varnothing \neq I \subseteq \mathbb{R}$ is an interval. It is straightforward to generalize the discussion in this section to hyperrectangles. The weight function has the form $W(x_1, ..., x_d) = W_1(x_1)^d$. The univariate rule is chosen in such a way that the contribution of W_1 is eliminated altogether.

The problem formulation is the following: given the function f and univariate rules U_j with the nodes $(x_i^{(j)})_{i=1}^{n_j}$ and weights $(w_i^{(j)})_{i=1}^{n_j}$, compute

$$\sum_{\ell=\max\{d,k-d+1\}}^{k} \sum_{\substack{|\alpha|_1=\ell \\ \alpha\in\mathbb{N}^d,\ \alpha\geq\mathbf{1}}} \sum_{i_1=1}^{n_{\alpha_1}} \cdots \sum_{i_d=1}^{n_{\alpha_d}} \operatorname{coef}(k,d,\ell) w_{i_1}^{(\alpha_1)} \cdots\ w_{i_d}^{(\alpha_d)} f(x_{i_1}^{(\alpha_1)}, ..., x_{i_d}^{(\alpha_d)}),$$

where $\operatorname{coef}(k, d, \ell) = (-1)^{k-\ell}\binom{d-1}{k-\ell}$. The above is just the combination method (1.4) written down explicitly. Observe that the cumbersome nested sums can be replaced by summation over all occurring combinations of univariate nodes and weights.

Before we can construct the algorithm, we need to account for the following components:

i) a generator of univariate nodes and weights;

ii) a combinatorial algorithm that generates all multi-indices $\alpha \in \mathbb{N}^d$ such that $\alpha \geq \mathbf{1}$ and $|\alpha|_1 = \ell$, $\ell \geq d$;

iii) for any vector sequence $(v^{(i)})_{i=1}^{\ell}$, $v^{(i)} = (v_1^{(i)}, ..., v_{n_i}^{(i)}) \in \mathbb{R}^{n_i}$, determine its *vector combination* defined inductively by

$$\operatorname{combvec}((v^{(i)})_{i=1}^{1}) = v^{(1)};$$

$$\operatorname{combvec}((v^{(i)})_{i=1}^{\ell}) = \begin{pmatrix} \operatorname{combvec}((v^{(i)})_{i=1}^{\ell-1}) & \cdots & \operatorname{combvec}((v^{(i)})_{i=1}^{\ell-1}) \\ v_1^{(\ell)} \ \cdots \ v_1^{(\ell)} & \cdots & v_{n_i}^{(\ell)} \ \cdots \ v_{n_i}^{(\ell)} \end{pmatrix},$$

where the notation on the bottom row of the matrix means that each individual vector element of $v^{(\ell)}$ is repeated to match the width of the previous vector combination iteration.

For example, the vector combination of $v^{(1)} = (1)$, $v^{(2)} = (2, 3)$ and $v^{(3)} = (4, 5, 6)$ is the matrix

$$\operatorname{combvec}((v^{(i)})_{i=1}^{3}) = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 3 & 2 & 3 & 2 & 3 \\ 4 & 4 & 5 & 5 & 6 & 6 \end{pmatrix}.$$

We address parts i)-iii):

i) We assume that the nodes and weights of the univariate quadrature rules are provided on the part of the user. Suggestions for the generation of univariate rules are given in appendix A.

ii) We use the drop algorithm by Thomas Gerstner [11, algorithm 8.1.1].

Algorithm 1.11.

Generate all d-dimensional multi-indices $(k_1, ..., k_d)$ with $k_1 + ... + k_d = \ell$.

```
Input: dimension $d$, order $\ell$
Output: matrix $ind$ containing multi-indices, $count$ containing its size
$p = 1$;
$count = 0$;
initialize matrix $ind$;
for $i = 1, ..., d$ do
    $k_i = 1$;
    $\hat{k}_i = \ell - d + 1$;
end
while $k_d < \ell - d + 1$ do
    $k_p = k_p + 1$;
    if $k_p > \hat{k}_p$ then
        $k_p = 1$;
        $p = p + 1$;
    else
        for $i = 1, ..., d$ do
            $\hat{k}_i = \hat{k}_p - k_p + 1$;
        end
        $k_1 = \hat{k}_1$;
        $p = 1$;
        $count = count + 1$;
        append matrix $ind$ with $(k_1, ..., k_d)$;
    end
end
```

See program C.2 in appendix C for the MATLAB implementation of this algorithm.

iii) The MATLAB Neural Network Toolbox function `combvec` works as advertised above when the input is given as *row* vectors.[†]

For notational simplicity, we define an auxiliary notation. Let A be the $k \times n$ matrix defined by $A = (a_{ij})$, $1 \leq i \leq k$ and $1 \leq j \leq n$, and define the

[†]If the Neural Network Toolbox is not installed, I have included the program C.7 in appendix C with identical functionality in this context.

column-wise matrix product with itself by setting

$$\prod_{\text{column-wise}} A = \left(\prod_{i=1}^{k} a_{ij}\right)_{j=1}^{n}.$$

With these requisites, we are ready to write down the algorithm for the generation of Smolyak quadrature nodes and weights.

Algorithm 1.12 (Smolyak quadrature rule)**.**

Generate the d-dimensional Smolyak quadrature rule of order k.

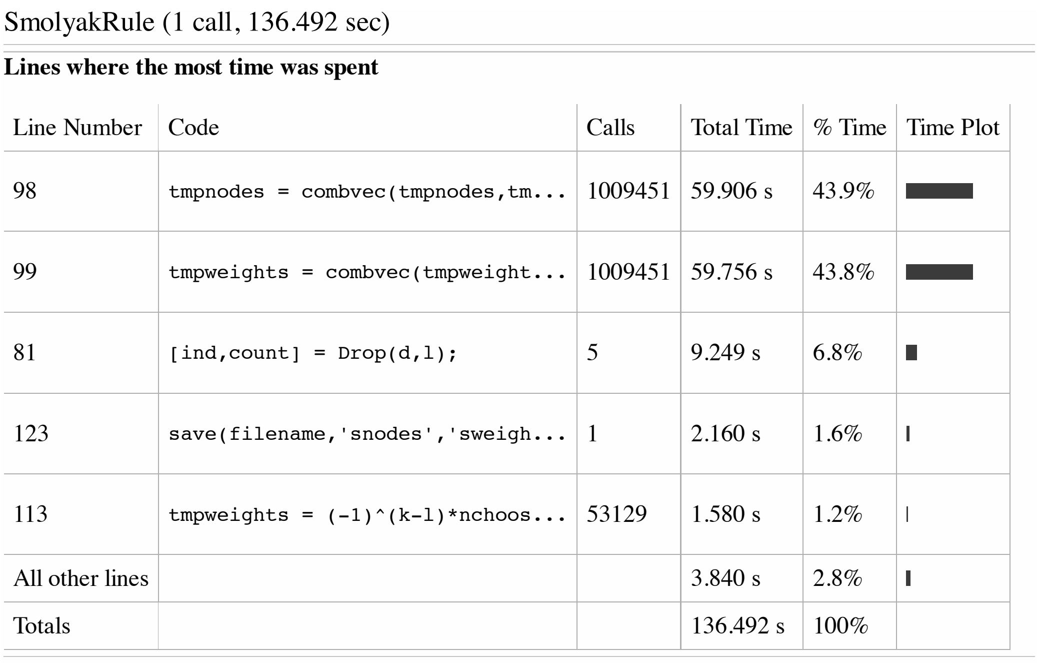
```
Input: dimension $d$, order $k$
Output: matrix snodes containing the Smolyak quadrature nodes,
        vector sweights containing the respective weights
initialize matrix snodes;
initialize vector sweights;
for $\ell = \max\{d, k-d+1\}, ..., k$ do
    generate ind with dimension $d$ and order $\ell$ using algorithm 1.11;
    for each multi-index $\alpha$ in ind do
        determine univariate nodes $x^{(\alpha_j)} = (x_1^{(\alpha_j)}, ..., x_{n_{\alpha_j}}^{(\alpha_j)})$ and weights
        $w^{(\alpha_j)} = (w_1^{(\alpha_j)}, ..., w_{n_{\alpha_j}}^{(\alpha_j)})$;
        $x = \text{combvec}((x^{(\alpha_j)})_{j=1}^{d})$;
        $w = \text{coef}(k, d, \ell) \prod_{\text{column-wise}} \text{combvec}((w^{(\alpha_j)})_{j=1}^{d})$;
        append matrix snodes with $x$;
        append vector sweights with $w$;
    end
end
```

See program C.3 for the MATLAB implementation of this algorithm.

The nodes are aligned vertically in the matrix *snodes*. This choice is one of being practical: MATLAB built-in functions operate natively on column vectors. If we want to compute the integral of the function `func` defined in MATLAB environment, we can load the computed Smolyak quadrature nodes `snodes` and weights `sweights` and run

```
>> sum(sweights.*feval(func,snodes))
```

A test run was conducted to determine the Smolyak quadrature rule based on a slowly increasing Gauss-Legendre basis sequence with input parameters $d = 20$ and $k = 25$. This produced the profile report on the next page.



SmolyakRule (1 call, 136.492 sec)

Lines where the most time was spent

Line Number	Code	Calls	Total Time	% Time	Time Plot
98	`tmpnodes = combvec(tmpnodes,tm...`	1009451	59.906 s	43.9%	▬
99	`tmpweights = combvec(tmpweight...`	1009451	59.756 s	43.8%	▬
81	`[ind,count] = Drop(d,l);`	5	9.249 s	6.8%	▪
123	`save(filename,'snodes','sweigh...`	1	2.160 s	1.6%	|
113	`tmpweights = (-1)^(k-l)*nchoos...`	53129	1.580 s	1.2%	|
All other lines			3.840 s	2.8%	|
Totals			136.492 s	100%	

Figure 1.3: The profile report produced by a test run of the program C.3.

The majority of runtime was split evenly between computing the Smolyak quadrature nodes and weights, which is the intent of the program. The auxiliary function `Drop` is not vectorized so its performance contributes notably to the total runtime. However, the increasing complexity of the `combvec` process with high values of d and k diminishes `Drop` by a wide margin.

Numerical experiments using the program C.3 are postponed until sections 2.1 and 2.3 of the next chapter.

Chapter 2

Error analysis

In this chapter we explore the rate at which the Smolyak quadrature rule approximates the integral

$$\mathcal{I}_W^d f = \int\limits_{I_1\times\cdots\times I_d} W(x_1,...,x_d)f(x_1,...,x_d)\,\mathrm{d}x_d\cdots\mathrm{d}x_1, \quad \varnothing \neq I_j \subseteq \mathbb{R} \text{ an interval,}$$

under sufficient smoothness conditions for the integrand f and the weight function W. Our first goal is to quantify the admissible integrands and weight functions.

Let $\Omega = I_1 \times \cdots \times I_d$. In the general case where Ω is allowed to be unbounded, we are mostly limited to inspecting the polynomial exactness of the Smolyak rule. If we assume that Ω is bounded, we can proceed to derive error estimates for functions in the class $H^r(\Omega)$.

The class of weight functions is defined in the following way: $W \in \mathscr{W}(\Omega)$ if and only if W satisfies the following postulates:

W1) $W(x_1,...,x_d) = W_1(x_1)\cdots W_d(x_d)$ for all $(x_1,...,x_d) \in \Omega$ and $W_j \geq 0$ for all $j = 1,...,d$;

W2) the mapping $x \mapsto x^k W_j(x)$ is integrable in I_j for all $k \in \mathbb{N}$ and $j = 1,...,d$;

W3) there exists a sequence of univariate quadrature rules $(U_i^{(j)})_{i=1}^{\infty}$ such that $||\mathcal{I}_{W_j}^1 - U_i^{(j)}|| \leq BC^i$ for some positive constants B and C.

The norm $||\cdot||$ denotes the operator norm in the dual of $H^r(\Omega)$ for the entirety of this chapter.

Define the operator $\mathcal{S}^d \colon H^r(\Omega) \to \mathbb{R}$ by setting

$$\mathcal{S}^d f = \sum_{i=1}^{n_d} w_i f(x_i),$$

where $(w_i)_{i=1}^{n_d}$ is a collection of positive weights and the sequence $(x_i)_{i=1}^{n_d}$ belongs to Ω as usual. We append our definition 1.1 of the tensor product by setting

$$\mathcal{I}_W^{d_1} \otimes \mathcal{I}_V^{d_2} f = \int\limits_{I_{d_1} \times \dots \times I_{d_1+d_2}} W(z_1, ..., z_{d_1}) V(z_{d_1+1}, ..., z_{d_1+d_2}) f(z_1, ..., z_{d_1+d_2}) \, \mathrm{d}z;$$

$$\mathcal{S}^{d_1} \otimes \mathcal{I}_V^{d_2} f = \sum_{i=1}^{n_{d_1}} w_i \int\limits_{I_{d_1+1} \times \dots \times I_{d_1+d_2}} V(z) f(x_i, z) \, \mathrm{d}z;$$

$$\mathcal{I}_W^{d_1} \otimes \mathcal{S}^{d_2} f = \int\limits_{I_1 \times \dots \times I_{d_1}} \sum_{i=1}^{n_{d_2}} w_i W(z) f(z, x_i) \, \mathrm{d}z$$

for $f \in H^r(\Omega) \cap L^1(\Omega)$, $W \in \mathscr{W}(I_1 \times \dots \times I_{d_1})$ and $V \in \mathscr{W}(I_{d_1+1} \times \dots \times I_{d_1+d_2})$, where we take $L^1(\Omega)$ to mean the set of integrable functions in Ω. It turns out that the tensor product theorem 1.2 holds for our appended definition of the tensor product. Checking this is arduous, so we shall only verify the parts which we will need.

Suppose that $||\mathcal{I}_W^{d_1}|| < \infty$ and $||\mathcal{I}_V^{d_2}|| < \infty$. First we note that the new definition retains the associative and distributive properties of definition 1.1. The relation $||\mathcal{I}_W^{d_1} \otimes \mathcal{I}_V^{d_2}|| \leq ||\mathcal{I}_W^{d_1}|| \cdot ||\mathcal{I}_V^{d_2}||$ holds according to Fubini's theorem. It suffices to check the condition $||\mathcal{S}^{d_1} \otimes \mathcal{I}_V^{d_2}|| \leq ||\mathcal{S}^{d_1}|| \cdot ||\mathcal{I}_V^{d_2}||$. The inequality $||\mathcal{I}_W^{d_1} \otimes \mathcal{S}^{d_2}|| \leq ||\mathcal{I}_W^{d_1}|| \cdot ||\mathcal{S}^{d_2}||$ follows in a similar fashion.

Let $\Omega_1 = I_1 \times \dots \times I_{d_1}$, $\Omega_2 = I_{d_1+1} \times \dots \times I_{d_1+d_2}$ and $\Omega = \Omega_1 \times \Omega_2$. Additionally, let $f \in H^r(\Omega)$ and $V \in \mathscr{W}(\Omega_2)$. Fix $x_0 \in \Omega_1$ and $\alpha \in \mathbb{N}^{d_1}$, $|\alpha|_\infty \leq r$. Define the functions $g \colon \Omega_1 \to \mathbb{R}$ and $h \colon \Omega_2 \to \mathbb{R}$ by setting

$$g(x) = \int\limits_{\Omega_2} V(z) f(x, z) \, \mathrm{d}z \quad \text{and} \quad h(x) = \left. \frac{\partial^\alpha f(z, x)}{\partial z^\alpha} \right|_{z=x_0}.$$

It immediately follows that $g \in H^r(\Omega_1)$, $h \in H^r(\Omega_2)$ and $||h||_{H^r(\Omega_2)} \leq ||f||_{H^r(\Omega)}$ hold regardless of x_0 and α. The rest of the proof is completely analogous to the first part of the proof of theorem 1.2. The key observations are

$$||g||_{H^r(\Omega_1)} \leq ||\mathcal{I}_V^{d_2}|| \cdot ||h||_{H^r(\Omega_2)} \quad \text{and} \quad \mathcal{S}^{d_1} g = \mathcal{S}^{d_1} \otimes \mathcal{I}_V^{d_2} f.$$

Combining the above yields

$$|\mathcal{S}^{d_1} \otimes \mathcal{I}_V^{d_2} f| = |\mathcal{S}^{d_1} g| \leq ||\mathcal{S}^{d_1}|| \cdot ||g||_{H^r(\Omega_1)} \leq ||\mathcal{S}^{d_1}|| \cdot ||\mathcal{I}_V^{d_2}|| \cdot ||f||_{H^r(\Omega)}.$$

Taking the supremum over the set $\{f \in H^r(\Omega);\ ||f||_{H^r(\Omega)} \leq 1\}$ yields the desired result. For details, we refer the reader to the proof of theorem 1.2.

2.1 Polynomial exactness

It is customary to compare the exactness of univariate quadrature rules by their polynomial exactness. While not the best universal measure of exactness outside the realm of polynomials (for exploration on this, see e.g. [26]), one can infer from considerable polynomial exactness of the integration rule that high smoothness of the integrand translates to high accuracy of the approximation. In the domain of univariate rules, the highest polynomial exactness that can be achieved using n evaluation points is $2n-1$. These rules are called Gaussian and in this section we shall see that this property translates in some capacity to the Smolyak rule with a Gaussian basis sequence.

Another interesting reason to inspect the polynomial exactness of the Smolyak rule is the generality of the results stated in this section. For example, we do not even have to assume that the intervals I_j be bounded. The results derived in this section hold especially for Gaussian integrals

$$\int\limits_{\mathbb{R}^d} e^{-||x||^2} f(x)\,\mathrm{d}x,$$

which can be approximated using the non-nested Gauss-Hermite or the nested Genz-Keister rule by choosing $W_j(x) = e^{-x^2}$ and $I_j = \mathbb{R}$.

The polynomial exactness of Smolyak quadrature is explored in detail by Novak and Ritter in [20] and partly by Heiss and Winschel in [16]. The results in this section are based on these papers.

First we need to quantify the relevant polynomial spaces. Define the following notation for multivariate monomials:

$$x^\beta = \prod_{i=1}^{d} x_i^{\beta_i},$$

where $x = (x_1, ..., x_d) \in \mathbb{R}^d$ and $\beta \in \mathbb{N}^d$. If $\beta = (0, ..., 0)$, then we define $x^\beta = 1$. We use this notation to define the space of multivariate polynomials

$$\mathbb{P}_k^d = \left\{ \mathbb{R}^d \ni x \mapsto \sum_{\substack{|\beta|_1 \le k \\ \beta \in \mathbb{N}^d}} a_\beta x^\beta \in \mathbb{R};\ a_\beta \in \mathbb{R} \text{ for all } \beta \in \mathbb{N}^d \right\}.$$

If $p(x) = \sum a_\beta x^\beta \in \mathbb{P}_k^d$ is not the zero-mapping, then we define the *total degree* of p by setting $\deg(p) = \max\{|\beta|_1;\ a_\beta \neq 0 \text{ for some } \beta \in \mathbb{N}^d,\ |\beta|_1 \le k\}$. If $p(x) = 0$ for all $x \in \mathbb{R}^d$, then we set $\deg(p) = -\infty$.

We provide the following definition for polynomial tensor product spaces:

$$\bigotimes_{i=1}^{d} \mathbb{P}^1_{m_i} = \left\{ \mathbb{R}^d \ni (x_1, ..., x_d) \mapsto \prod_{i=1}^{d} p_i(x_i) \in \mathbb{R};\ p_i \in \mathbb{P}^1_{m_i} \text{ for } i = 1, ..., d \right\}.$$

Example 2.1. $\mathbb{P}^d_k$ is clearly spanned by the sequence $\left(x^\beta\right)^{\beta\in\mathbb{N}^d}_{0\leq|\beta|_1\leq k}$, which is linearly independent. The cardinality of this sequence is called the *dimension* of $\mathbb{P}^d_k$. We find that

$$\dim(\mathbb{P}^d_k) = \sum_{i=0}^{k} \#\{\beta \in \mathbb{N}^d;\ |\beta|_1 = i\}.$$

Using the combinatorial result B.4 yields

$$\#\{\beta \in \mathbb{N}^d;\ |\beta|_1 = i\} = \binom{d+i-1}{d-1}$$

and utilizing property (B.2) nets us

$$\dim(\mathbb{P}^d_k) = \sum_{i=0}^{k} \binom{d+i-1}{d-1} = \binom{d+k}{d}.$$

As a special case, we verify the well-known identity

$$\dim(\mathbb{P}^1_k) = \binom{1+k}{1} = k+1.$$

Recall that interpolatory quadrature formulae with polynomial exactness m are derived for predetermined points by solving a linear system of $\dim(\mathbb{P}^1_m) = m+1$ equations for $d = 1$. To establish polynomial exactness of total degree m for $d > 1$, we would similarly need to solve a system of $\dim(\mathbb{P}^d_m) = \binom{d+m}{d}$ equations. As one might hanker a guess, this is not the most efficient approach to the problem. Hence the need for multivariate quadrature algorithms.

We are ready to prove the first main result of this section.

Proposition 2.2. *Let $U^{(j)}_i$ be univariate quadrature rules that correspond to the weight W_j and have polynomial exactness m_i such that $m_i \leq m_{i+1}$. Let $W(x_1, ..., x_d) = W_1(x_1) \cdots W_d(x_d)$. Then*

$$\mathcal{I}^d_W f = \mathcal{Q}^d_k f \quad \textit{for all } f \in \sum_{\substack{|\alpha|_1=k \\ \alpha\in\mathbb{N}^d}} \bigotimes_{i=1}^{d} \mathbb{P}^1_{m_{\alpha_i}} \ \textit{and } k \geq d,$$

where we define $A + B = \{a + b;\ a \in A \text{ and } b \in B\}$ for sets A and B.

Proof. The proof is carried out by dimension-wise induction. If $d = 1$, then the claim is reduced into $\mathcal{I}^1_{W_1} f = U^{(1)}_k f$ for $f \in \mathbb{P}^1_{m_k}$. This is true by our assumptions. Suppose that the claim is true for some $d \geq 1$.

Let $\beta \in \mathbb{N}^{d+1}$ such that $|\beta|_1 = k$ and $k \geq d+1$. Define $f(x_1, ..., x_{d+1}) = g(x_1, ..., x_d) f_{d+1}(x_{d+1})$, where $g(x_1, ..., x_d) = f_1(x_1) \cdots f_d(x_d)$ and $f_i \in \mathbb{P}^1_{m_i}$ for $i = 1, ..., d+1$. Now clearly $f \in \bigotimes_{i=1}^{d+1} \mathbb{P}^1_{m_{\beta_i}}$. It is sufficient to prove the claim for the function f since linearity of the Smolyak rule implies that the claim then holds for any element in $\sum_{|\alpha|_1 = k} \bigotimes_{i=1}^{d+1} \mathbb{P}^1_{m_{\alpha_i}}$ as well.

Using dimension recursion and the product structure of f we get

$$\mathcal{Q}^{d+1}_k f = \sum_{i=d}^{k-1} \mathcal{Q}^d_i \otimes \Delta^{(d+1)}_{k-i} f = \sum_{i=d}^{k-1} \mathcal{Q}^d_i g \cdot \Delta^{(d+1)}_{k-i} f_{d+1}.$$

If $\beta_{d+1} \leq k - i - 1$, then $m_{\beta_{d+1}} \leq m_{k-i-1} \leq m_{k-i}$ and we have $U^{(d+1)}_{k-i} f_{d+1} = U^{(d+1)}_{k-i-1} f_{d+1} = \mathcal{I}^1_{W_{d+1}} f_{d+1}$. Especially $\Delta^{(d+1)}_{k-i} f_{d+1} = 0$ and we can trucate the expression for $\mathcal{Q}^{d+1}_k$ by considering summation over the indices $k - \beta_{d+1} \leq i \leq k-1$.

Using the fact that $k = |\beta|_1$ allows us to write the rule $\mathcal{Q}^{d+1}_k$ in the form

$$\mathcal{Q}^{d+1}_k f = \sum_{i=\beta_1+...+\beta_d}^{k-1} \mathcal{Q}^d_i g \cdot \Delta^{(d+1)}_{k-i} f_{d+1}.$$

Our induction hypothesis implies that $\mathcal{I}^d_{W_1 \cdots W_d} g = \mathcal{Q}^d_i g$ for $\beta_1 + ... + \beta_d \leq i \leq k-1$ and we achieve

$$\begin{aligned}
\mathcal{Q}^{d+1}_k f &= \sum_{i=\beta_1+...+\beta_d}^{k-1} \mathcal{I}^d_{W_1 \cdots W_d} g \cdot \Delta^{(d+1)}_{k-i} f_{d+1} \\
&= \mathcal{I}^d_{W_1 \cdots W_d} g \cdot U^{(d+1)}_{k-\beta_1-...-\beta_d} f_{d+1} \\
&= \mathcal{I}^d_{W_1 \cdots W_d} g \cdot U^{(d+1)}_{\beta_{d+1}} f_{d+1} \\
&= \mathcal{I}^d_{W_1 \cdots W_d} g \cdot \mathcal{I}^1_{W_d} f_{d+1} = \mathcal{I}^{d+1}_W f
\end{aligned}$$

proving the claim. $\square$

Combining the Smolyak quadrature rule with the high polynomial exactness of Gaussian rules, we achieve a remarkable result concerning the exactness of the Smolyak rule using a slowly increasing Gaussian basis sequence.

Proposition 2.3. *Let $U_i^{(j)}$ be a slowly increasing basis sequence of Gaussian rules corresponding to the weight function W_j. Let $W(x_1, ..., x_d) = W_1(x_1) \cdots W_d(x_d)$. Then*

$$\mathcal{I}_W^d f = \mathcal{Q}_k^d f \quad \text{for all } f \in \mathbb{P}^d_{2(k-d)+1} \text{ and } k \geq d.$$

Proof. It suffices to prove the claim for monomials. In the case $d = 1$ the claim is reduced into one describing the polynomial exactness of univariate Gaussian rules. We assume that for some $d \geq 1$, it holds that

$$\mathcal{I}^d_{W_1 \cdots W_d} x^\beta = \mathcal{Q}_k^d x^\beta \quad \text{for all } k \geq d,\ \beta \in \mathbb{N}^d \text{ and } |\beta|_1 \leq 2(k-d)+1.$$

Let $\beta \in \mathbb{N}^{d+1}$ be such that $|\beta|_1 \leq 2(k-d-1)+1 = 2(k-d)-1$. Using dimension recursion we can write

$$\mathcal{Q}_k^{d+1} x^\beta = \sum_{i=d}^{k-1} \mathcal{Q}_i^d x^{(\beta_1, ..., \beta_d)} \cdot \Delta_{k-i}^{(d+1)} x^{\beta_{d+1}}.$$

If $\beta_{d+1} \leq 2(k-i-1)-1$, then $U_{k-i}^{(d+1)} x^{\beta_{d+1}} = U_{k-i-1}^{(d+1)} x^{\beta_{d+1}} = \mathcal{I}^1_{W_{d+1}} x^{\beta_{d+1}}$ and hence $\Delta_{k-i}^{(d+1)} x^{\beta_{d+1}} = 0$. We can assume that $\beta_{d+1} \geq 2(k-i-1)$. This implies that $\beta_1 + ... + \beta_d + 2(k-i-1) \leq |\beta|_1$. Our selection of β now yields $\beta_1 + ... + \beta_d + 2(k-i-1) \leq 2(k-d)-1$. In other words $\beta_1 + ... + \beta_d \leq 2(i-d)+1$. According to our induction hypothesis we get

$$\mathcal{Q}_k^{d+1} x^\beta = \mathcal{I}_W^d x^{(\beta_1, ..., \beta_d)} \sum_{i=d}^{k-1} \Delta_{k-i}^{(d+1)} x^{\beta_{d+1}},$$

where $\beta_{d+1} \geq 2(k-i-1)$. If $\beta_{d+1} = 2r$ for some $r \in \mathbb{N}$, then the sum can be written over the indices $k-r-1 \leq i \leq k-1$ and we achieve

$$\begin{aligned}\mathcal{Q}_k^{d+1} x^\beta &= \mathcal{I}^d_{W_1 \cdots W_d} x^{(\beta_1, ..., \beta_d)} \sum_{i=k-r-1}^{k-1} \Delta_{k-i}^{(d+1)} x^{\beta_{d+1}} \\ &= \mathcal{I}^d_{W_1 \cdots W_d} x^{(\beta_1, ..., \beta_d)} \cdot U_{r+1}^{(d+1)} x^{2r} = \mathcal{I}^d_{W_1 \cdots W_d} x^{(\beta_1, ..., \beta_d)} \cdot \mathcal{I}^1_{W_{d+1}} x^{\beta_{d+1}}\end{aligned}$$

since $2(r+1)-1 = 2r+1 \geq \beta_{d+1}$. On the other hand, if $\beta_{d+1} = 2r+1$ for some $r \in \mathbb{N}$, then we consider the indices $i \in \mathbb{N}$ with $k-r-3/2 \leq i \leq k-1$ and we attain

$$\begin{aligned}\mathcal{Q}_k^{d+1} x^\beta = \mathcal{I}^d_{W_1 \cdots W_d} x^{(\beta_1, ..., \beta_d)} \sum_{i=k-r-1}^{k-1} \Delta_{k-i}^{(d+1)} x^{\beta_{d+1}} &= \mathcal{I}^d_{W_1 \cdots W_d} x^{(\beta_1, ..., \beta_d)} \cdot U_{r+1}^{(d+1)} x^{\beta_{d+1}} \\ &= \mathcal{I}^{d+1}_{W_1 \cdots W_{d+1}} x^\beta\end{aligned}$$

since $2(r+1)-1 = 2r+1 \geq \beta_{d+1}$. This concludes the proof. □

We conclude this section with two examples demonstrating integration of polynomials using the program C.3. We first present a well-behaved example.

Example 2.4. Consider the integral

$$\int_{[0,1]^d} 3^d \prod_{i=1}^{d} x_i^2 \, \mathrm{d}x_d \cdots \mathrm{d}x_1,$$

when $d = 10$.

The integrand is normalized in such a way that the exact result is equal to 1. Clearly the integrand has total degree $k' = 20$. The smallest order to fulfill the criterion $2(k-d)+1 \geq k'$ is $k = 20$, which yields the relative error of $1.68326 \cdot 10^{-7}$ in our numerical experiment.

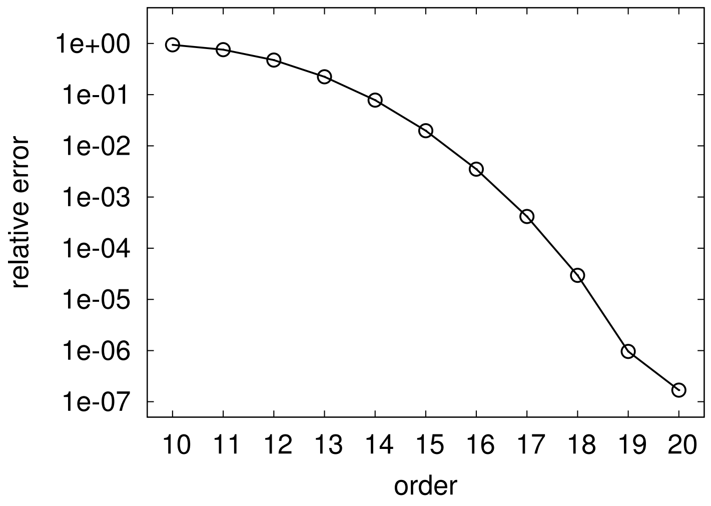


Figure 2.1: The integral of example 2.4 computed using the program C.3.

When the integration region is symmetric with respect to the origin, difficulties begin to arise. The following example should be considered cautionary.

Example 2.5. Consider the integral

$$\int_{\mathbb{R}^d} \frac{2^d}{\pi^{d/2}} \exp\left(-\sum_{i=1}^{d} x_i^2\right) \prod_{i=1}^{d} x_i^2 \, \mathrm{d}x_d \cdots \mathrm{d}x_1,$$

when $d = 10$.

The integrand is normalized such that the result is equal to 1. The total degree of the integrand is 20, discounting the Gaussian weight function. The Smolyak rule of order 20 yields the relative error $6.99441 \cdot 10^{-15}$ but the results look rather peculiar this time around!

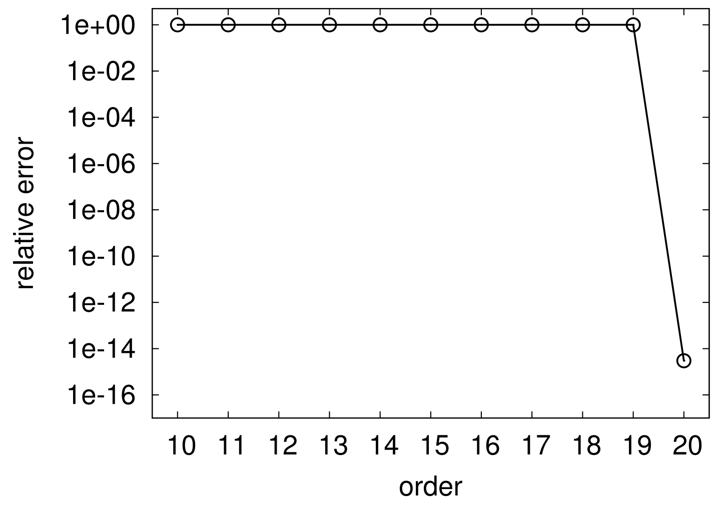


Figure 2.2: The integral of example 2.5 computed using the program C.3.

What happened? The answer has to do with the structure of the Cartesian coordinate system. In the case of symmetric interpolatory quadrature rules, the lower order evaluation points of the Smolyak rule are distributed along the coordinate axes. Due to the product structure of the integrand, the quadrature rule is guaranteed to produce the value zero for any quadrature order $k \leq 2d - 1$. However, proposition 2.3 assures us that the precise nonzero result is attained not later than the order $k = 20$.

The discussion above alludes to the unfortunate side effect of the Smolyak construction: its dependence on the Cartesian coordinate system. This phenomenon has no bearing on the fact that we integrated over the whole space $\mathbb{R}^d$; the same result can be observed by integrating over bounded regions that are symmetric with respect to the origin.

2.2 Fundamental theorem of Smolyak quadrature

The results of the preceding section were delightfully general: we are always assured some notion of convergence when we use the Smolyak rule in conjunction with an interpolatory basis sequence. However, polynomial exactness does not provide us with a tangible convergence rate when we operate outside the class of polynomials.

Before we impose conditions on the basis sequence, we begin this section by introducing a general convergence result due to Wasilkowski and Woźniakowski [28]. As long as the assumptions are valid, we obtain a crude error estimate for the Smolyak rule.

Lemma 2.6. *Suppose that* $||\mathcal{I}^1_{W_j}|| \le A$ *and let* $U^{(j)}_i$ *be univariate quadrature rules such that* $||\mathcal{I}^1_{W_j} - U^{(j)}_i|| \le BC^i$ *for some positive constants* A*,* B *and* C*. Suppose that* $||\Delta^{(j)}_i|| \le DC^i$ *for some* $D > 0$*. Then the following holds:*

$$||\mathcal{I}^d_{W_1\cdots W_d} - \mathcal{Q}^d_k|| \le BC^k \max\left\{\frac{A}{C}, D\right\}^{d-1} \binom{k}{d-1} \quad \textit{for all } k \ge d.$$

Proof. The proof is carried out by induction with respect to d. If $d = 1$, then the claim follows immediately from the univariate case since

$$||\mathcal{I}^1_{W_1} - \mathcal{Q}^1_k|| = \left|\left|\mathcal{I}^1_{W_1} - \sum_{i=1}^{k} \Delta^{(1)}_i\right|\right| = ||\mathcal{I}^1_{W_1} - U^{(1)}_k|| \quad \text{for all } k \ge 1.$$

Suppose that the claim holds for some $d \ge 1$. If $k+1 \ge d+1$, then by utilizing the triangle inequality we attain

$$\begin{aligned}
||\mathcal{I}^{d+1}_{W_1\cdots W_{d+1}} - \mathcal{Q}^{d+1}_{k+1}|| &= ||\mathcal{I}^{d+1}_{W_1\cdots W_{d+1}} - \mathcal{Q}^d_k \otimes \mathcal{I}^1_{W_{d+1}} + \mathcal{Q}^d_k \otimes \mathcal{I}^1_{W_{d+1}} - \mathcal{Q}^{d+1}_{k+1}|| \\
&\le ||(\mathcal{I}^d_{W_1\cdots W_d} - \mathcal{Q}^d_k) \otimes \mathcal{I}^1_{W_{d+1}}|| \\
&\quad + \left|\left| \sum_{\substack{|\alpha|_1 \le k \\ \alpha \in \mathbb{N}^d,\ \alpha \ge \mathbf{1}}} \bigotimes_{i=1}^{d} \Delta^{(i)}_{\alpha_i} \otimes \mathcal{I}^1_{W_{d+1}} - \sum_{\substack{|\alpha|_1 \le k+1 \\ \alpha \in \mathbb{N}^{d+1},\ \alpha \ge \mathbf{1}}} \bigotimes_{i=1}^{d+1} \Delta^{(i)}_{\alpha_i} \right|\right|.
\end{aligned}$$

Using the tensor product theorem and assumption $||\mathcal{I}^1_{W_j}|| \le A$, we can evaluate the former term as follows:

$$||(\mathcal{I}^d_{W_1\cdots W_d} - \mathcal{Q}^d_k) \otimes \mathcal{I}^1_{W_{d+1}}|| \le ||\mathcal{I}^d_{W_1\cdots W_d} - \mathcal{Q}^d_k|| \cdot ||\mathcal{I}^1_{W_{d+1}}|| \le A||\mathcal{I}^d_{W_1\cdots W_d} - \mathcal{Q}^d_k||.$$

Apply dimension recursion to the latter sum in the second term to get

$$\left|\left|\sum_{\substack{|\alpha|_1\le k\\ \alpha\in\mathbb{N}^d,\ \alpha\ge\mathbf{1}}}\bigotimes_{i=1}^{d}\Delta_{\alpha_i}^{(i)}\otimes\mathcal{I}_{W_{d+1}}^1-\sum_{\substack{|\alpha|_1\le k+1\\ \alpha\in\mathbb{N}^{d+1},\ \alpha\ge\mathbf{1}}}\bigotimes_{i=1}^{d+1}\Delta_{\alpha_i}^{(i)}\right|\right|$$
$$\le\left|\left|\sum_{\substack{|\alpha|_1\le k\\ \alpha\in\mathbb{N}^d,\ \alpha\ge\mathbf{1}}}\bigotimes_{i=1}^{d}\Delta_{\alpha_i}^{(i)}\otimes\mathcal{I}_{W_{d+1}}^1-\sum_{\substack{|\alpha|_1\le k\\ \alpha\in\mathbb{N}^d,\ \alpha\ge\mathbf{1}}}\bigotimes_{i=1}^{d}\Delta_{\alpha_i}^{(i)}\otimes U_{k+1-|\alpha|_1}^{(d+1)}\right|\right|.$$

Regrouping the formula above nets us

$$||\mathcal{I}_{W_1\cdots W_{d+1}}^{d+1}-\mathcal{Q}_{k+1}^{d+1}||\le A||\mathcal{I}_{W_1\cdots W_d}^{d}-\mathcal{Q}_k^d||$$
$$+\left|\left|\sum_{\substack{|\alpha|_1\le k\\ \alpha\in\mathbb{N}^d,\ \alpha\ge\mathbf{1}}}\bigotimes_{i=1}^{d}\Delta_{\alpha_i}^{(i)}\otimes(\mathcal{I}_{W_{d+1}}^1-U_{k+1-|\alpha|_1}^{(d+1)})\right|\right|.$$

The tensor product theorem implies

$$||\mathcal{I}_{W_1\cdots W_{d+1}}^{d+1}-\mathcal{Q}_{k+1}^{d+1}||\le A||\mathcal{I}_{W_1\cdots W_d}^{d}-\mathcal{Q}_k^d||$$
$$+\sum_{\substack{|\alpha|_1\le k\\ \alpha\in\mathbb{N}^d,\ \alpha\ge\mathbf{1}}}\prod_{i=1}^{d}||\Delta_{\alpha_i}^{(i)}||\cdot||\mathcal{I}_{W_{d+1}}^1-U_{k+1-|\alpha|_1}^{(d+1)}||.$$

Utilizing the rest of the assumptions yields

$$||\mathcal{I}_{W_1\cdots W_{d+1}}^{d+1}-\mathcal{Q}_{k+1}^{d+1}||\le ABC^k\max\left\{\frac{A}{C},D\right\}^{d-1}\binom{k}{d-1}$$
$$+\sum_{\substack{|\alpha|_1\le k\\ \alpha\in\mathbb{N}^d,\ \alpha\ge\mathbf{1}}}D^dC^{|\alpha|_1}BC^{k+1-|\alpha|_1}$$
$$=ABC^k\max\left\{\frac{A}{C},D\right\}^{d-1}\binom{k}{d-1}$$
$$+BC^{k+1}D^d\sum_{i=d}^{k}\sum_{\substack{|\alpha|_1=i\\ \alpha\in\mathbb{N}^d,\ \alpha\ge\mathbf{1}}}1.$$

The latter sum is equal to the cardinality of the set $\{\alpha \in \mathbb{N}^d;\ |\alpha|_1 = i \text{ and } \alpha \geq \mathbf{1}\}$, which is equal to $\binom{i-1}{d-1}$ by theorem B.3ii). Moreover, property (B.2) implies that

$$\sum_{i=d}^{k} \binom{i-1}{d-1} = \binom{k}{d}.$$

Hence

$$||\mathcal{I}^{d+1}_{W_1 \cdots W_{d+1}} - \mathcal{Q}^{d+1}_{k+1}|| \leq ABC^k \max\left\{\frac{A}{C}, D\right\}^{d-1} \binom{k}{d-1} + BC^{k+1}D^d \binom{k}{d}.$$

If $A/C \geq D$, then by Pascal's identity:

$$\begin{aligned} ||\mathcal{I}^{d+1}_{W_1 \cdots W_{d+1}} - \mathcal{Q}^{d+1}_{k+1}|| &\leq A^d BC^{k-d+1} \binom{k}{d-1} + BA^d C^{k+1-d} \binom{k}{d} \\ &= A^d BC^{k-d+1} \left[\binom{k}{d-1} + \binom{k}{d}\right] \\ &= A^d BC^{k-d+1} \binom{k+1}{d} \\ &= BC^{k+1} \frac{A^d}{C^d} \binom{k+1}{d}. \end{aligned}$$

If $A/C \leq D$ holds instead, we have similarly

$$\begin{aligned} ||\mathcal{I}^{d+1}_{W_1 \cdots W_{d+1}} - \mathcal{Q}^{d+1}_{k+1}|| &\leq ABC^k D^{d-1} \binom{k}{d-1} + BC^{k+1}D^d \binom{k}{d} \\ &\leq BC^{k+1}D^d \binom{k}{d-1} + BC^{k+1}D^d \binom{k}{d} \\ &= BC^{k+1}D^d \binom{k+1}{d}. \end{aligned}$$

This proves the claim. □

Call back to mind the Landau notation. Let f and g be functions $\mathbb{N} \to \mathbb{R}$. We write $f(n) = \mathcal{O}(g(n))$, if there exist constants $C > 0$ and $M \in \mathbb{N}$ such that $|f(n)| \leq C|g(n)|$ for all $n \geq M$. Observe that the following characterization is valid:

$$f(n) = \mathcal{O}(g(n)) \quad \text{if and only if} \quad \limsup_{n\to\infty} \left|\frac{f(n)}{g(n)}\right| < \infty.$$

Example 2.7. Fix $d \in \mathbb{N} \setminus \{0\}$. If we let $k \geq d$, then

$$\binom{k-1}{d-1} = \frac{(k-1) \cdot \ldots \cdot (k-(d-1))}{(d-1) \cdot \ldots \cdot 1} = \mathcal{O}(k^{d-1}). \tag{2.1}$$

This implies

$$\binom{k}{d-1} = \mathcal{O}((k+1)^{d-1}) = \mathcal{O}(k^{d-1}). \tag{2.2}$$

In the expression above we have used the fact that the leading polynomial term is dominant in asymptotic behaviour:

$$a_n x^n + \ldots + a_1 x + a_0 = \mathcal{O}(x^n), \quad a_0, \ldots, a_n \in \mathbb{R}.$$

Let us go back to the implications of lemma 2.6. At this point a couple of problems occur. To achieve a convergence rate, we need to have a good grasp on the convergence properties of the univariate rules. For example, the convergence rate of the Gauss-Hermite rule differs greatly from the Gaussian rules in a bounded interval [7]. Moreover, it is difficult to provide classical convergence rates for the Gauss-Patterson and Genz-Keister schemes due to their construction.

For the remainder of this section we consider the integral with unitary weight function over the hypercube $[-1,1]^d$

$$\mathcal{I}^d f = \int\limits_{[-1,1]^d} f(x_1, \ldots, x_d) \, \mathrm{d}x_d \cdots \mathrm{d}x_1,$$

where $f \in H^r([-1,1]^d)$ and $d \geq 1$. We choose the univariate rules U_i to be interpolatory quadrature rules with positive weights and n_i evaluation points such that $n_i < n_{i+1}$. This allows us to use the following result from approximation theory to our advantage.

Theorem 2.8. *Let U_i be interpolatory quadrature rules for the positive functional $\mathcal{I}^1$ with n_i evaluation points in $[-1,1]$. If the rules have positive weights and $n_i < n_{i+1}$, then the following holds in the class of functions of regularity r:*

$$\limsup_{i \to \infty} (n_i^r ||\mathcal{I}^1 - U_i||) < \infty.$$

Proof. This result is proven by Brass in [3]. A closely related theorem is proven by Petras in [22]. □

We are now able to verify that the assumptions of lemma 2.6 are fulfilled. The choice of interval implies that $A = 2$. As a corollary to theorem 2.8, it is possible to find a constant $\beta_r = \beta(r)$ corresponding to the function space of regularity r such that

$$||\mathcal{I}^1 - U_i|| \leq \beta_r n_i^{-r} \quad \text{for all } i = 1, 2, 3, ..., \tag{2.3}$$

when U_i have the properties stated above. Selecting quadrature rules with $n_i = 2^{i-1}$ nodes, we can set $B = B(r) = 2^r \beta_r$ and $C = C(r) = 2^{-r}$. Furthermore, we compute

$$||\Delta_i|| \leq ||U_i - \mathcal{I}^1|| + ||\mathcal{I}^1 - U_{i-1}|| \leq B(1 + 2^r) \cdot 2^{-ri} = B(1 + 2^r)C^i.$$

Set $D = D(r) = B(1 + 2^r)$ to fulfill the assumptions of lemma 2.6.

The previous discussion yields an initial estimate for the convergence of the Smolyak quadrature rule.

Corollary 2.9. *Suppose that $f \in H^r([-1, 1]^d)$ and let U_i be interpolatory quadrature rules in $[-1, 1]$ with positive weights and $n_i = 2^{i-1}$ nodes. Then*

$$\left| \int\limits_{[-1,1]^d} f(x_1, ..., x_d)\, \mathrm{d}x_d \cdots \mathrm{d}x_1 - \sum_{\substack{|\alpha|_1 \leq k \\ \alpha \in \mathbb{N}^d}} \bigotimes_{i=1}^{d} \Delta_{\alpha_i} f \right| = \mathcal{O}(2^{-rk} k^{d-1}),$$

where we have suppressed the parameters r and d in the Landau notation.

Proof. We use the fact that the result of lemma 2.6 can be written in the form

$$|\mathcal{I}^d f - \mathcal{Q}_k^d f| \leq BC^k \max\{A/C, D\}^{d-1} \mathcal{O}(k^{d-1})$$

by using (2.2). Plugging in the values of A, B, C and D determined above yields the desired result. □

The corollary states in no uncertain terms that the Smolyak rule is an approximation of a d-dimensional integral provided that $r \geq 1$. It turns out that the convergence rate can be expressed classically as a function of quadrature nodes, making it possible to compare the convergence rate of this quadrature rule to rivalling methods. In this thesis I have dubbed this result the *fundamental theorem of Smolyak quadrature*.

We did most of the work regarding the cost of Smolyak quadrature in proposition 1.10. For the following discussion, denote $N(k, d) = \#\eta(k, d)$. Proposition 1.10 now has a simple consequence.

Corollary 2.10. *Let U_i be univariate quadrature rules with $n_i = 2^{i-1}$ nodes. Then*

$$N(k,d) = \mathcal{O}(2^k k^{d-1}).$$

Proof. We previously ascertained that

$$N(k,d) = \sum_{i=\max\{d,k-d+1\}}^{k} 2^{i-d} \binom{i-1}{d-1}.$$

The mapping $i \mapsto \binom{i-1}{d-1}$ is non-decreasing for each fixed $d \geq 1$. Hence

$$N(k,d) \leq \binom{k-1}{d-1} \sum_{i=\max\{d,k-d+1\}}^{k} 2^{i-d} = 2^{1-d} \binom{k-1}{d-1} (2^k - 2^{\max\{d-1,k-d\}}).$$

Since $\mathcal{O}(2^k - 2^{\max\{d,k-d+1\}}) = \mathcal{O}(2^k)$, we can estimate the asymptotic behaviour of the binomial coefficient by (2.1) to achieve

$$N(k,d) = \mathcal{O}(2^k k^{d-1})$$

proving the claim. □

Let $n_i = 2^{i-1}$ and $k \geq d$ as usual. For sufficiently large k, this implies $2^k \leq N(k,d)$ and hence $k \leq \log_2(N(k,d))$. The logarithm base is arbitrary in terms of asymptotic behaviour, so we opt to suppress it. As a follow-up to corollary 2.10 we achieve

$$2^{-k} = \mathcal{O}\left(\frac{k^{d-1}}{N(k,d)}\right) = \mathcal{O}\left(\frac{\log(N(k,d))^{d-1}}{N(k,d)}\right).$$

It follows that

$$\begin{aligned} ||\mathcal{I}^d - \mathcal{Q}_k^d|| &= \mathcal{O}(2^{-rk} k^{d-1}) = \mathcal{O}\big((2^{-k})^r \log(N(k,d))^{d-1}\big) \\ &= \mathcal{O}\left(\frac{\log(N(k,d))^{r(d-1)}}{N(k,d)^r} \log(N(k,d))^{d-1}\right) \\ &= \mathcal{O}\left(\frac{\log(N(k,d))^{(r+1)(d-1)}}{N(k,d)^r}\right). \end{aligned}$$

This leads to the main result of this section.

Theorem 2.11 (Fundamental theorem of Smolyak quadrature)**.** *Let $n_i = 2^{i-1}$ denote the number of evaluation points of interpolatory quadrature rules U_i with positive weights in $[-1,1]$. If we denote $N(k,d) = \#\eta(k,d)$, then the corresponding Smolyak quadrature rule of degree k has the asymptotic convergence rate of*

$$\left| \int\limits_{[-1,1]^d} f(x_1,...,x_d)\,\mathrm{d}x_d \cdots \mathrm{d}x_1 - \sum_{\substack{|\alpha|_1 \le k \\ \alpha \in \mathbb{N}^d}} \bigotimes_{i=1}^{d} \Delta_{\alpha_i} f \right| = \mathcal{O}\left(\frac{\log(N(k,d))^{(r+1)(d-1)}}{N(k,d)^r} \right)$$

for all $f \in H^r([-1,1]^d)$.

Let N denote the number of quadrature evaluation points, d the dimension and r the regularity. The convergence rate of Smolyak quadrature should be compared with the respective error terms listed in table 2.1. The convergence rates of tensor product quadrature and MC are derived in [7] and the rate of QMC is derived in [18].

tensor product	MC	QMC
$\mathcal{O}(N^{-r/d})$	$\mathcal{O}\left(\frac{1}{\sqrt{N}}\right)$	$\mathcal{O}\left(\frac{\log(N)^d}{N}\right)$

Table 2.1: Convergence rates of various multivariate quadrature rules.

The denominator of the asymptotic convergence rate dominates the logarithmic term in the numerator for sufficiently high values of N. The regularity parameter r provides an additional boost to the Smolyak rule over the alternatives in table 2.1 implying that the Smolyak method is preferable in the integration of smooth functions.

The issue of "sufficiently large N" should not be left unaddressed. Simply put, the threshold is determined both by the integrand's regularity and the dimension of the problem. The dichotomy between these is the following: the dimension has an increasing effect on the order of N and the regularity parameter acts as a counterbalance. To achieve good or even moderate performance in high-dimensional problems, the applicability of the Smolyak rule is effectively restricted to smooth integrands.

In conclusion, the results of this section are validated only for very high values of N – possibly unattainably high by the standards of modern computers – but they do describe the nature of Smolyak's construction insofar as we expect good performance for smooth integrands. This nature is reflected in the polynomial exactness of the Smolyak rule as well.

2.3 Numerical experiments

Genz [9] introduced the following family of functions to test the performance of high-dimensional quadrature rules in the unit hypercube $[0,1]^d$:

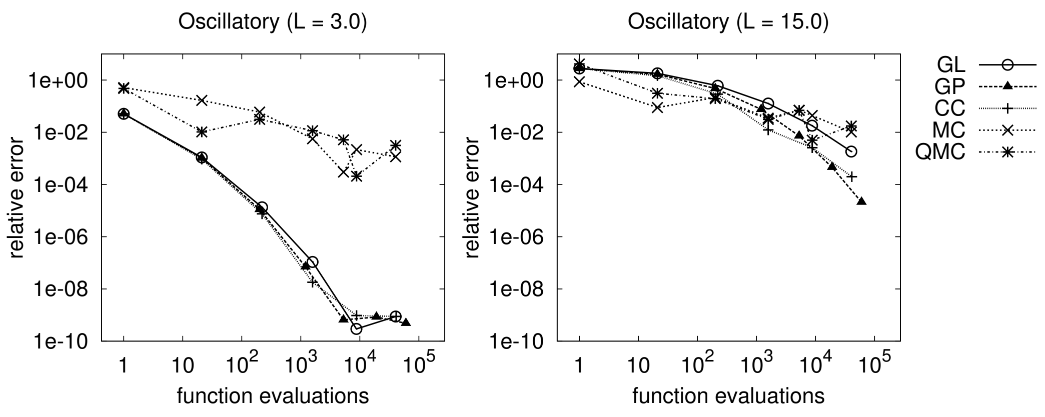
Attribute	Integrand Family
Oscillatory	$f_1(x) = \cos\left(2\pi u_1 + \sum_{i=1}^{d} a_i x_i\right)$
Product Peak	$f_2(x) = \prod_{i=1}^{d} \frac{1}{a_i^{-2} + (x_i - u_i)^2}$
Corner Peak	$f_3(x) = \left(1 + \sum_{i=1}^{d} a_i x_i\right)^{-(d+1)}$
Gaussian	$f_4(x) = \exp\left(-\sum_{i=1}^{d} a_i^2 (x_i - u_i)^2\right)$
Continuous	$f_5(x) = \exp\left(-\sum_{i=1}^{d} a_i \lvert x_i - u_i \rvert\right)$
Discontinuous	$f_6(x) = \begin{cases} 0, & \text{if } x_1 > u_1 \text{ or } x_2 > u_2; \\ \exp\left(\sum_{i=1}^{d} a_i x_i\right) & \text{otherwise.} \end{cases}$

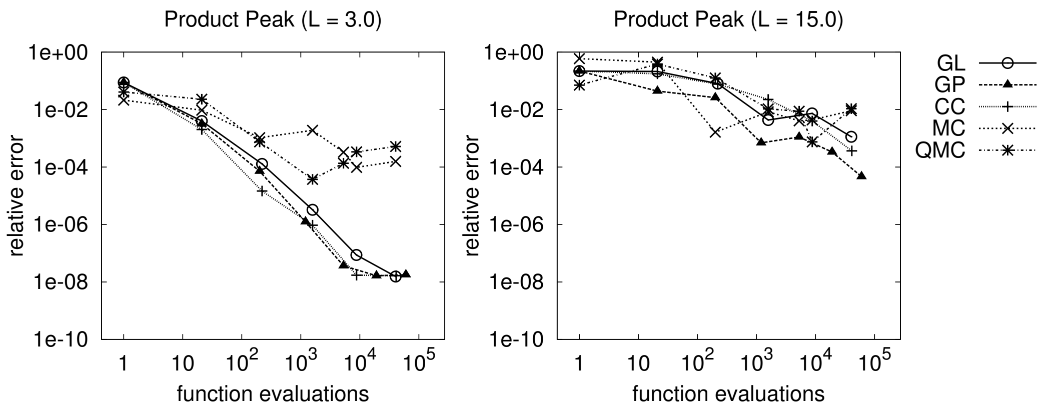
The sequences $(a_i)_{i=1}^d$ and $(u_i)_{i=1}^d$ should be uniformly distributed pseudo-random numbers in $[0,1]$. The sequence $(a_i)_{i=1}^d$ can be scaled to match any given *difficulty parameter* $L = a_1 + ... + a_d$. The higher the difficulty, the less numerically stable the integrand.

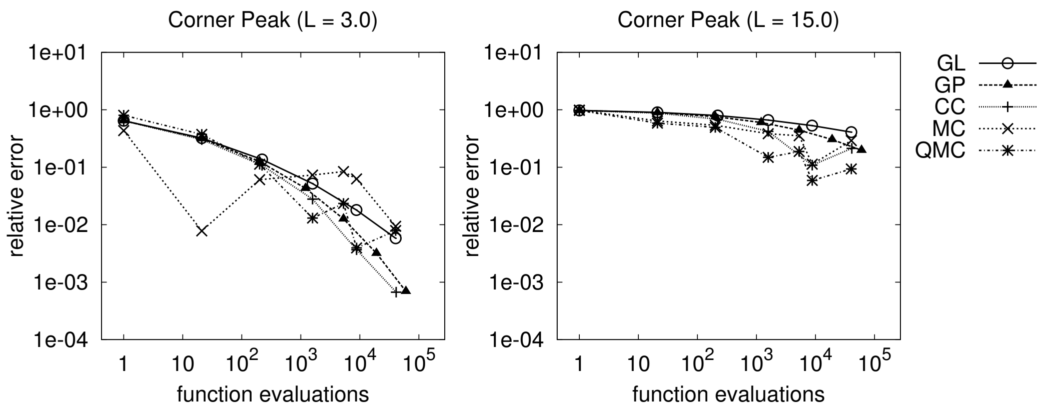
We tested the program C.3 using the Genz functions with $d = 10$ and the difficulty levels $L = 3.0$ and $L = 15.0$. The Smolyak rule was generated using three different basis sequences: slowly increasing Gauss-Legendre, delayed Gauss-Patterson and rapidly increasing Clensaw-Curtis. We compared the performance of the Smolyak rule to the MC and QMC rules, which were computed using the formula

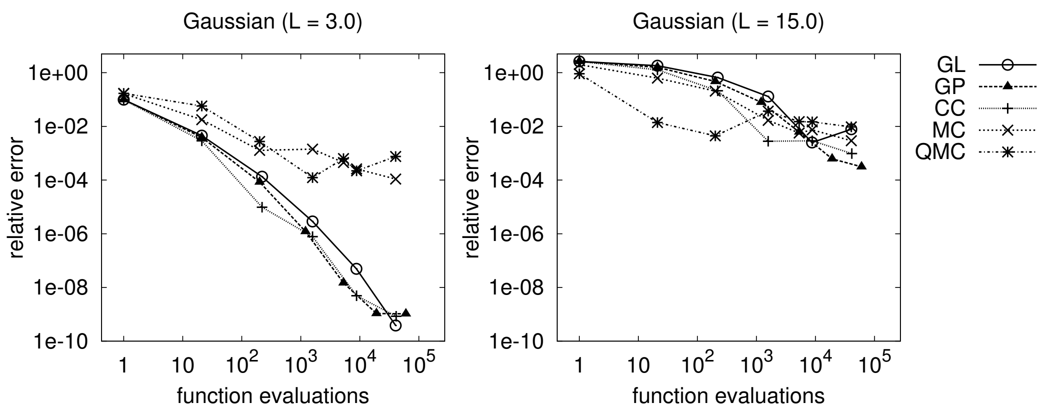
$$\int_\Omega f(x)\,\mathrm{d}x \approx \frac{|\Omega|}{n} \sum_{i=1}^{n} f(x_i),$$

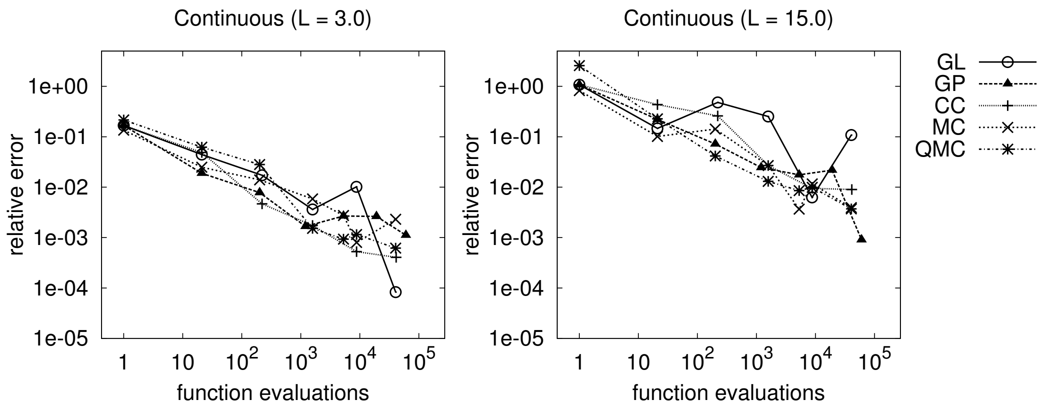
where $\varnothing \neq \Omega \subseteq \mathbb{R}^d$ is a hypercube and $|\cdot|$ denotes the Lebesgue measure of a set. The point set $(x_i)_{i=1}^n$ in $\Omega = [0,1]^d$ was generated using pseudo-random numbers in the case of MC and the Sobol sequence in the case of QMC. The reference values were computed using Wolfram Mathematica 7.

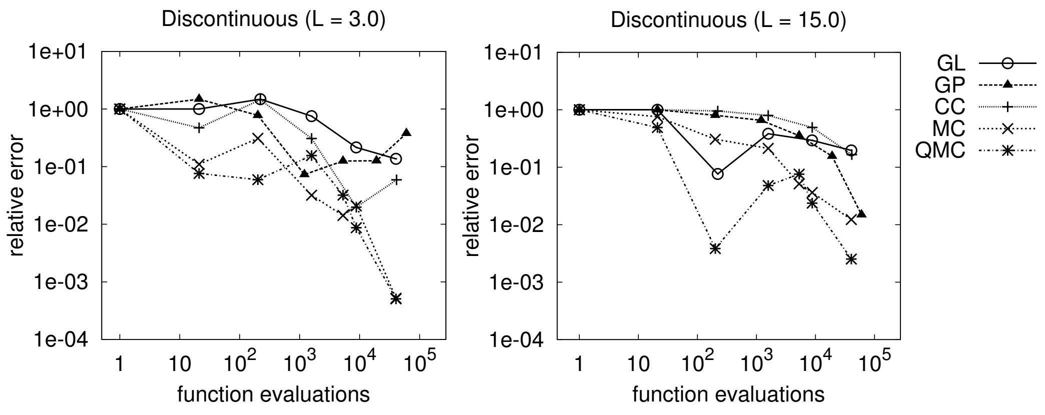

Oscillatory (L = 3.0)
Oscillatory (L = 15.0)
relative error
function evaluations
GL
GP
CC
MC
QMC


Product Peak (L = 3.0)
Product Peak (L = 15.0)
relative error
function evaluations
GL
GP
CC
MC
QMC


Corner Peak (L = 3.0)
Corner Peak (L = 15.0)
relative error
function evaluations
GL
GP
CC
MC
QMC


Gaussian (L = 3.0)
Gaussian (L = 15.0)
relative error
1e+00
1e-02
1e-04
1e-06
1e-08
1e-10
function evaluations
1
10
10^2
10^3
10^4
10^5
GL
GP
CC
MC
QMC


Continuous (L = 3.0)
Continuous (L = 15.0)
relative error
1e+00
1e-01
1e-02
1e-03
1e-04
1e-05
function evaluations
1
10
10^2
10^3
10^4
10^5
GL
GP
CC
MC
QMC


Discontinuous (L = 3.0)
Discontinuous (L = 15.0)
relative error
1e+01
1e+00
1e-01
1e-02
1e-03
1e-04
function evaluations
1
10
10^2
10^3
10^4
10^5
GL
GP
CC
MC
QMC

In the case of integrands with low difficulty, the Smolyak algorithm outperformed MC and QMC in all but the discontinuous case. In the evaluation of the oscillatory, product peak and Gaussian integrands the routine actually reached the accuracy goal of the Mathematica software under a "mere" 10^5 function evaluations. These results reflect the theoretical foundation explored in the first part of this chapter as the user would expect high accuracy from smooth integrands.

Once the difficulty of the Genz functions is increased, the convergence rate of the Smolyak algorithm becomes comparable with the MC and QMC rules. In the case of the differentiable integrands, the Smolyak rule has a tendency to increase in accuracy with each successive order while the evolution of the MC and QMC routines does not have this monotony.

Venturing outside the class of differentiable functions the convergence properties of the Smolyak routine diminish noticeably. None of the Smolyak rules exhibit convergence in any meaningful way in the discontinuous case, and even in the continuous case the Gauss-Legendre rule ceases to converge monotonously.

Example 2.12. Consider the integral

$$\int_{\mathbb{R}^d} e^{-||x||^2} \sin(||x||^2)\,\mathrm{d}x,$$

when $d = 5$.

In this case, the exact value of the integral can be computed as a reference by using formula 3.923.1 in the table of Gradshteyn and Ryzhik [15]

$$\int_{\mathbb{R}} e^{-x^2} \sin(x^2 + r)\,\mathrm{d}x = \frac{\sqrt{\pi}}{\sqrt[4]{2}} \sin\left(\frac{\pi}{8} + r\right) \quad \text{for all } r \in \mathbb{R}.$$

Iterating the formula above d times reveals that

$$\int_{\mathbb{R}^d} e^{-||x||^2} \sin(||x||^2)\,\mathrm{d}x = \frac{\pi^{d/2}}{2^{d/4}} \sin\left(\frac{d\pi}{8}\right) \quad \text{for all } d \geq 1.$$

Using the slowly increasing Gauss-Hermite and delayed Genz-Keister rules, the contribution of the exponential term can be eliminated. We can compare the results attained using program C.3 to the exact value. The results suggest that the delayed Genz-Keister rule is preferable to the slowly increasing Gaussian basis sequence.

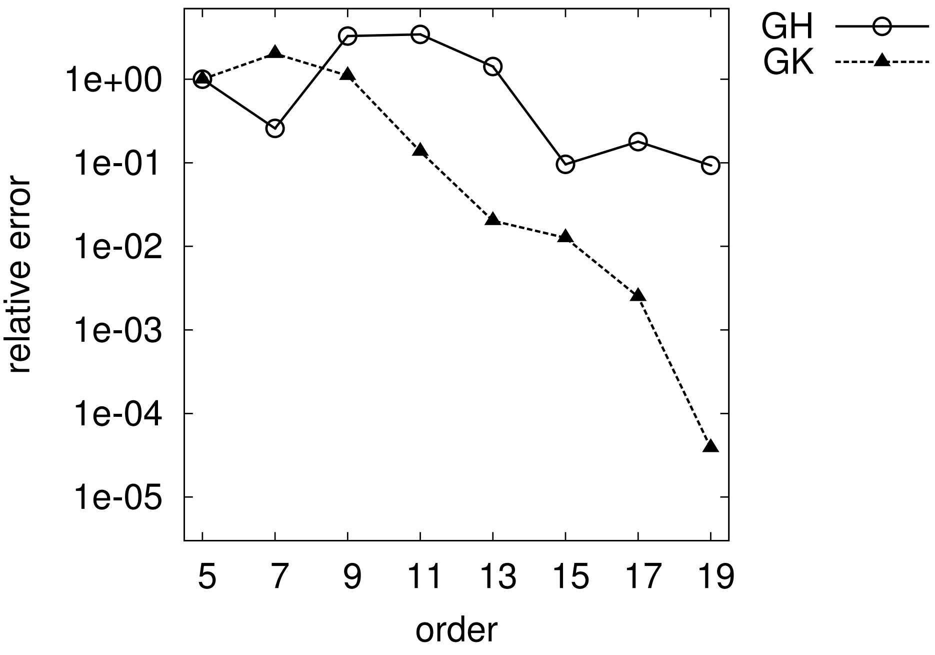


Figure 2.3: The integral of example 2.12 computed using the program C.3.

The difference between the results obtained by these two methods can be explained by the fact that univariate rules with higher polynomial exactness are passed to the Genz-Keister rule at a lower order than the respective Gauss-Hermite rule. While this could be seen as problematic in terms of the number of function evaluations, it should be noted that the 19th order delayed Genz-Keister rule has 98,523 unique nodes compared to the 1,868,878 unique nodes of the 19th order Gauss-Hermite rule. The high exactness and low number of unique evaluation points make delayed basis sequences very attractive in high-dimensional applications.

Chapter 3

Generalized sparse grids

In this section we touch upon the concept of generalized sparse grid quadrature. The subject matter is considered qualitatively only and in the ensuing discussion we suppress the spatial direction of the univariate rules by writing $U_i = U_i^{(j)}$ for univariate rules and $\Delta_i = \Delta_i^{(j)}$ for the respective difference operators.

Definition 3.1. Given an index set $\mathscr{I} \subseteq \mathbb{N}^d$, the *generalized sparse grid quadrature rule* is the operator

$$\mathcal{Q}^d_{\mathscr{I}} = \sum_{\alpha \in \mathscr{I}} \bigotimes_{i=1}^{d} \Delta_{\alpha_i}. \tag{3.1}$$

Remark 5. By choosing $\mathscr{I} = \mathscr{I}(k,d) = \{\alpha \in \mathbb{N}^d;\ |\alpha|_1 \leq k\}$ in the formula (3.1) we attain the Smolyak rule of order k. Selecting $\mathscr{I} = \mathscr{I}(k,d) = \{\alpha \in \mathbb{N}^d;\ |\alpha|_\infty \leq k\}$ nets us the corresponding tensor product quadrature formula. These can therefore be regarded as special cases of the generalized sparse grid method. Program C.6 is a MATLAB implementation of the rule (3.1), where the index set is supplied by the user. The tensor product of difference operators is computed using proposition 1.7.

We shall present two strategies to compute generalized sparse grid quadrature index sets: the method of weighted index sets and the dimension-adaptive construction by Gerstner [13]. The methods are integrand-specific insofar as they use spatial properties of the integrand to determine which indices can be suppressed based on their contribution to the sum total.

3.1 Weighted index set

Consider the ordinary Smolyak quadrature rule of dimension d and order k:

$$\mathcal{Q}_k^d = \sum_{\substack{|\alpha|_1 \le k \\ \alpha \in \mathbb{N}^d}} \bigotimes_{i=1}^{d} \Delta_{\alpha_i}.$$

We may have a priori knowledge that certain multi-indices have less contribution to the sum total than others. For instance, such a scenario may present itself in the case of polynomials. If we code this information to the *weight vector* $\gamma = (\gamma_1, ..., \gamma_d) \in \mathbb{R}_+^d$ in such a way that the so-called *Hadamard product* $\alpha \circ \gamma = (\alpha_1\gamma_1, ..., \alpha_d\gamma_d)$ is in $\mathbb{N}^d$ for all $\alpha \in \mathbb{N}^d$, $|\alpha|_1 \le k$, then we can replace the quadrature rule by the formula

$$\widetilde{\mathcal{Q}}_k^d = \sum_{\substack{\beta = \alpha\circ\gamma \\ |\alpha|_1 \le k,\ \alpha \in \mathbb{N}^d}} \bigotimes_{i=1}^{d} \Delta_{\beta_i}.$$

We illustrate this strategy with an example.

Example 3.2. Consider the function $f(x,y,z) = g(x,z)$ in $[-1,1]^3$. Determine $\mathcal{Q}_5^3 f$ using univariate rules U_1, U_2 and U_3 in $[-1,1]$.

The difference operator Δ_i measures the change in the quadrature formulae U_i and U_{i-1} for $i > 1$. It is apparent from the structure of the integrand that there is no contribution to the difference operator along the y-axis. In the expression for the 5th order Smolyak rule, we can safely remove the marked terms without a decrease in quadrature accuracy for the specific integrand:

$$\begin{aligned}
\mathcal{Q}_5^3 f &= \Delta_1 \otimes \Delta_1 \otimes \Delta_1 f + \Delta_2 \otimes \Delta_1 \otimes \Delta_1 f + \cancel{\Delta_1 \otimes \Delta_2 \otimes \Delta_1 f} \\
&\quad + \Delta_1 \otimes \Delta_1 \otimes \Delta_2 f + \cancel{\Delta_2 \otimes \Delta_2 \otimes \Delta_1 f} + \Delta_2 \otimes \Delta_1 \otimes \Delta_2 f \\
&\quad + \cancel{\Delta_1 \otimes \Delta_2 \otimes \Delta_2 f} + \Delta_3 \otimes \Delta_1 \otimes \Delta_1 f + \cancel{\Delta_1 \otimes \Delta_3 \otimes \Delta_1 f} \\
&\quad + \Delta_1 \otimes \Delta_1 \otimes \Delta_3 f \\
&= U_1 \otimes U_1 \otimes U_1 f + U_2 \otimes U_1 \otimes U_1 f - U_1 \otimes U_1 \otimes U_1 f \\
&\quad + U_1 \otimes U_1 \otimes U_2 f - U_1 \otimes U_1 \otimes U_1 f + U_2 \otimes U_1 \otimes U_2 f \\
&\quad - U_1 \otimes U_1 \otimes U_2 f - U_2 \otimes U_1 \otimes U_1 f + U_1 \otimes U_1 \otimes U_1 f \\
&\quad + U_3 \otimes U_1 \otimes U_1 f - U_2 \otimes U_1 \otimes U_1 f + U_1 \otimes U_1 \otimes U_3 f - U_1 \otimes U_1 \otimes U_2 f \\
&= -U_2 \otimes U_1 \otimes U_1 f - U_1 \otimes U_1 \otimes U_2 f + U_2 \otimes U_1 \otimes U_2 f \\
&\quad + U_3 \otimes U_1 \otimes U_1 f + U_1 \otimes U_1 \otimes U_3 f.
\end{aligned}$$

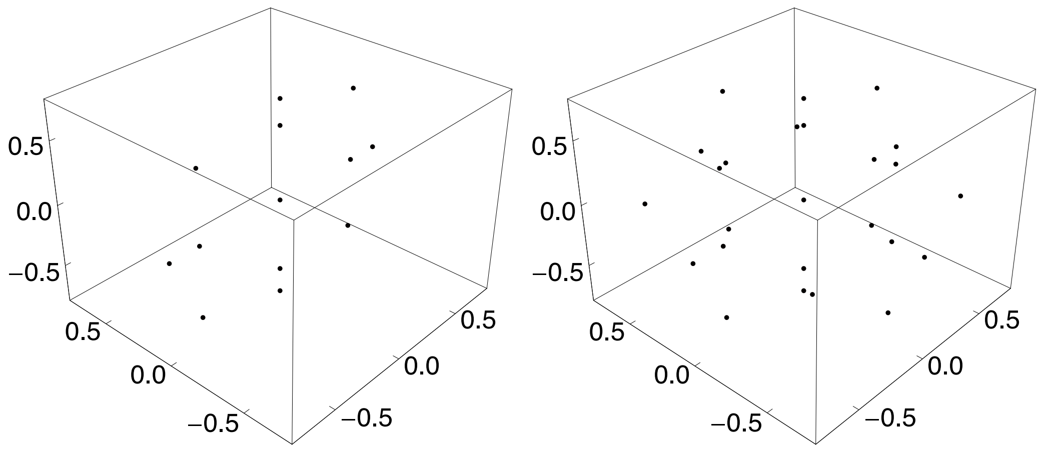


Figure 3.1: The weighted and unweighted grids of example 3.2 using a slowly increasing sequence of Gauss-Legendre rules with 1, 2 and 3 evaluation points.

We can apply the inference in the previous example to e.g. any spatially oriented integrand, such as a polynomial. In practice, this method is much too reliant on foreknowledge of the integrand to be absolutely reliable.

3.2 Dimension-adaptive quadrature

In the case of the Smolyak or tensor product quadrature rule, the index set $\mathscr{I} = \mathscr{I}(k,d)$ is predetermined by the dimension d and order k. The *dimension-adaptive* construction of the index set takes into account the spatial structure of the integrand and adds increments to the index set along the spatial direction that contributes *most* to its respective difference operator. This is the diametric opposite of the method presented in example 3.2, where we removed the directions with the *least* contribution to the sum. As an added benefit, it is possible to terminate the algorithm at any point and still attain an estimate for the integral.

Let f be the integrand and $\hat{\mathbf{e}}_i = (0,...,1,...,0) \in \mathbb{R}^d$ the standard basis vector which is zero with the exception of its i^{th} component. The idea is to start with an *active index set* $\mathscr{A} = \{\mathbf{1}\}$ and an *old index set* $\mathscr{B} = \{\beta \in \mathbb{N}^d;\ \beta_i = 0 \text{ for some } i = 1,...,d\}$. At each successive iteration, we take the element $\alpha \in \mathscr{A}$ with the highest *local error indicator*

$$g_\alpha = |\Delta_{\alpha_1} \otimes \cdots \otimes \Delta_{\alpha_d} f|$$

and update the index sets by setting

$$\mathscr{A} = \mathscr{A} \setminus \{\alpha\} \quad \text{and} \quad \mathscr{B} = \mathscr{B} \cup \{\alpha\}.$$

New indices are added to $\mathscr{A}$ if they fulfill the following admissibility criterion: if for some $i = 1, ..., d$ the multi-index $\beta = \alpha + \hat{\mathbf{e}}_i$ satisfies $\beta - \hat{\mathbf{e}}_j \in \mathscr{B}$ for all $j = 1, ..., d$, then we add it to the active index set $\mathscr{A} = \mathscr{A} \cup \{\beta\}$ and update the integrand value and error estimate accordingly. The integral can be estimated by computing the sum (3.1) over the union of $\mathscr{A}$ and $\mathscr{B}$

$$\mathcal{Q}^d_{\mathscr{A}\cup\mathscr{B}} = \sum_{\alpha\in\mathscr{A}\cup\mathscr{B}} \bigotimes_{i=1}^{d} \Delta_{\alpha_i}.$$

Algorithm 3.3 (Dimension-adaptive quadrature).

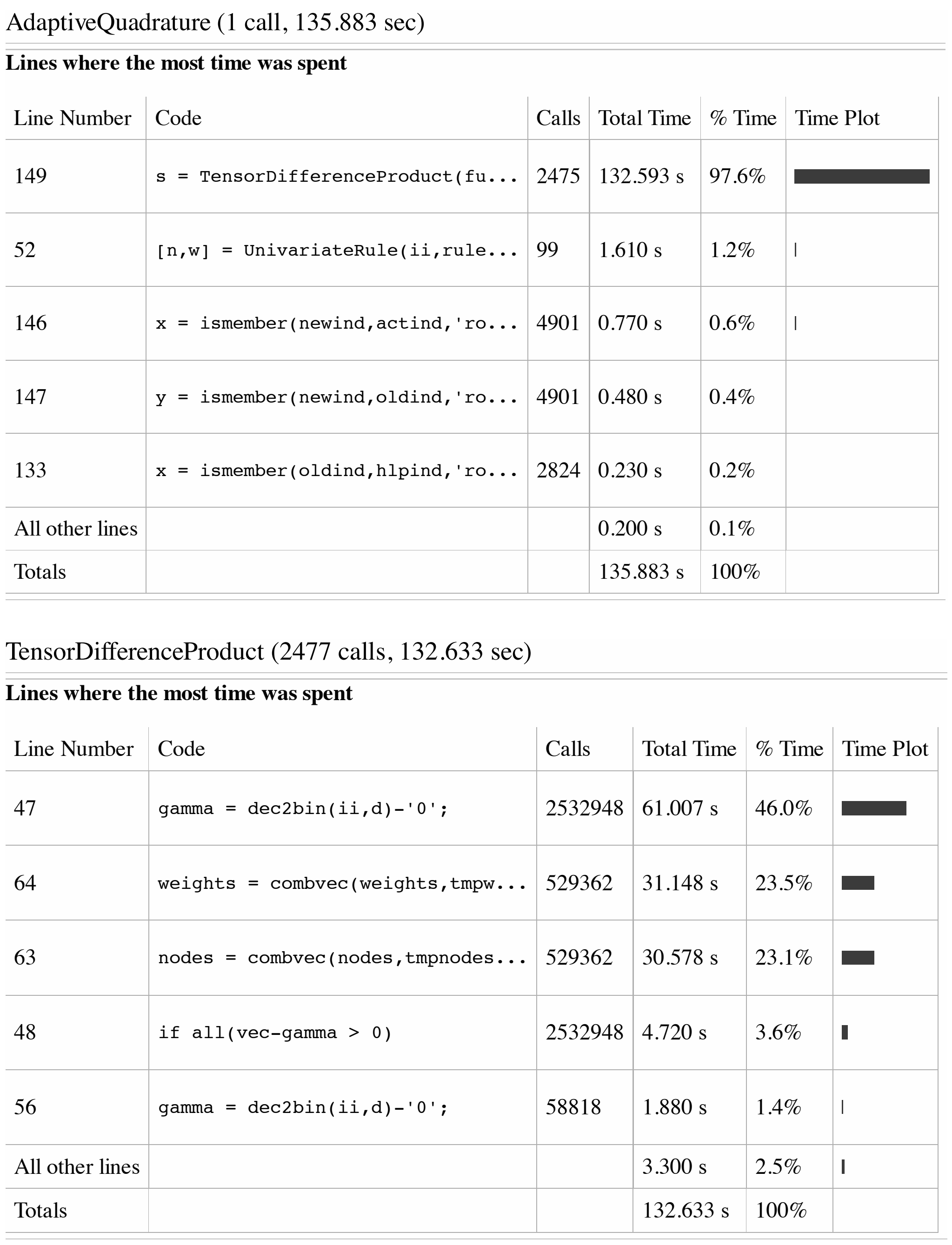
Compute the generalized sparse grid index set and integral estimate of the function f using dimension-adaptive quadrature.

Input: function f, dimension d, tolerance TOL
Output: multi-index set $\mathscr{A} \cup \mathscr{B}$, integral estimate res

```
$\alpha = \mathbf{1}$;
$\mathscr{A} = \{\alpha\}$;
$\mathscr{B} = \{\beta \in \mathbb{N}^d;\ \beta_i = 0$ for some $i = 1, ..., d\}$;
$res = \Delta_{\alpha_1} \otimes \cdots \otimes \Delta_{\alpha_d} f$;
$\eta = g_\alpha$;
while $\eta > TOL$ do
    select $\alpha$ from $\mathscr{A}$ with the largest local error indicator $g_\alpha$;
    remove $\alpha$ from $\mathscr{A}$;
    add $\alpha$ to $\mathscr{B}$;
    $\eta = \eta - g_\alpha$;
    for $i = 1, ..., d$ do
        $\beta = \alpha + \hat{\mathbf{e}}_i$;
        if $\beta - \hat{\mathbf{e}}_j \in \mathscr{B}$ for all $j = 1, ..., d$ then
            add $\beta$ to $\mathscr{A}$;
            $s = \Delta_{\beta_1} \otimes \cdots \otimes \Delta_{\beta_d} f$;
            $res = res + s$;
            $\eta = \eta + g_\beta$;
        end
    end
end
return $res$, $\mathscr{A} \cup \mathscr{B}$.
```

Program C.4 is the MATLAB implementation of this algorithm.

A test run to integrate the function $f(x_1, ..., x_{10}) = 3^{10}x_1^2 \cdots x_{10}^2$ over the unit hypercube $[0, 1]^{10}$ with tolerance 1 % produced the following profile report.



AdaptiveQuadrature (1 call, 135.883 sec)

Lines where the most time was spent

Line Number	Code	Calls	Total Time	% Time	Time Plot
149	`s = TensorDifferenceProduct(fu...`	2475	132.593 s	97.6%	▬▬▬▬▬
52	`[n,w] = UnivariateRule(ii,rule...`	99	1.610 s	1.2%	|
146	`x = ismember(newind,actind,'ro...`	4901	0.770 s	0.6%	|
147	`y = ismember(newind,oldind,'ro...`	4901	0.480 s	0.4%	
133	`x = ismember(oldind,hlpind,'ro...`	2824	0.230 s	0.2%	
All other lines			0.200 s	0.1%	
Totals			135.883 s	100%	

TensorDifferenceProduct (2477 calls, 132.633 sec)

Lines where the most time was spent

Line Number	Code	Calls	Total Time	% Time	Time Plot
47	`gamma = dec2bin(ii,d)-'0';`	2532948	61.007 s	46.0%	▬▬▬
64	`weights = combvec(weights,tmpw...`	529362	31.148 s	23.5%	▬
63	`nodes = combvec(nodes,tmpnodes...`	529362	30.578 s	23.1%	▬
48	`if all(vec-gamma > 0)`	2532948	4.720 s	3.6%	▮
56	`gamma = dec2bin(ii,d)-'0';`	58818	1.880 s	1.4%	|
All other lines			3.300 s	2.5%	|
Totals			132.633 s	100%	

Figure 3.2: The profile report produced by a test run of the program C.4.

The run time is distributed evenly between the `combvec` routine and the generator of binary vectors. The complexity of these is interchangeable: should we directly use the formula (3.1), the amount of calls to the binary vector routine would be transitioned over to the `combvec` routine.

In problematic cases it may be advantageous to attempt brute forcing a solution by setting the tolerance to zero and placing a maximum iteration count. Using a different basis sequence may also speed up the convergence of this routine.

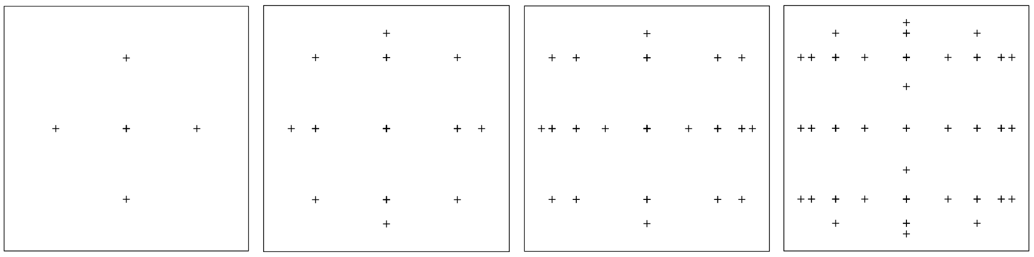


Figure 3.3: The evolution of the dimension-adaptive grid for $f(x, y) = e^{x+y}$ using the slowly increasing Gauss-Legendre basis sequence in $[-1, 1] \times [-1, 1]$.

We end this section with a numerical example conducted using the program C.4.

Example 3.4. Consider the integral

$$\int\limits_{[-1,1]^d} \frac{1}{2^d \sinh^d(1)} \exp\left(\sum_{i=1}^{d} x_i\right) \mathrm{d}x_d \cdots \mathrm{d}x_1,$$

where $d = 10$.

The integral has been normalized such that the result is equal to 1. The development of the estimate is documented in the figure on the next page. After 500 iterations, the relative error is $3.38788 \cdot 10^{-3}$.

3.3 Concluding remarks

The Smolyak quadrature rule provides a competitive alternative to the market of multidimensional quadrature rules saturated by MC and QMC routines. The tests of section 2.3 showed that the Smolyak rule outperforms its competitors in the domain of smooth integrands. This observation is validated by the theory explored in chapter 2.

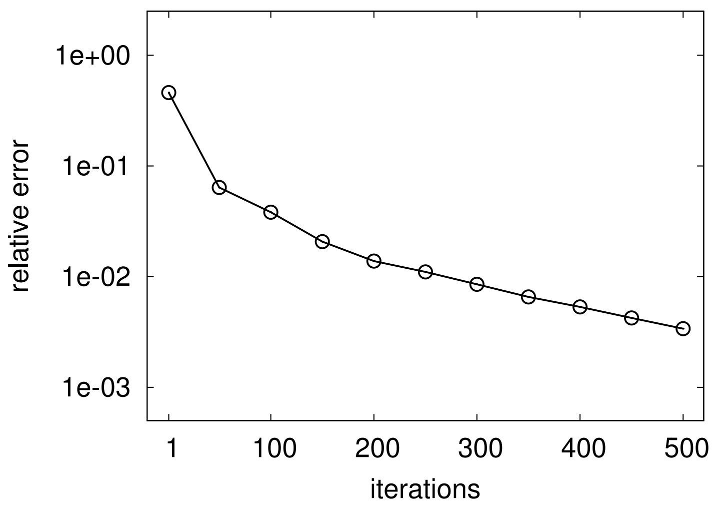


Figure 3.4: The integral of example 3.4 computed using the program C.4.

There are benefits to the use of Smolyak quadrature. The construction of the Smolyak rule provides it with a strong theoretical foundation. As a deterministic method it does not call costly random functions and its rate of convergence overcomes both MC and QMC in problems with sufficiently high smoothness. The Smolyak quadrature rule also retains the polynomial exactness of the univariate quadrature rules, a property for which neither MC nor QMC accounts.

The Smolyak rule provides an interesting utility in the evaluation of Gaussian integrals

$$\int\limits_{\mathbb{R}^d} e^{-||x||^2} f(x)\,\mathrm{d}x$$

without the need to apply a smoothness reducing transformation to the integral. Gaussian integrals arise in a variety of applications in physics, stochastics, and finance. The dimensionality of these problems may easily be in the range of hundreds of variables and as such their computation with any reasonable degree of accuracy practically requires the use of low-discrepancy methods. The dimension-adaptive construction may provide additional computational relief if the problem is spatially oriented.

Appendix A

On univariate quadrature rules

In this section we list some recommendations for the use of univariate quadrature rules. The rules we discuss have the form

$$Uf = \sum_{i=1}^{n} w_i f(x_i) \tag{A.1}$$

for a sequence of positive weights $(w_i)_{i=1}^n$ and a sequence of nodes $(x_i)_{i=1}^n$ in some interval $\varnothing \neq I \subseteq \mathbb{R}$. We call the rule (A.1) *interpolatory* if the following holds:

$$Up = \int_I W(x)p(x)\,\mathrm{d}x \quad \text{for all polynomials } p \text{ whose degree is at least } n-1.$$

We assume that the weight function W is nonnegative and the mapping $x \mapsto W(x)x^k$ is integrable in I for all $k \in \mathbb{N}$.

If $x_1 < ... < x_n$ is a sequence of points in I, then we can always find an interpolatory formula (A.1) by solving the weights $(w_i)_{i=1}^n$ from the linear system

$$\begin{cases} w_1 + ... + w_n = \displaystyle\int_I W(x)\,\mathrm{d}x; \\ w_1x_1 + ... + w_nx_n = \displaystyle\int_I W(x)x\,\mathrm{d}x; \\ \vdots \\ w_1x_1^{n-1} + ... + w_nx_n^{n-1} = \displaystyle\int_I W(x)x^{n-1}\,\mathrm{d}x. \end{cases} \tag{A.2}$$

The determinant of the system (A.2) is known as the *Vandermonde determinant*, which is nonzero whenever the nodes are distinct [30].

We will discuss the construction of selected quadrature rules in the interval $[-1, 1]$.

Clenshaw-Curtis quadrature. The Clenshaw-Curtis rule is a de facto nested quadrature rule. It is included in this thesis on account of the praise it received by Novak and Ritter in [19]. The cardinality of the sequence is

$$n_1 = 1 \quad \text{and} \quad n_i = 2^{i-1} + 1 \quad \text{for } i > 1.$$

In the first iteration we choose the node $x_1^{(1)} = 0$ and the weight $w_1^{(1)} = 2$. The nodes of the i^{th} iteration can be computed using the formula

$$x_j^{(i)} = -\cos\left(\pi \cdot \frac{j-1}{n_i - 1}\right) \quad \text{for } j = 1, ..., n_i$$

and their respective weights are

$$\begin{aligned} w_1^{(i)} &= w_{n_i}^{(i)} = \frac{1}{n_i(n_i-2)}; \\ w_j^{(i)} &= w_{n_i+1-j}^{(i)} = \frac{2}{n_i - 1}\left(1 - \frac{\cos(\pi(j-1))}{n_i(n_i-2)}\right. \\ &\qquad\qquad \left. - 2\sum_{k=1}^{(n_i-3)/2} \frac{1}{4k^2-1}\cos\left(2\pi k \cdot \frac{j-1}{n_i-1}\right)\right) \end{aligned}$$

for $j = 2, ..., n_i - 1$.

Gaussian quadrature. Gaussian rules are characterized by the fact that they have the maximal degree of exactness $2n-1$ attainable with n evaluation points. Their construction is a delicate matter and will be covered in an extremely abridged manner.

If the weight function W fulfills the requirements listed above, then it is possible to define the inner product

$$\langle p, q \rangle_W = \int_I W(x)p(x)q(x)\,\mathrm{d}x, \quad p \text{ and } q \text{ polynomials},$$

and construct a family $(P_i)_{i=0}^{\infty}$ of *orthogonal polynomials* by applying Gram-Schmidt orthogonalization to the initiator polynomial $P_0(x) = 1$ in the geometry induced by $\langle \cdot, \cdot \rangle_W$. Orthogonal polynomials have various interesting properties which include:

O1) The sequence $(P_i)_{i=0}^{k}$ spans the space of polynomials whose degree is at most k. Especially every polynomial can be expressed as a linear combination of a given sequence of orthogonal polynomials.

O2) Each P_k has k distinct real roots in I.

It can be shown [4] that the orthogonal polynomials are characterized by a three-term recurrence formula

$$\begin{aligned} P_0(x) &= 1; \\ P_1(x) &= (x - a_1)P_0(x); \\ P_{k+1}(x) &= (x - a_{k+1})P_k(x) - b_{k+1}P_{k-1}(x), \quad k \geq 1, \end{aligned}$$

where the coefficients are

$$a_{k+1} = \frac{\langle xP_k, P_k\rangle_W}{\langle P_k, P_k\rangle_W} \quad \text{and} \quad b_{k+1} = \frac{\langle P_k, P_k\rangle_W}{\langle P_{k-1}, P_{k-1}\rangle_W}.$$

In the thesis we use two Gaussian rules: the Gauss-Legendre rule in $[-1, 1]$ with the weight function $W \equiv 1$ and the Gauss-Hermite rule in $\mathbb{R}$ with the weight function $W(x) = e^{-x^2}$. The orthogonal polynomials that are related to these rules are called *Legendre polynomials* defined by

$$a_k = 0 \quad \text{and} \quad b_{k+1} = \frac{1}{4 - k^{-2}} \quad \text{for } k = 1, 2, 3, ...$$

and *Hermite polynomials* defined by

$$a_k = 0 \quad \text{and} \quad b_{k+1} = \frac{k}{2} \quad \text{for } k = 1, 2, 3, ...$$

Writing the three-term recurrence as a telescoping linear system and performing an appropriately chosen similarity transformation, Golub and Welsch [14] found that the nodes $x_j^{(n)}$ of the n^{th} order Gaussian quadrature are precisely the eigenvalues of the symmetric tridiagonal matrix

$$A_n = \begin{pmatrix} a_1 & \sqrt{b_2} & 0 & \cdots & 0 \\ \sqrt{b_2} & a_2 & \sqrt{b_3} & \cdots & 0 \\ 0 & \sqrt{b_3} & a_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_n \end{pmatrix}.$$

Let $v_j = (v_{1,j}, ..., v_{n,j}) \in \mathbb{R}^n$ be the eigenvector $A_n x_j^{(n)} = x_j^{(n)} v_j$ such that $||v_j|| = 1$. Then the weights of the Gaussian quadrature are $w_j^{(n)} = v_{1,j}^2 \langle 1, 1\rangle_W$.

Gaussian rules are notoriously nonnested, meaning that the nodes of the previous iteration cannot be reiterated in subsequent computations. There lies interest in modifying Gaussian rules to be nested even at the penalty of decreased accuracy.

Gaussian rules with n evaluation points can be extended by addition of p nodes. Let p_{n+2p-1} be an arbitrary polynomial whose degree is at most $n+2p-1$. It can be written in the form

$$p_{n+2p-1}(x) = q_{n+p-1}(x) + P_{n+p}(x) \sum_{k=0}^{p-1} c_k x^k,$$

where q_{n+p-1} is a polynomial with degree at most $n+p-1$ and P_{n+p} is the $(n+p)^{\text{th}}$ order orthogonal polynomial corresponding to the interval and weight function. The extended quadrature rule is interpolatory by construction so it integrates q_{n+p-1} exactly. Moreover, it holds that $\langle P_{n+p}, x^k \rangle = 0$ for $k = 0, ..., p-1$. Hence the extended quadrature rule has the degree of exactness $n+2p-1$.

Gauss-Patterson quadrature is the extension of Gauss-Legendre rules in $[-1, 1]$ discovered by Patterson in [21]. The nested counterpart of the Gauss-Hermite rule is known as *Genz-Keister quadrature*, the derivation of which is described in [10]. Tables of the Gauss-Patterson and Genz-Keister quadrature rules can be found in e.g. the QUADPACK library at http://people.sc.fsu.edu/~jburkardt/f_src/quadpack/quadpack.html.

Appendix B

Combinatorial results

In this section we go through some of the combinatorial results used in the course of this thesis. Especially of interest are the results concerning cardinalities of multi-index sets.

Consider an arbitrary set A and call back to mind the definition of the power set: $\mathcal{P}(A) = \{B;\ B \subseteq A\}$. We can use this to establish the standard definition of the binomial coefficient.

Definition B.1. The *binomial coefficient* is defined by setting

$$\binom{d}{k} = \#\{A \in \mathcal{P}(\{0, 1, ..., , d-1\});\ \#A = k\}$$

for integers $0 \leq k \leq d$ and 0 otherwise. Especially $\binom{0}{0} = 1$.

Recall that the binomial coefficient $\binom{d}{k}$ denotes the number of ways of picking k unordered outcomes out of d possibilities when $0 \leq k \leq d$. This allows us to discover a curious recursive property. Denote $\mathcal{P}(d, k) = \{A \in \mathcal{P}(\{0, ..., d-1\});\ \#A = k\}$. The following can be shown:

i) there exists a bijective mapping $\phi \colon \mathcal{P}(d+1, k+1) \to \mathcal{P}(d, k) \cup \mathcal{P}(d, k+1)$;

ii) $\mathcal{P}(d, k) \cap \mathcal{P}(d, k+1) = \varnothing$.

Since ϕ is a bijection and its image set can be expressed in terms of two distinct point sets, we have

$$\begin{aligned}\#\mathcal{P}(d+1, k+1) = \#\phi[\mathcal{P}(d+1, k+1)] &= \#\mathcal{P}(d, k) \cup \mathcal{P}(d, k+1) \\ &= \#\mathcal{P}(d, k) + \#\mathcal{P}(d, k+1).\end{aligned}$$

Using the definition of the binomial coefficient yields the following result.

Lemma B.2 (Pascal's identity)**.** *Let $0 \leq k \leq d$. Then the following holds:*

$$\binom{d+1}{k+1} = \binom{d}{k} + \binom{d}{k+1}.$$

Repeated application of the previous lemma reveals a simple recursive method to generate the binomial coefficient for any pair $(d, k) \in \mathbb{N}^2$. Visually it has the representation known as *Pascal's triangle*:

$$\begin{array}{ccccccccc}
 & & & & 1 & & & & \\
 & & & 1 & & 1 & & & \\
 & & 1 & & 2 & & 1 & & \\
 & 1 & & 3 & & 3 & & 1 & \\
1 & & 4 & & 6 & & 4 & & 1 \\
\vdots & & \vdots & & \vdots & & \vdots & & \vdots
\end{array}$$

In other words, the element on the $(d+1)^{\text{th}}$ row and $(k+1)^{\text{th}}$ column of Pascal's triangle is precisely the binomial coefficient of (d, k).

Pascal's identity is a powerful tool when combined with mathematical induction to derive combinatorial identities. The two identities we have listed here can be found in the table of Gradshteyn and Ryzhik [15, equations 0.151.1 and 0.151.4]:

$$\sum_{i=0}^{k} (-1)^i \binom{d}{i} = (-1)^k \binom{d-1}{k} \quad \text{for } d \geq 1 \text{ and } k \geq 0 \tag{B.1}$$

and

$$\sum_{i=0}^{k} \binom{d+i}{d} = \binom{d+k+1}{d+1} \quad \text{for } d \geq 1 \text{ and } k \geq 0. \tag{B.2}$$

Of special interest to us are the cardinalities of multi-index sets, which can be expressed in terms of combinatorial quantities.

Theorem B.3.

i) $\#\{\gamma \in \{0,1\}^d;\ |\gamma|_1 = k\} = \binom{d}{k}$ *for* $0 \leq k \leq d$*;*

ii) $\#\{\alpha \in \mathbb{N}^d;\ |\alpha|_1 = k \text{ and } \alpha \geq \mathbf{1}\} = \binom{k-1}{d-1}$ *for* $0 \leq d \leq k$*.*

Proof. We follow the classical "stars and bars" argument presented by Feller [8]. We represent by stars ($\star$) the amount of *objects* that are enclosed in *bins*, which we separate by *bars* ($|$). For example, we can represent $k = 9$ objects by using nine stars

$$\star\star\star\star\star\star\star\star\star$$

and one possible way to enclose them into $d = 5$ bins is

$$\star\star\,|\,\star\,|\,\star\,|\,\star\star\star\star|\,\star\,.$$

Feller argues that the nature of the objects does not matter, hence the abstraction of combinatorial quantities by arbitrary objects that are placed into bins can be used to determine general combinatorial results.

i) We interpret the vector $\gamma \in \{0,1\}^d$ as a collection of d bins such that 0 denotes an empty bin and 1 a nonempty one. If $|\gamma|_1 = k$, then precisely k bins are nonempty. By the very definition of the binomial coefficient there exist $\binom{d}{k}$ possible configurations of k nonempty bins out of a total of d bins.

ii) The vector elements are positive in this case, which forbids the existence of empty bins. Let us separate the d bins by $d-1$ bars. Multiple bars cannot be placed adjacent to each other. Between k objects there exist now $k-1$ gaps, which we are free to fill with $d-1$ bars. The amount of possible configurations of $d-1$ bars placed into $k-1$ free gaps is $\binom{k-1}{d-1}$. □

By accounting for the possibility of empty loci in the multi-index, we can exploit theorem B.3ii) to $k \to k+d$ objects and arrive at the following corollary.

Corollary B.4.

$$\#\{\alpha \in \mathbb{N}^d;\ |\alpha|_1 = k\} = \binom{d+k-1}{d-1} \quad \textit{for } d \geq 1 \textit{ and } k \geq 0.$$

Appendix C

MATLAB programs

This is a repository of the programs created for this thesis. The programs are © Vesa Kaarnioja (2013) and they can be freely distributed and modified with the author's permission.

Program C.1 (Univariate quadrature rules).

```
function [nodes,weights] = UnivariateRule(n,rule)

% This function loads the univariate nodes and weights from the
% files '/rule/nodes#.dat' and '/rule/weights#.dat', where # =
% n. The available rules are slowly increasing Gauss-Legendre
% ('GL'), delayed Gauss-Patterson ('GP'),  Clenshaw-Curtis
%  ('CC') in [0,1] with 2^(n-1)+1 nodes for n > 1, Gauss-
% Hermite ('GH') and delayed Genz-Keister ('GK'). With the
% exception of GH and GK, the rules are given in [0,1].

nodes = zeros(n,1);
weights = zeros(n,1);
switch rule

    % Gauss-Legendre rule in [0,1]

    case 'GL'
        name1 = sprintf('GL/nodes%d.dat',n);
        name2 = sprintf('GL/weights%d.dat',n);

    % Gauss-Hermite rule
    case 'GH'
        name1 = sprintf('GH/nodes%d.dat',n);
        name2 = sprintf('GH/weights%d.dat',n);

    % Clenshaw-Curtis rule in [0,1] (Novak & Ritter)

    case 'CC'
        name1 = sprintf('CC/nodes%d.dat',n);
        name2 = sprintf('CC/weights%d.dat',n);

    % Delayed Gauss-Patterson rule in [0,1]

    case 'GP'
        name1 = sprintf('GP/nodes%d.dat',n);
        name2 = sprintf('GP/weights%d.dat',n);

    % Delayed Genz-Keister rule

    case 'GK'
        name1 = sprintf('GK/nodes%d.dat',n);
        name2 = sprintf('GK/weights%d.dat',n);
end
nodes = load(name1);
weights = load(name2);
nodes = reshape(nodes,1,[]);
weights = reshape(weights,1,[]);
```

Program C.2 (Generator of multi-indices $\alpha \in \mathbb{N}^d$ such that $\alpha \geq \mathbf{1}$ and $|\alpha|_1 = \ell$.).

```
function [ind,count] = Drop(d,ell)

% This function determines all d-tuples the sum of whose
% elements is equal to k. It is based on algorithm 8.1.1 in
% "Sparse Grid Quadrature Methods for Computational Finance"
% (2007) by Thomas Gerstner.

k = ones(1,d);
khat = (ell-d+1)*k;
ind = zeros(1,d);
count = 0;
```

```
p = 1;
while k(d) <= ell
    k(p) = k(p)+1;
    if k(p) > khat(p)
        if p ~= d
            k(p) = 1;
            p = p+1;
        end
    else
        khat(1:1:p-1) = khat(p)-k(p)+1;
        k(1) = khat(1);
        p = 1;
        count = count+1;
        ind(count,:) = k;
    end
end
```

Program C.3 (Generator of Smolyak quadrature nodes and weights).

```
function SmolyakRule(d,k,rule)

% This function stores the d-dimensional Smolyak quadrature
% rule of order k in the mat-file SmolyakRule_rule_d#_k##.mat,
% where # = dimension and ## = order. The value of k should be
% at least d and the value of d should be at least 2.
%
% USES: Drop.m, UnivariateRule.m,
%       Neural Network Toolbox (optional).

% Force input values to be integers.

d = round(d);
k = round(k);

% Check that the input parameters are sensible.

if d < 2 || k < d
    error('Invalid input! Enter values k >= d and d >= 2.');
end

% Prompt the user in case of GL, GP or CC rules.
```

```
if strcmp(rule,'GL') || strcmp(rule,'GP') || strcmp(rule,'CC')
    lbnd = input('Enter lower univariate integration bound >');
    ubnd = input('Enter upper univariate integration bound >');
elseif ~strcmp(rule,'GH') && ~strcmp(rule,'GK')
    error('Invalid univariate rule!');
end

% Store the univariate rules to memory.

nw = struct('nodes',[],'weights',[]);
for ii = 1:1:k-d+1
    [nodes,weights] = UnivariateRule(ii,rule);
    nw(ii).nodes = nodes;
    nw(ii).weights = weights;
end

% Initialize the respective arrays for Smolyak quadrature
% nodes and weights.

snodes = [];
sweights = [];

% Designate the name of output data file.

filename = sprintf('SmolyakRule_%s_d%d_k%d.mat',rule,d,k);

% The outmost loop is determined by the Smolyak quadrature
% order.

for l = max(d,k-d+1):1:k

    % Handle the trivial case separately.

    if l == d
        snodes = nw(1).nodes;
        sweights = nw(1).weights;
        snodes = repmat(snodes,d,1);
        if strcmp(rule,'GL') || strcmp(rule,'GP') ...
           || strcmp(rule,'CC')
            snodes = (ubnd-lbnd)*snodes+lbnd;
            sweights = (ubnd-lbnd)*sweights;
        end

        % Account for the Smolyak quadrature coefficient.

        sweights = (-1)^(k-l)*nchoosek(d-1,k-l)*sweights^d;
        if l == k
            save(filename,'snodes','sweights');
        end
    else

        % Determine all multi-indices of dimension d, whose
        % 1-norm is equal to l using the drop algorithm.

        [ind,count] = Drop(d,l);
        for ii = 1:1:count

            % Construct all combinations of univariate
            % quadrature nodes and weights.

            alpha = ind(ii,:);
            tmpnodes = nw(alpha(1)).nodes;
            tmpweights = nw(alpha(1)).weights;
            for jj = 2:1:d
                tmp1 = nw(alpha(jj)).nodes;
                tmp2 = nw(alpha(jj)).weights;

                % The Neural Network Toolbox function
                % combvec is used at this point to determine
                % all vector combinations of the univariate
                % quadrature nodes and weights.

                tmpnodes = combvec(tmpnodes,tmp1);
                tmpweights = combvec(tmpweights,tmp2);
            end

            % If the univariate integration interval is
            % specified, apply the appropriate transformations
            % to the nodes and weights.

            if strcmp(rule,'GL') || strcmp(rule,'GP') ...
```

```
               || strcmp(rule,'CC')
                tmpnodes = (ubnd-lbnd)*tmpnodes+lbnd;
                tmpweights = (ubnd-lbnd)*tmpweights;
            end

            % Account for the Smolyak quadrature coefficient
            % and collect the Smolyak quadrature data into the
            % arrays snodes and sweights.

            tmpweights = (-1)^(k-l)*nchoosek(d-1,k-l)*...
                         prod(tmpweights);
            snodes = [snodes,tmpnodes];
            sweights = [sweights,tmpweights];
        end
    end
end
if k ~= d

    % Store the results to a data file.

    save(filename,'snodes','sweights');
end
```

Program C.4 (Dimension-adaptive quadrature).

```
function [res,ind,count] = AdaptiveQuadrature(func,d,rule,...
                           tol,maxiter)

% This function is an implementation of the dimension-adaptive
% quadrature rule. The user supplies the integrand, dimension,
% desired univariate rule, tolerance and maximum iteration
% count. The dimension must not be less than 2. The function
% returns the integral value, corresponding generalized sparse
% grid quadrature index set and final iteration count.
%   If there are issues with convergence, use a different basis
% sequence or attempt to brute force a solution by setting the
% tolerance to zero.
%
% USES: UnivariateRule.m, TensorDifferenceProduct.m,
%       Neural Network Toolbox (optional).
% Initializations.

d = round(d);
maxiter = round(maxiter);
lbnd = 0;
ubnd = 1;
if d < 2
    error('The dimension must not be less than 2!');
end
if nargin < 4
    error('Enter tolerance and maximum iteration count!');
elseif maxiter < 1 || nargin < 5
    error('No maximum iteration count specified!');
end

% Prompt the user in case of GL, GP or CC rules.

if strcmp(rule,'GL') || strcmp(rule,'GP') || strcmp(rule,'CC')
    lbnd = input('Enter lower integration bound > ');
    ubnd = input('Enter upper integration bound > ');
elseif ~strcmp(rule,'GH') && ~strcmp(rule,'GK')
    error('Invalid univariate rule!');
end

% Store univariate rules to memory.

nw = struct('nodes',[],'weights',[]);
maxdepth = 99; % Change the maximum depth if higher resolution
               % is required.
for ii = 1:1:maxdepth
   [n,w] = UnivariateRule(ii,rule);
   nw(ii).nodes = n;
   nw(ii).weights = w;
end

% Format the active index set actind and old index set oldind.

actind = ones(1,d);
oldind = [];

% Compute the initial integral increment and its absolute value
% Delta.

res = TensorDifferenceProduct(func,actind,rule,nw,lbnd,ubnd);
Delta = abs(res);
gerr = Delta;
eta = Delta;
count = 0;
actsize = 1;

% Make sure that the integrand is not constant in the
% neighbourhood of the initial active index set. Otherwise
% shift the active index set.

flag = 1;
for ii = 1:1:d
    hlpind = actind;
    hlpind(ii) = hlpind(ii)+1;
    hlpres = TensorDifferenceProduct(func,hlpind,rule,nw,...
             lbnd,ubnd);
    if abs(hlpres) > max(1e-12,tol)
        flag = 0;
        break;
    end
end
if flag
    actind = [actind;actind+ones(1,d)];
    s = TensorDifferenceProduct(func,actind(2,:),rule,nw,...
        lbnd,ubnd);
    res = res+s;
    Delta = abs(s);
    eta = eta+Delta;
    gerr = [gerr;Delta];
    actsize = actsize+1;
end

% Construct a loop that terminates once the global error
% estimate eta is less than the supplied tolerance or once the
% maximum iteration count is reached.

while eta >= tol && count < maxiter
```

```
% Select the vector maxind from the active index set actind
% that corresponds to the largest local error indicator in
% gerr.

[tmp,gind] = max(gerr);
maxind = actind(gind,:);

% Move the vector maxind to the old index set oldind and
% remove it from the active index set actind.

oldind = [oldind;maxind];
actind = [actind(1:1:gind-1,:);actind(gind+1:1:actsize,:)];
gerr = [gerr(1:1:gind-1,:);gerr(gind+1:1:actsize,:)];
actsize = actsize-1;

% Update the global error estimate.

eta = eta-tmp;

% Search for admissible indices in the neighbourhood of
% maxind.

for ii = 1:1:d
    newind = maxind;
    newind(ii) = newind(ii)+1;
    flag = 1;

    % Apply the admissibility condition to newind.

    for jj = 1:1:d
        hlpind = newind;
        hlpind(jj) = hlpind(jj)-jj;

        % Discard null indices.

        if hlpind(jj) > 0
            x = ismember(oldind,hlpind,'rows');
            if ~any(x)
                flag = 0;
                break;
            end
            end
        end

        % If newind is admissible, append the active index set
        % actind, compute the integral increment s and update
        % the integral value and error estimates.

        if flag
            x = ismember(newind,actind,'rows');
            y = ismember(newind,oldind,'rows');
            if ~any(x) && ~any(y)
                s = TensorDifferenceProduct(func,newind,...
                    rule,nw,lbnd,ubnd);
                actind = [actind;newind];
                actsize = actsize+1;
                res = res+s;
                Delta = abs(s);
                eta = eta+Delta;
                gerr = [gerr;Delta];
            end
        end
    end
    count = count+1;
end

% The sparse grid quadrature index set is the union of the
% active and old index sets.

ind = [actind;oldind];
```

Program C.5 (Tensor product of difference operators).

```
function res = TensorDifferenceProduct(func,vec,rule,nw,...
               lbnd,ubnd)

% Computes the integral increment res of the function func
% using the multi-index  vec. The univariate rule should be
% supplied in the struct nw containing nodes and weights in the
% format specified in the programs SmolyakRule.m,
% AdaptiveQuadrature.m and SparseGridQuadrature.m. If any one
% of the rules GL, GP or CC is used, then the lower and upper
% integration bounds should be input as lbnd and ubnd respec-
% tively.
%
% USES: Neural Network Toolbox (optional)

% Initializations.

res = 0.0;
s = size(vec);
d = s(2);

% If the upper and lower integration bounds are not supplied,
% use the unit interval.

if nargin < 5
    lbnd = 0;
    ubnd = 1;
elseif nargin == 5
    error('Input the upper integration bound!');
end
if norm(vec-ones(1,d),1) < .5

    % Handle the trivial case separately.

    nodes = nw(1).nodes;
    weights = nw(1).weights;
    nodes = repmat(nodes,d,1);
    if strcmp(rule,'GL') || strcmp(rule,'GP') ...
       || strcmp(rule,'CC')
       nodes = (ubnd-lbnd)*nodes+lbnd;
       weights = (ubnd-lbnd)*weights;
    end
    res = weights^d*feval(func,nodes);
else

    % Generate binary multi-indices. We keep the indices that
    % fulfill the criterion vec-gamma >= 1 as specified in
    % proposition 1.7.

    ind = 0;
    for ii = 1:1:2^d-1
```

```
        gamma = dec2bin(ii,d)-'0';
        if all(vec-gamma > 0)
            ind = [ind,ii];
        end
    end

    % Determine quadrature nodes exactly as in SmolyakRule.m.

    for ii = ind
        gamma = dec2bin(ii,d)-'0';
        beta = vec-gamma;
        nodes = nw(beta(1)).nodes;
        weights = nw(beta(1)).weights;
        for jj = 2:1:d
            tmpnodes = nw(beta(jj)).nodes;
            tmpweights = nw(beta(jj)).weights;
            nodes = combvec(nodes,tmpnodes);
            weights = combvec(weights,tmpweights);
        end
        if strcmp(rule,'GL') || strcmp(rule,'GP') || ...
           strcmp(rule,'CC')
            nodes = (ubnd-lbnd)*nodes+lbnd;
            weights = (ubnd-lbnd)*weights;
        end
        res = res+(-1)^sum(gamma)*...
              sum(prod(weights).*feval(func,nodes));
    end
end
```

Program C.6 (Sparse grid quadrature).

```
function res = SparseGridQuadrature(func,ind,rule)

% This function determines the integral of func using the
% generalized sparse grid quadrature rule. The user must supply
% the multi-index set ind and the corresponding univariate rule.
%
% USES: UnivariateRule.m, TensorDifferenceProduct.m,
%       Neural Network Toolbox (optional).

lbnd = 0;
```

```
ubnd = 1;

% Prompt the user in case of GL, GP or CC rules.

if strcmp(rule,'GL') || strcmp(rule,'GP') || strcmp(rule,'CC')
    lbnd = input('Enter lower univariate integration bound >');
    ubnd = input('Enter upper univariate integration bound >');
elseif ~strcmp(rule,'GH') && ~strcmp(rule,'GK')
    error('Invalid univariate rule!');
end

% Initializations.

s = size(ind);
maxiter = s(1);
res = 0.0;

% Store univariate rules to memory.

nw = struct('nodes',[],'weights',[]);
for ii = 1:1:max(max(ind))
    [n,w] = UnivariateRule(ii,rule);
    nw(ii).nodes = n;
    nw(ii).weights = w;
end

% Use the formula for the generalized sparse grid rule to
% compute the integral.

for ii = 1:1:maxiter
    alpha = ind(ii,:);
    res = res+TensorDifferenceProduct(func,alpha,rule,nw,...
          lbnd,ubnd);
end
```

Program C.7 (Alternative combvec routine).

```
function vec = combvec(vec1,vec2)

% An alternative function to use in place of the Neural Network
% Toolbox function of the same name.
```

```
vec = [kron(vec1,ones(1,length(vec2)));
      reshape(vec2'*ones(1,length(unique(vec1,'rows'))),1,[])];
```

Bibliography

[1] Aarts, R. & Weisstein, E. ”Fubini theorem” from *Wolfram MathWorld.* http://mathworld.wolfram.com/FubiniTheorem.html

[2] Björck, Å & Dahlquist, G. *Numerical Methods in Scientific Computing, Volume I.* Society for Industrial and Applied Mathematics, 2008.

[3] Brass, H. ”Error Bounds Based on Approximation Theory” in *NATO ASI Series*, Vol. 357 (1992), pp. 147-163.

[4] Bulirsch, R. & Stoer, J. *Introduction to Numerical Analysis.* Springer, New York, 1980.

[5] Bungartz, H. & Griebel, M. ”Sparse Grids” in *Acta Numerica*, Vol. 13 (2004), pp. 1-123.

[6] Cools, R. ”Constructing cubature formulae: the science behind the art” in *Acta Numerica*, Vol. 6 (1997), pp. 1-54.

[7] Davis, P. & Rabinowitz, P. *Methods of Numerical Integration.* Academic Press, New York, 1975.

[8] Feller, W. *An Introduction to Probability Theory and Its Applications. Volume 1.* Second edition. John Wiley & Sons Inc., New York, 1971.

[9] Genz, A. ”Testing Multidimensional Integration Routines” in *Tools, Methods, and Languages for Scientific and Engineering Computation.* Edited by Ford, B., Rault, J. & Thomasset, F. North-Holland, 1984.

[10] Genz, A. & Keister, B. ”Fully symmetric interpolatory rules for multiple integrals over infinite regions with Gaussian weight” in *Journal of Computational and Applied Mathematics*, Vol. 71, No. 2 (1996), pp. 299-309.

[11] Gerstner, T. *Sparse Grid Quadrature Methods for Computational Finance.* University of Bonn, lecture notes, 2007.

[12] Gerstner, T. & Griebel, M. ”Numerical Integration using Sparse Grids” in *Numerical Algorithms*, Vol. 18, No. 3-4 (1998), pp. 209-232.

[13] Gerstner, T. & Griebel, M. ”Dimension-Adaptive Tensor-Product Quadrature” in *Computing*, Vol. 71, No. 1 (2003), pp. 65-87.

[14] Golub, G. & Welsch, J. ”Calculation of Gauss Quadrature Rules” in *Mathematics of Computation*, Vol. 23, No. 106 (1969), pp. 221-230.

[15] Gradshteyn, I. & Ryzhik, I. *Table of Integrals, Series, and Products.* Fourth edition. Academic Press, New York, 1980.

[16] Heiss, F. & Winschel, V. ”Likelihood approximation by numerical integration on sparse grids” in *Journal of Econometrics*, Vol. 144, No. 1 (2008), pp. 62-80.

[17] Holtz, M. *Sparse Grid Quadrature in High Dimensions with Applications in Finance and Insurance.* Springer, Heidelberg, 2011.

[18] Niederreiter, H. *Random Number Generation and Quasi-Monte Carlo Methods.* Society for Industrial and Applied Mathematics, Philadelphia, 1992.

[19] Novak, E. & Ritter, K. ”High dimensional integration of smooth functions over cubes” in *Numerische Mathematik*, Vol. 75, No. 1 (1996), pp. 79-97.

[20] Novak, E. & Ritter, K. ”Simple Cubature Formulas with High Polynomial Exactness” in *Constructive Approximation*, Vol. 15, No. 4 (1999), pp. 499-522.

[21] Patterson, T. ”The Optimum Addition of Points to Quadrature Formulae” in *Mathematics of Computation*, Vol. 22, No. 104 (1968), pp. 847-856.

[22] Petras, K. ”Asymptotic behaviour of Peano kernels of fixed order” in *Numerical Integration III*, *International Series of Numerical Mathematics 85.* Birkhäuser Verlag, Basel, 1988, pp. 186-198.

[23] Press, W., Teukolsky, S., Vetterling, W. & Flannery, B. *Numerical Recipes: The Art of Scientific Computing.* Third edition. Cambridge University Press, New York, 2007.

[24] Reed, M. & Simon, B. *Methods of Modern Mathematical Physics: Functional Analysis I (Revised and enlarged edition).* Academic Press, San Diego, 1980, pp. 49-54.

[25] Smolyak, S. ”Quadrature and interpolation formulas for tensor products of certain classes of functions” in *Soviet Mathematics*, Vol. 4 (1963), pp. 240-243. Translation of Doklady Akademii Nauk SSSR.

[26] Trefethen, L. ”Is Gauss Quadrature better than Clenshaw-Curtis?” in *SIAM Review*, Vol. 50, No. 1 (2008), pp. 67-87.

[27] Urban, K. *Numerical Finance.* University of Ulm, lecture notes, 2009.

[28] Wasilkowski, G. & Woźniakowski, H. ”Explicit Cost Bounds of Algorithms for Multivariate Tensor Product Problems” in *Journal of Complexity*, Vol. 11 (1995), pp. 1-56.

[29] Weisstein, E. ”Binomial Coefficient” from *Wolfram MathWorld.* http://mathworld.wolfram.com/BinomialCoefficient.html

[30] Weisstein, E. ”Vandermonde Determinant” from *Wolfram MathWorld.* http://mathworld.wolfram.com/VandermondeDeterminant.html