mathematics

MDPI

Article

Challenging the Curse of Dimensionality in Multidimensional Numerical Integration by Using a Low-Rank Tensor-Train Format

Boian Alexandrov [1], Gianmarco Manzini [1,*], Erik W. Skau [1], Phan Minh Duc Truong [1] and Radoslav G. Vuchov [2]

[1] Theoretical Division, Los Alamos National Laboratory, Los Alamos, NM 87545, USA
[2] Computational Mathematics Department, Sandia National Laboratories, Albuquerque, NM 87185, USA
* Correspondence: gmanzini@lanl.gov

Abstract: Numerical integration is a basic step in the implementation of more complex numerical algorithms suitable, for example, to solve ordinary and partial differential equations. The straightforward extension of a one-dimensional integration rule to a multidimensional grid by the tensor product of the spatial directions is deemed to be practically infeasible beyond a relatively small number of dimensions, e.g., three or four. In fact, the computational burden in terms of storage and floating point operations scales exponentially with the number of dimensions. This phenomenon is known as the curse of dimensionality and motivated the development of alternative methods such as the Monte Carlo method. The tensor product approach can be very effective for high-dimensional numerical integration if we can resort to an accurate low-rank tensor-train representation of the integrand function. In this work, we discuss this approach and present numerical evidence showing that it is very competitive with the Monte Carlo method in terms of accuracy and computational costs up to several hundredths of dimensions if the integrand function is regular enough and a sufficiently accurate low-rank approximation is available.

Keywords: multidimensional numerical integration; highly-dimensional cubatures; Gauss–Legendre integration rule; Clenshaw–Curtis integration rule

MSC: 26B15; 26B20

check for updates

Citation: Alexandrov, B.; Manzini, G.; Skau, E.W.; Truong, P.M.D.; Vuchov, R.G. Challenging the Curse of Dimensionality in Multidimensional Numerical Integration by Using a Low-Rank Tensor-Train Format. *Mathematics* **2023**, *11*, 534. https://doi.org/10.3390/math11030534

Academic Editors: Vladislav Kovalnogov and Nadezhda Yarushkina

Received: 25 October 2022
Revised: 13 January 2023
Accepted: 16 January 2023
Published: 19 January 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The ability to calculate very high-dimensional integrals is crucial, for example in quantum mechanics, statistical physics, inverse problems, uncertainty quantifications, probability theory, and many other fields in science and engineering (see [1–3]).

To calculate such integrals accurately, we need to overcome the *curse of dimensionality*, which is bad scaling of the computational complexity in terms of storage and floating point operations with the number of dimensions. In the case of numerical integration, we need one function evaluation at each integration node and one floating point operation to accumulate the result, i.e., a multiplication by the quadrature weight and an addition. If a one-dimensional quadrature formula is composed of m nodes and weights, the corresponding tensor product in d dimensions is composed by m^d nodes and weights. This exponential scaling in the number of dimensions makes this straightforward approach practically impossible except for very small values of d.

The situation is even worse if we try to decompose the integration domain into subdomains (cells) to improve the resolution or adapt the cubature rule when the integrand function shows a low regularity. Partitioning each direction into n sub-intervals leads to a tensor grid with n^d cells and, eventually, to $n^d m^d$ function evaluations and floating point operations. Even if we take only one integration node per cell by setting $m = 1$ as in the midpoint rule, such approach to numerical integration still remains impractical. A

"pedantic" example illustrates why. Setting $d = n = 100$ leads to 10^{200} function evaluations and as many floating point operations. Assuming, very optimistically, that a function evaluation costs only one floating point operation, this approach requires about 2×10^{188} s on a teraflop machine, which virtually performs 10^{12} floating point operations per second. This time is about 4.58×10^{170} times the age of the Universe, which has been recently estimated as 4.17×10^{17} s (see *Planck Collaboration* (2020) [4]).

A non-exhaustive list of methods for multidimensional numerical integration includes the Monte Carlo method and its variants such as the quasi Monte Carlo method [5,6], and the multi-level Monte Carlo (MLMC) method [7]; the sparse grid method [8,9] and the combination technique based on sparse grids [10]; grid-based approaches such as the lattice method [11,12]. It is also worth mentioning that several software libraries are available in the public domain to address this issue, such as *The Sparse Grids Matlab Kit* [13], a collection of Matlab functions for high-dimensional quadrature and interpolation, based on the combination technique version of sparse grids, and *PyApprox* [14], which implements methods such as low-rank tensor-decomposition, Gaussian processes, and polynomial chaos expansions. A detailed presentation of these methods or a systematic review of the huge literature that is available on numerical integration are beyond the scope of this paper and we refer the interested reader to a few of the many textbooks and review papers dealing with these topics, see, e.g., [15–18] and the references therein.

A breakthrough for integrating a multivariate function on a multidimensional grid is offered by the compressive nature of low-rank tensor formats, see [19,20] for a general introduction to tensor representations and a review. The most commonly used tensor formats include the tensor train (TT) and tensor-train cross approximation [21,22], the quantized tensor train (QTT) [23], the canonical polyadic decomposition (CPD) [24], Tucker decomposition [25], and hierarchical Tucker decomposition [26]. TT-based integration has been applied in many fields, such as, just to mention a few, quantum physics and chemistry [27,28], signal processing [29], stochastics, and uncertainty quantification [30]. It is worth mentioning that recent applications of low-rank tensor formats can also be found for solving high-dimensional parabolic PDEs [31] and large ODEs with conservation laws [32]. Using such formats makes it possible to handle efficiently the function evaluation at the integration nodes and the *saxpy* floating point operations that accumulate the numerical value of the integral (the acronym "saxpy" stands for "scalar alpha x plus y", i.e., multiplication by a scalar number and addition). Numerical integration of high-dimensional singular function using the tensor-train format on parallel machines is also investigated in references [33,34].

In this work, we mainly focus on the TT format, which we discuss in the next sections, and its application to composite numerical integration. Our major goal is to show that if it is possible to use an accurate low-rank tensor-train format to represent the integrand as a grid function, we can design very accurate and very efficient multivariate cubature formulas from univariate quadrature formulas in an almost straightforward way. To this end, we focus on cubatures built from the Newton–Cotes (trapezoidal and Simpson), Clenshaw–Curtis, and Gauss–Legendre formulas. However, the strategy is very general and can be adopted for all univariate integration formulas. For the sake of comparison, we consider the Monte Carlo method and its publicly available implementation provided by the library GAIL [35].

The outline of the paper is as follows. In Section 2, we discuss the problem of multidimensional numerical integration and the composite approach, where the computational domain is split into a grid of regularly equispaced cells and a univariate quadrature rule is applied inside each cell and in any direction. In Section 3, we review some basic concepts about the tensor-train format. In Section 4, we investigate the performance of our method for the integration of a set of representative functions and assess its performance. In Section 5, we offer our final remarks and discuss possible future work.

2. Multidimensional Numerical Integration

In this section, we introduce the notation of the paper and briefly review the construction of composite cubature formulas through the tensor product of univariate high-order accurate quadrature rules. First, we split the multidimensional domain into the union of hypercells, which are the tensor product of univariate partitions along the space directions. Then, inside each cell, we build the multidimensional cubature as the tensor product of univariate quadrature rules, and we characterize the accuracy of such a numerical approximation by considering the contribution of each direction to the total integration error.

2.1. Tensor Product Construction of Composite Cubature Formulas

Let $f : \Omega \to \mathbb{R}$ be a real-valued function defined on the multidimensional hypercube $\Omega = \bigtimes_{\ell=1}^{d} \Omega^{(\ell)} = \big(0,1\big)^d$. We want to design a *composite* integration algorithm that numerically approximates the multidimensional integral

$$I(f) = \int_{\Omega} f(x_1, \dots x_d) dx_1 \dots dx_d, \tag{1}$$

where x_ℓ is the position variable along the ℓ-th direction, and $\ell = 1, \dots, d$ is the integer index running over all the space directions. To this end, we introduce a family of multidimensional grids $\{\Omega_{\underline{n}}\}$. We uniquely identify each member $\Omega_{\underline{n}}$ of such a family by the array of strictly positive integers $\underline{n} = (n_1, \dots, n_d)$. Each grid $\Omega_{\underline{n}}$ is the tensor product of d independent partitionings $\Omega_{n_\ell}^{(\ell)}$ for $\ell = 1, \dots, d$, into $n_\ell - 1$ equally spaced subintervals of $[0,1]$, so that $\Omega_{\underline{n}} = \bigtimes_{\ell=1}^{d} \Omega_{n_\ell}^{(\ell)}$. We let $h_\ell = 1/(n_\ell - 1)$ to be the space size in the ℓ-th direction, i.e., the distance between two consecutive nodes of $\Omega_{n_\ell}^{(\ell)}$. We assume that h_ℓ tends to zero uniformly in ℓ, which implies that n_ℓ tends to ∞ uniformly in ℓ. We label the grid nodes of $\Omega_{\underline{n}}$ by using the multi-index $\underline{i} = (i_1, \dots, i_d)$, where $i_\ell = 1, \dots, n_\ell$. We let $\mathcal{I}_{\underline{n}}$ denote the set of all admissible multi-indices in accordance with the given $\underline{n}$, i.e., all possible d-uple combinations of positive integers i_ℓ between 1 and n_ℓ for the multidimensional mesh $\Omega_{\underline{n}}$. The multidimensional grid nodes are given by

$$\mathbf{x}_{\underline{i}} = \mathbf{x}_{(i_1,\dots,i_d)} = \Big(x_{i_1}^{(1)}, \dots, x_{i_d}^{(d)}\Big),$$

where $x_{i_\ell}^{(\ell)} = h_\ell(i_\ell - 1)$ is the position of the i_ℓ-th node along the ℓ-th direction, so that, for all ℓ, it holds that $x_1^\ell = 0$ when $i_\ell = 1$ and $x_{n_\ell}^\ell = 1$ when $i_\ell = n_\ell$. In view of this definition, every grid $\Omega_{\underline{n}}$ decomposes Ω into a set of $N_{\underline{n}} = (n_1 - 1) \dots (n_d - 1)$ smaller hypercubes, the *cells*, that are isomorphic to $\big[0, h_1\big] \times \dots \times \big[0, h_d\big]$. We denote the cell corresponding to the multi-index $\underline{i}$ by $C_{\underline{i}}$, so that the "$\underline{i}$-th" cell is formally given by

$$C_{\underline{i}} = \big[x_{i_1}^{(1)}, x_{i_1+1}^{(1)}\big] \times \dots \times \big[x_{i_d}^{(d)}, x_{i_d+1}^{(d)}\big].$$

Every mesh $\Omega_{\underline{n}} = \{C_{\underline{i}}\}_{\underline{i} \in \mathcal{I}_{\underline{n}}}$ forms a finite cover of Ω, i.e., $\overline{\Omega} = \cup_{\underline{i} \in \mathcal{I}_{\underline{n}}} C_{\underline{i}}$, and the mesh cells are *nonoverlapping* in the sense that the d-dimensional Lebesgue measure (hypervolume) of the intersection of any possible pair of distinct cells is zero, i.e., $|C_{\underline{i}} \cap C_{\underline{j}}| = 0$ for all $\underline{i}, \underline{j} \in \mathcal{I}_{\underline{n}}$, $\underline{i} \neq \underline{j}$. Therefore, we can reformulate $I(f)$ in (1) as the sum of cell subintegrals,

$$I(f) = \sum_{\underline{i} \in \mathcal{I}_{\underline{n}}} I_{C_{\underline{i}}}(f) \quad \text{where} \quad I_{C_{\underline{i}}}(f) = \int_{C_{\underline{i}}} f(x_1, \dots x_d) dx_1 \dots dx_d,$$

and devise a computer algorithm for the *composite numerical integration* over $\Omega_{\underline{n}}$ by assembling local multidimensional cubature rules that approximate all integrals $I_{C_{\underline{i}}}(f)$. In the literature, the word *composite* (or, sometimes, *compound*) normally refers to the combination of one-dimensional Newton–Cotes formulas to subintegrals obtained after splitting a univariate integration domain into a finite number of nonoverlapping subdomains. Hereafter,

we shall use this word to refer to a generic integration algorithm on a multidimensional tensor grid. We will consider composite formulas obtained from the Newton–Cotes as well as Gauss–Legendre and Clenshaw–Curtis quadratures. For the sake of exposition, we assume that the quadrature rule is the same in all univariate subintervals $\left[x_{i_\ell}^{(\ell)}, x_{i_\ell+1}^{(\ell)}\right]$. Different integration formulas can be considered in any subinterval at the price of a more complex description of the algorithm.

Let $Q^m = \left\{\left(\xi_q, \eta_q\right)\right\}_{q=1}^m$ be an m-point quadrature rule with nodes ξ_q and weights η_q for the approximation of a line integral over the reference segment $[-1,1]$. We do not require (nor exclude) that the extremal points ± 1 are part of the integration nodes. We map the reference domain $[-1,1]$ onto all the d one-dimensional intervals $\left[x_{i_\ell}^{(\ell)}, x_{i_\ell+1}^{(\ell)}\right]$ forming a given cell $C_{\underline{i}}$, identified by the multi-index $\underline{i}$. Accordingly, the q_ℓ-th reference node with coordinates $\xi_{q_\ell} \in [-1,1]$ is mapped onto the node with the ℓ-th coordinate equal to $x_{i_\ell,q_\ell}^{(\ell)} = x_{i_\ell}^{(\ell)} + \left(x_{i_\ell+1}^{(\ell)} - x_{i_\ell}^{(\ell)}\right)(\xi_{q_\ell}+1)/2$, and the corresponding weight becomes $w_{q_\ell} = \eta_{q_\ell}\left(x_{i_\ell+1}^{(\ell)} - x_{i_\ell}^{(\ell)}\right)/2$. The tensor product of the d directions provides a local subgrid of quadrature nodes inside $C_{\underline{i}}$. We denote these subgrid nodes by using the multi-index $\underline{q} = (q_1, \ldots q_d)$, so that

$$\mathbf{x}_{\underline{i},\underline{q}} = \left(x_{i_1,q_1}^{(1)}, \ldots, x_{i_d,q_d}^{(d)}\right). \tag{2}$$

The multi-index $\underline{i}$ identifies the $\underline{i}$-th cell, i.e., $C_{\underline{i}}$, and the multi-index $\underline{q} = (q_1, \ldots, q_d)$ identifies the quadrature node inside $C_{\underline{i}}$. In fact, the index $q_\ell = 1, \ldots, m$ labels the q_ℓ-th node of the m-node univariate quadrature rule along the ℓ-th space direction, denoted by the superscript (ℓ) in $x_{i_\ell,q_\ell}^{(\ell)}$. We denote the set of all possible values of $\underline{q}$ by $\mathcal{Q}_m$. The nodes $\mathbf{x}_{\underline{i},\underline{q}}$ are all located in the interior of $C_{\underline{i}}$ if the corresponding reference nodes are inside $(-1,1)$ as in the case of the Gauss–Legendre integration rules. Figure 1 illustrates the notation for the case with $d = 2$.

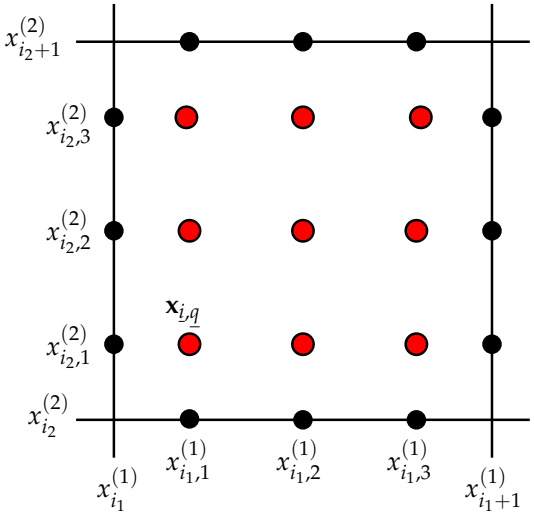


Figure 1. Notation. For $d = 2$, we show the generic cell $C_{\underline{i}} = \left[x_{i_1}^{(1)}, x_{i_1+1}^{(1)}\right] \times \left[x_{i_2}^{(2)}, x_{i_2+1}^{(2)}\right]$, with $\underline{i} = (i_1, i_2)$; the univariate quadrature nodes (black dots) with coordinate $x_{i_\ell,q_\ell}^{(\ell)}$, with $\ell = 1,2$, $q_\ell = 1,2,3$, which corresponds to the 3-point Gauss–Legendre integration rule; the cubature nodes inside the cell (red dots) are provided by the tensor product of the one-dimensional quadrature nodes along the directions $\ell = 1,2$. To make the figure easily readable, we show all the quadrature and cubature nodes relative to cell $C_{\underline{i}}$, but we report only the coordinates of the quadrature nodes at the bottom and left sides, and the coordinates of the internal bottom-left node, i.e., $\mathbf{x}_{\underline{i},\underline{q}}$, with $\underline{i} = (i_1, i_2)$, and $\underline{q} = (1,1)$.

The integration weight associated with the $\underline{q}$-th node of the $\underline{i}$-th cell is the product of the one-dimensional weights identified inside $C_{\underline{i}}$ by $\underline{q} = (q_1, \ldots, q_d)$, where $q_\ell = 1, \ldots, m$ for $\ell = 1, \ldots, d$. We have that

$$w_{\underline{q}} = w_{q_1}^{(1)} \ldots w_{q_d}^{(d)}, \tag{3}$$

where $w_{q_\ell}^{(\ell)}$ is the weight associated to the q_ℓ-th node along the (univariate) ℓ-th direction of cell $C_{\underline{i}}$. In Equation (3), we do not need to label the weights with the cell index $\underline{i}$ since we assumed that the integration formula is the same in every cell, and so are the weights.

Finally, our composite cubature rule on the multidimensional mesh $\Omega_{\underline{n}}$ is obtained by applying the local cubature rule to all cells $C_{\underline{i}}$ of grid $\Omega_{\underline{n}}$ and adding the results, so that

$$Q_{\underline{n}}^m(f) := \sum_{\underline{i} \in \mathcal{I}_{\underline{n}}} \sum_{\underline{q} \in \mathcal{Q}_m} w_{\underline{q}} f(\mathbf{x}_{\underline{i},\underline{q}}) = \sum_{\underline{i} \in \mathcal{I}_{\underline{n}}} \sum_{q_1=1}^{m} \ldots \sum_{q_d=1}^{m} w_{q_1}^{(1)} \ldots w_{q_d}^{(d)} f(x_{i_1,q_1}^{(1)}, \ldots, x_{i_d,q_d}^{(d)}), \tag{4}$$

where we take the summation over all the possible instances of the multi-indices $\underline{i}$ and $\underline{q}$.

2.2. Accuracy of Multidimensional Cubature Formulas on the Reference Cell

The numerical integration error that characterizes the composite numerical integration formula with $\underline{n}$ partitionings of Ω reads as:

$$E_{\underline{n}}(f) := |I(f) - Q_{\underline{n}}^m(f)|.$$

An estimate of such error is fundamental to assess the accuracy of our cubature formulas. We can estimate the accuracy of a multidimensional numerical integration rule if we know the accuracy of the univariate quadrature rule used in each space direction of the tensor product construction. Let $\widehat{\Omega} = [-1, 1]^d$ be the multidimensional reference cell, and $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_d)^T \in \mathbb{R}^d$ with $-1 \leq \xi_\ell \leq 1$ for $\ell = 1, \ldots, d$, the multidimensional position vector in $\widehat{\Omega}$. For all integers $\ell = 2, \ldots, d$, we consider the notation $\boldsymbol{\xi}_{(1,\ell-1)} = (\xi_1, \ldots, \xi_{\ell-1})$ and $\boldsymbol{\xi}_{(\ell,d)} = (\xi_\ell, \ldots, \xi_d)$, with the convention that $\boldsymbol{\xi}_{(1,1)} = (\xi_1)$ and $\boldsymbol{\xi}_{(d,d)} = (\xi_d)$. Using this notation, we introduce the partition $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_{\ell-1}, \xi_\ell, \ldots, \xi_d) = (\boldsymbol{\xi}_{(1,\ell-1)}, \boldsymbol{\xi}_{(\ell,d)})$. Similarly, we introduce the notation $\underline{i}_{(1,\ell-1)} = (i_1, \ldots, i_{\ell-1})$ and $\underline{i}_{(\ell,d)} = (i_\ell, \ldots, i_d)$. Again, for consistency, we say that $\underline{i}_{(1,1)} = (i_1)$, $\underline{i}_{(d,d)} = (i_d)$, and we split the d-dimensional multi-index $\underline{i}$ as $\underline{i} = (i_1, \ldots, i_{\ell-1}, i_\ell, \ldots, i_d) = (\underline{i}_{(1,\ell-1)}, \underline{i}_{(\ell,d)})$. Next, we consider the reduced domain of integration $\widehat{\Omega}^{(\ell)} = [-1, 1]^{d-(\ell-1)} \subset \mathbb{R}^{d-(\ell-1)}$, which contains all the vectors $\boldsymbol{\xi}_{(\ell,d)}$ such that $(\mathbf{0}, \boldsymbol{\xi}_{(\ell,d)}) \in \widehat{\Omega}$, and the reduced set of integration indices $\mathcal{I}_{\ell,d}$ that contains all $(d - (\ell - 1))$-sized multi-indices of the form $\underline{i} = (i_\ell, \ldots, i_d)$. Finally, we set $dV(\boldsymbol{\xi}_{(1,\ell-1)}) = d\xi_1 \ldots d\xi_{\ell-1}$, and introduce the *partial numerical integration formula* for the function $\widehat{f} : \widehat{\Omega} \to \mathbb{R}$:

$$Q_{(\ell,d)}(\widehat{f})(\boldsymbol{\xi}_{(1,\ell-1)}) = \sum_{q_\ell=1}^{m} \ldots \sum_{q_d=1}^{m} \eta_{q_\ell}^{(\ell)} \ldots \eta_{q_d}^{(d)} \widehat{f}(\boldsymbol{\xi}_{(1,\ell-1)}, \xi_{q_\ell}^{(\ell)}, \ldots, \xi_{q_d}^{(d)}),$$

where we apply the quadrature formula in all directions from ℓ to d. Finally, we define the *partial cubature rule* that integrates $\widehat{f}$ exactly on the first $\ell - 1$ directions and numerically along the remaining ones from ℓ to d, so that for $1 < \ell \leq d$ we write that

$$I_{\ell-1}(\widehat{f}) := \int_{\widehat{\Omega}^{(\ell-1)}} Q_{(\ell,d)}(\widehat{f})(\boldsymbol{\xi}_{(1,\ell-1)})\, dV(\boldsymbol{\xi}_{(1,\ell-1)}). \tag{5}$$

We complete this definition by setting $I_0(\widehat{f}) = Q_{\underline{n}}^m(\widehat{f})$ and $I_d(\widehat{f}) = I(\widehat{f})$. To ease the notation, hereafter we will write I_ℓ instead of $I_\ell(\widehat{f})$. The integration error on $\widehat{\Omega}$ is the telescopic sum

$$\widehat{E}(\widehat{f}) := |I_d - I_0| = \left| \sum_{\ell=1}^{d} \left(I_\ell - I_{\ell-1} \right) \right| \leq \sum_{\ell=1}^{d} |I_\ell - I_{\ell-1}|.$$

Since it holds that $\boldsymbol{\xi}_{(1,\ell)} = (\boldsymbol{\xi}_{(1,\ell-1)}, \xi_\ell)$, and $dV(\boldsymbol{\xi}_{(1,\ell)}) = dV(\boldsymbol{\xi}_{(1,\ell-1)})\, d\xi_\ell$, we split the integral over $\widehat{\Omega}^{(\ell)} = \widehat{\Omega}^{(\ell-1)} \times [-1,1]$ as follows

$$\begin{aligned}\int_{\widehat{\Omega}^{(\ell)}} \left[\dots \right] dV(\boldsymbol{\xi}_{(1,\ell)}) &= \int_{\widehat{\Omega}^{(\ell-1)} \times [-1,1]} \left[\dots \right] dV(\boldsymbol{\xi}_{(1,\ell-1)}) d\xi_\ell \\ &= \int_{\widehat{\Omega}^{(\ell-1)}} \left(\int_{\xi_\ell=-1}^{\xi_\ell=1} \left[\dots \right] d\xi_\ell \right) dV(\boldsymbol{\xi}_{(1,\ell-1)}).\end{aligned}$$

Using this splitting, we find that

$$\begin{aligned}
I_\ell - I_{\ell-1} &= \int_{\widehat{\Omega}^{(\ell)}} Q_{(\ell+1,d)}(\widehat{f})(\boldsymbol{\xi}_{(1,\ell)})\, dV(\boldsymbol{\xi}_{(1,\ell)}) - \int_{\widehat{\Omega}^{(\ell-1)}} Q_{(\ell,d)}(\widehat{f})(\boldsymbol{\xi}_{(1,\ell-1)})\, dV(\boldsymbol{\xi}_{(1,\ell-1)}) \\
&= \sum_{q_{\ell+1}=1}^{m} \cdots \sum_{q_d=1}^{m} \eta_{q_{\ell+1}}^{(\ell+1)} \cdots \eta_{q_d}^{(d)} \int_{\widehat{\Omega}^{(\ell)}} \widehat{f}(\boldsymbol{\xi}_{(1,\ell)}, \xi_{q_{\ell+1}}^{(\ell+1)}, \dots, \xi_{q_d}^{(d)})\, dV(\boldsymbol{\xi}_{(1,\ell)}) \\
&\quad - \sum_{q_\ell=1}^{m} \cdots \sum_{q_d=1}^{m} \eta_{q_\ell}^{(\ell)} \cdots \eta_{q_d}^{(d)} \int_{\widehat{\Omega}^{(\ell-1)}} \widehat{f}(\boldsymbol{\xi}_{(1,\ell-1)}, \xi_{q_\ell}^{(\ell)}, \dots, \xi_{q_d}^{(d)})\, dV(\boldsymbol{\xi}_{(1,\ell-1)}) \\
&= \sum_{q_{\ell+1}=1}^{m} \cdots \sum_{q_d=1}^{m} \eta_{q_{\ell+1}}^{(\ell+1)} \cdots \eta_{q_d}^{(d)} \int_{\widehat{\Omega}^{(\ell-1)} \times [-1,1]} \widehat{f}(\boldsymbol{\xi}_{(1,\ell-1)}, \xi_\ell, \xi_{q_{\ell+1}}^{(\ell+1)}, \dots, \xi_{q_d}^{(d)})\, dV \boldsymbol{\xi}_{(1,\ell-1)} d\xi_\ell \\
&\quad - \sum_{q_\ell=1}^{m} \sum_{q_{\ell+1}=1}^{m} \cdots \sum_{q_d=1}^{m} \eta_{q_\ell}^{(\ell)} \eta_{q_{\ell+1}}^{(\ell+1)} \cdots \eta_{q_d}^{(d)} \int_{\widehat{\Omega}^{(\ell-1)}} \widehat{f}(\boldsymbol{\xi}_{(1,\ell-1)}, \xi_{q_\ell}^{(\ell)}, \xi_{q_{\ell+1}}^{(\ell+1)}, \dots, \xi_{q_d}^{(d)})\, dV(\boldsymbol{\xi}_{(1,\ell-1)}) \\
&= \sum_{q_{\ell+1}=1}^{m} \cdots \sum_{q_d=1}^{m} \eta_{q_{\ell+1}}^{(\ell+1)} \cdots \eta_{q_d}^{(d)} \int_{\widehat{\Omega}^{(\ell-1)}} \left[\int_{[-1,1]} \widehat{f}(\boldsymbol{\xi}_{(1,\ell-1)}, \xi_\ell, \xi_{q_{\ell+1}}^{(\ell+1)}, \dots, \xi_{q_d}^{(d)}) d\xi_\ell \right. \\
&\qquad \left. - \sum_{q_\ell=1}^{m} \eta_{q_\ell}^{(\ell)} \widehat{f}(\boldsymbol{\xi}_{(1,\ell-1)}, \xi_{q_\ell}^{(\ell)}, \xi_{q_{\ell+1}}^{(\ell+1)}, \dots, \xi_{q_d}^{(d)}) \right] dV(\boldsymbol{\xi}_{(1,\ell-1)}).
\end{aligned}$$

The univariate quadrature rule $\{(\xi_q, \eta_q)\}_{q=1}^m$ is applied in the ℓ-th direction to the integral inside the brackets in the last step. Let $\widehat{\mathcal{E}}_\ell(\widehat{f})$ be an upper bound on the integration error along direction ℓ, clearly dependent on the regularity of $\widehat{f}$, so that

$$\left| \int_{[-1,1]} \widehat{f}(\boldsymbol{\xi}_{(1,\ell-1)}, \xi_\ell, \xi_{q_{\ell+1}}^{(\ell+1)}, \dots, \xi_{q_d}^{(d)}) d\xi_\ell - \sum_{q_\ell=1}^{m} \eta_{q_\ell}^{(\ell)} \widehat{f}(\boldsymbol{\xi}_{(1,\ell-1)}, \xi_{q_\ell}^{(\ell)}, \xi_{q_{\ell+1}}^{(\ell+1)}, \dots, \xi_{q_d}^{(d)}) \right| \leq \widehat{\mathcal{E}}_\ell(\widehat{f}),$$

for all points $\boldsymbol{\xi}_{(1,\ell-1)} \in \widehat{\Omega}^{(\ell-1)}$ and all sets of quadrature nodes along the remaining $(d-\ell)$ directions $\{\xi_{q_{\ell+1}}^{(\ell+1)}\}, \dots, \{\xi_{q_d}^{(d)}\}$. Since the quadrature weights are positive (see Remark 1 below), we find that

$$|I_\ell - I_{\ell-1}| \leq \widehat{\mathcal{E}}_\ell(\widehat{f}) \sum_{q_{\ell+1}=1}^{m} \cdots \sum_{q_d=1}^{m} \eta_{q_{\ell+1}}^{(\ell+1)} \cdots \eta_{q_d}^{(d)} \int_{\widehat{\Omega}^{(\ell-1)}} dV(\boldsymbol{\xi}_{(1,\ell-1)}) = 2^{d-1} \widehat{\mathcal{E}}_\ell(\widehat{f}),$$

since for $t = \ell + 1, \ldots, d$ we have the $(d - \ell)$ summations $\sum_{q_t=1}^{m_t} \eta_{q_t}^{(t)} = |[-1,1]| = 2$ and it holds that

$$\int_{\widehat{\Omega}^{(\ell-1)}} dV(\boldsymbol{\xi}_{(1,\ell-1)}) = |\widehat{\Omega}^{(\ell-1)}| = 2^{\ell-1},$$

$|\widehat{\Omega}^{(\ell-1)}|$ being the $(\ell-1)$-dimensional measure of $\widehat{\Omega}^{(\ell-1)}$ and $|[-1,1]| = 2$ the one-dimensional measure of the reference interval $[-1,1]$. An upper bound on the error of the multidimensional reference domain $\widehat{\Omega}$ is thus given by

$$\widehat{E}(\widehat{f}) \leq \sum_{\ell=1}^{d} |I_\ell - I_{\ell-1}| \leq 2^{d-1} \sum_{\ell=1}^{d} \widehat{\mathcal{E}}_\ell(\widehat{f}). \tag{6}$$

In view of (6), we only need to know $\widehat{\mathcal{E}}_\ell(\widehat{f})$ for $\ell = 1, \ldots, d$, i.e., the accuracy of the univariate quadrature in every direction to estimate the error of the multidimensional cubature formula. Let $\widehat{\mathcal{E}}(\widehat{f}) = \max_\ell \widehat{\mathcal{E}}_\ell(\widehat{f})$. Then, we find that

$$\widehat{E}(\widehat{f}) \leq C(d)\widehat{\mathcal{E}}(\widehat{f}), \tag{7}$$

where the constant C depends on d but is independent of m.

Remark 1 (Weight positivity)**.** *The weights of the Clenshaw–Curtis quadratures can be computed using the formulas given in [36] Equations (4) and (5), and it is immediate to deduce that they are all positive. A straightforward argument allows us to prove that the weights of the Gauss–Legendre quadratures must also be strictly positive. Consider the polynomial*

$$p_i(\xi) = \Pi_{\substack{1 \leq j \leq m \\ j \neq i}} (\xi - \xi_j)^2.$$

Its integral over $[-1,1]$ *must be strictly positive, independently of the number of points* m*, since* $p_i(\xi) \geq 0$ *for all* $\xi \in [-1,1]$*. The* m*-point Gauss–Legendre quadrature rule* Q^m *integrates* $p_i(\xi)$ *exactly since* $p_i(\xi)$ *is a polynomial of degree* $2(m-1)$*. Noting that* $p_i(\xi_j) = 0$ *for all* $j \neq i$ *by construction, we find that*

$$0 < \int_{-1}^{1} p_i(\xi) d\xi = \sum_{j=1}^{m} \eta_j p_i(\xi_j) = \eta_i p_i(\xi_i).$$

This relation implies that $\eta_j > 0$ *since* $p_i(\xi_i) > 0$*, and the positivity of all integration weights follows by repeating this argument for all* i *from* 1 *to* m*. The weights for the Clenshaw-Curtis* ($m = 2, 3, 4$) *and Gauss-Legendre* ($m = 1, 2, 3$) *formulas used for the numerical experiments of Section 4 are reported in Appendix A.*

Finally, we note the weights of two Newton–Cotes formulas used in this work, i.e., the trapezoidal and Simpson quadrature rule, are positive. However, the weight positivity is not a property of this family of quadrature rules, so we can apply the argument discussed in this section to the trapezoidal and Simpson quadrature rule but not to all the Newton–Cotes formulas.

For the cubature families built upon the Newton–Cotes (trapezoidal and Simpson) formulas, Clenshaw–Curtis formulas, and Gauss–Legendre formulas, the upper bound $\widehat{\mathcal{E}}_\ell(\widehat{f})$ depends on the number of quadrature nodes m and the regularity of $\widehat{f}$. We summarize these results, which are easily available from the literature (see [16–18,37]), in the following theorems. We assume that the univariate integrand function $\widehat{f}(\xi)$ and its first ν derivatives $\widehat{f}^{(\nu)}(\xi)$, up to some given integer $\nu > 0$, have (at least) the regularity that is mentioned in each theorem's assertion.

Theorem 1 (Newton–Cotes formulas for $m = 2, 3$)**.**

- *Let $\widehat{f} \in C^2(-1,1)$; then, the Newton–Cotes formula with two integration nodes ($m = 2$, trapezoidal rule) satisfies*

$$\widehat{\mathcal{E}}(\widehat{f}) \leq \frac{2}{3}|\widehat{f}^{(2)}(\zeta)| \tag{8}$$

  for some point $\zeta \in (-1,1)$.
- *Let $\widehat{f} \in C^4(-1,1)$; then, the Newton–Cotes formula with three integration nodes ($m = 3$, Simpson rule) satisfies*

$$\widehat{\mathcal{E}}(\widehat{f}) \leq \frac{16}{45}|\widehat{f}^{(4)}(\zeta)| \tag{9}$$

  for some point $\zeta \in (-1,1)$.

Proof. The theorem statements follow from [37] (Equations (9.12) and (9.16)), by setting $h = 2$ for the size of the reference one-dimensional integration cell $(-1,1)$. □

Theorem 2 (Clenshaw–Curtis formulas)**.** *Let the univariate integrand function $\widehat{f}(\xi)$ and its derivatives $\widehat{f}^{(\nu-1)}(\xi)$ be absolutely continuous on the integration domain $[-1,1]$. Let the ν-th derivative $\widehat{f}^{(\nu)}(\xi)$ have bounded variation $\mathcal{V}(\widehat{f}^{(\nu)})$ for some integer number $\nu \geq 1$. Then, the m-point Clenshaw–Curtis quadrature applied to $\widehat{f}$ on the integration domain $[-1,1]$ satisfies*

$$\widehat{\mathcal{E}}(\widehat{f}) \leq C\frac{\mathcal{V}(\widehat{f}^{(\nu)})}{\pi\nu(m-1-\nu)^{\nu}} \qquad \text{for } m > 1+\nu, \tag{10}$$

where C is a constant independent of m.

Proof. The theorem statement is adapted from [38] (Theorem 5.1) and the first statement of [18] (Theorem 19.4) by setting the number of quadrature nodes to m. □

Theorem 3 (Gauss–Legendre formulas)**.** *Let the univariate integrand function $\widehat{f}(\xi)$ and its derivatives $\widehat{f}^{(\nu-1)}(\xi)$ be absolutely continuous on the integration domain $[-1,1]$. Let the ν-th derivative $\widehat{f}^{(\nu)}(\xi)$ have bounded variation $\mathcal{V}(\widehat{f}^{(\nu)})$ for some integer number $\nu \geq 1$. Then, the m-point Gauss–Legendre quadrature applied to $\widehat{f}$ on the integration interval $[-1,1]$ satisfies*

$$\widehat{\mathcal{E}}(\widehat{f}) \leq C\frac{\mathcal{V}(\widehat{f}^{(\nu)})}{\pi\nu(m-2\nu-2)^{2\nu+1}} \qquad \text{for } m > 2\nu+2, \tag{11}$$

where C is a constant independent of m.

Proof. The theorem statement is adapted from [38] Theorem 5.1 and the second statement of [18] Theorem 19.4 by setting the number of quadrature nodes to m. □

2.3. Accuracy of Composite Cubature Formulas for Multidimensional Numerical Integration

Applying a scaling argument to a local estimate of the approximation error and summing over all the mesh cells, we obtain the upper bound on the integration error of the Newton–Cotes schemes. The error estimates depend on the regularity of $f(\mathbf{x})$. We assume that f in each formula has (at least) the regularity shown by its right-hand side infinity norm $||\ldots||_{\infty}$. We denote again the partial derivatives of order ν of the scalar function f by $f^{(\nu)}$.

Theorem 4 (Convergence of composite quadrature formulas)**.**
Let $f : \Omega \to \mathbb{R}$, and $E_{\underline{n}}(f) := |I(f) - Q^m_{\underline{n}}(f)|$. Under the same assumptions of Theorem 1, it holds that

- ***Composite Newton–Cotes formula*** *for $m = 2 : E_{\underline{n}}(f) \leq Ch^2||f^{(2)}||_\infty$, (trapezoidal rule);*
- ***Composite Newton–Cotes formula*** *for $m = 3 : E_{\underline{n}}(f) \leq Ch^4||f^{(4)}||_\infty$, (Simpson rule).*

Moreover, under the assumption that $f \in C^m$, it holds that

- ***Composite Clenshaw–Curtis formulas*** *for $m \geq 2 : E_{\underline{n}}(f) \leq Ch^m||f^{(m)}||_\infty$;*

and under the assumption that $f \in C^{2m}$, it holds that

- ***Composite Gauss–Legendre formulas*** *for $m \geq 2 : E_{\underline{n}}(f) \leq Ch^{2m}||f^{(2m)}||_\infty$.*

The positive constant factor C is independent of $\underline{n}$ (so, on the mesh size parameter h), but can depend on the number of dimensions d.

Proof. The theorem statements follow from an interpolation error estimate and a scaling argument. We split the one-dimensional domain of integration $[0, 1]$ into N equispaced subintervals $[x_k, x_k + h]$ of size $h = 1/N$, so that $x_0 = 0$ and $x_N = 1$. Accordingly, we decompose the integral of $f(x)$ and its numerical approximation via the m-node quadrature rule as the sum of local contributions over the N subintervals:

$$I(f) := \int_0^1 f(x)\,dx = \sum_{k=0}^{N-1} \int_{x_k}^{x_k+h} f(x)\,dx = \sum_{k=0}^{N-1} I_k(f), \tag{12}$$

$$Q^m(f) := \sum_{k=0}^{N-1} \sum_{q=1}^{m} w_q f(x_{k,q}) = \sum_{k=0}^{N-1} Q^m_k(f). \tag{13}$$

Then, we remap every subinterval $[x_k, x_k + h]$ onto the reference interval $[-1, 1]$. Thus, for all $x \in [x_k, x_k + h]$ and $\xi \in [-1, 1]$ such that $x(\xi) = x_k + \frac{h}{2}(\xi + 1)$ it holds that $f(x(\xi)) = \widehat{f}_k(\xi)$ (the subindex "k" in the symbol "$\widehat{f}_k$" indicates that this is the image of the restriction of f to $[x_k, x_k + h]$ through the coordinate mapping $x(\xi)$). For any positive integer ν, a straightforward calculation shows that

$$\frac{\partial^\nu \widehat{f}_k(\xi)}{\partial \xi^\nu} = \frac{\partial^\nu f(x)}{\partial x^\nu} \left(\frac{\partial x(\xi)}{\partial \xi}\right)^\nu = \left(\frac{h}{2}\right)^\nu \frac{\partial^\nu f(x)}{\partial x^\nu}. \tag{14}$$

We rewrite the exact and approximate integration formulas on the reference interval as follows:

$$I(f_k) = \frac{h}{2} \int_{-1}^{1} \widehat{f}_k(\xi) d\xi =: \frac{h}{2} \widehat{I}(\widehat{f}_k) \quad \text{and} \quad Q^m(f_k) = \frac{h}{2} \sum_q \eta_q \widehat{f}_k(\xi_q) =: \frac{h}{2} \widehat{Q}^m(\widehat{f}_k),$$

where the summation is taken on all the quadrature nodes (and using obvious definitions of the functionals "$\widehat{I}(\cdot)$" and "$\widehat{Q}^m(\cdot)$"). We bound the error on the k-th cell as follows

$$E_k(f) := |I_k(f) - Q^m_k(f)| = \frac{h}{2} \left|\widehat{I}(\widehat{f}_k) - \widehat{Q}^m(\widehat{f}_k)\right| =: \frac{h}{2} \widehat{\mathcal{E}}(\widehat{f}_k).$$

To estimate $\widehat{\mathcal{E}}(\widehat{f}_k)$, we introduce the polynomial interpolation $\mathcal{I}_m \widehat{f}_k(\xi)$ of the integrand function $\widehat{f}_k(\xi)$ at the m quadrature nodes on the reference interval $[-1, 1]$. This interpolation is the unique polynomial of degree $m - 1$ in ξ that takes the same values of $\widehat{f}_k$ at the m quadrature nodes ξ_q, i.e., $\widehat{f}_k(\xi_q) = \mathcal{I}_m \widehat{f}_k(\xi_q)$ for $q = 1, \ldots, m$. If we apply $\widehat{Q}^m$ to $\widehat{f}_k$ and $\mathcal{I}_m \widehat{f}_k$, we obtain the same integral approximation because the quadrature rule depends only on the quadrature node values $\widehat{f}_k(\xi_q) = \mathcal{I}_m \widehat{f}_k(\xi_q)$. Moreover, the quadrature rule $\widehat{Q}^m$

is of interpolatory type and is exact on all polynomials of degree (up to) $m-1$. Therefore, we find that

$$\widehat{Q}^m(\widehat{f}_k) = \widehat{Q}^m\big(\mathcal{I}_m\widehat{f}_k(\xi)\big) = \int_{-1}^{1} \mathcal{I}_m\widehat{f}_k(\xi)d\xi.$$

Using the formula above, we immediately find that

$$\widehat{\mathcal{E}}(\widehat{f}_k) = \left|\int_{-1}^{1} \widehat{f}_k(\xi)d\xi - \widehat{Q}^m(\widehat{f}_k)\right| = \left|\int_{-1}^{1} \big(\widehat{f}_k(\xi) - \mathcal{I}_m\widehat{f}_k(\xi)\big)d\xi\right| \leq \int_{-1}^{1} \big|\widehat{f}_k(\xi) - \mathcal{I}_m\widehat{f}_k(\xi)\big|d\xi$$

$$\leq \sup_{\xi\in(-1,1)} \big|\widehat{f}_k(\xi) - \mathcal{I}_m\widehat{f}_k(\xi)\big| \int_{-1}^{1} d\xi = 2||\widehat{f}_k - \mathcal{I}_m\widehat{f}_k||_{\infty,(-1,1)},$$

where $||\widehat{f}_k - \mathcal{I}_m\widehat{f}_k||_{\infty,(-1,1)}$ is the interpolation error over the reference cell $(-1,1)$. The regularity assumptions of the theorem imply that $\widehat{f}_k$ is ν times continuously differentiable for some integer $\nu > 0$, whose value changes according to the quadrature formula. From standard arguments of polynomial interpolation theory, we know that $||\widehat{f}_k - \mathcal{I}_m\widehat{f}_k||_{\infty,(-1,1)} \leq C||\widehat{f}_k^{(\nu)}||_{\infty,(-1,1)}$, where $||\widehat{f}_k^{(\nu)}||_{\infty,(-1,1)}$ denotes the ∞-norm of $\widehat{f}_k^{(\nu)}(\xi)$, the ν-th derivative of $\widehat{f}_k(\xi)$ on $(-1,1)$, and C is a real, positive constant that may depend on the number of quadrature nodes m. Using the derivative scaling (14), we find that

$$\widehat{\mathcal{E}}(\widehat{f}_k) \leq C||\widehat{f}^{(\nu)}||_{\infty,(-1,1)} \leq Ch^\nu||f^{(\nu)}||_{\infty,k},$$

where $||f^{(\nu)}||_{\infty,k} = \sup_{x\in(x_k,x_k+h)} |f^{(\nu)}(x)|$ denotes the ∞-norm of $f^{(\nu)}(x)$ on (x_k, x_k+h). Here, C is a constant independent of h, but that may still depend on the number of quadrature nodes m. Summing over all the cells and noting that $hN = 1$, we find the following estimate for the integration error in each direction:

$$E(f) = \sum_k E_k(f) \leq \max_k E_k(f)\, N \leq \max_k \widehat{\mathcal{E}}(\widehat{f}_k)\frac{h}{2}N \leq Ch^\nu \max_k ||f^{(\nu)}||_{\infty,k} \leq Ch^\nu||f^{(\nu)}||_\infty.$$

The theorem's assertions follow on first setting $\nu = 2$ for the trapezoidal rule and $\nu = 4$ for the Simpson rule (in accordance with Theorem 1), and setting $\nu = m$ for the Clenshaw–Curtis formulas and $\nu = 2m$ for the Gauss–Legendre formula (in accordance with the theorem's regularity assumptions), and then taking the maximum over all the directions. □

Remark 2. *The estimate for the Gauss–Legendre formulas can also be derived in a straightforward way by using the formula for "R_n" in [39] (Example 5.2, page 146) and setting directly "$b - a = h$" (according to the notation used therein). Note, however, that the derivation of such a formula is not presented in [39].*

3. Low-Rank TT Cross Approximation and Numerical Integration of a Multidimensional Function

In this section, we briefly review the low-rank tensor-train approximation of a multidimensional grid function and its application to the numerical integration. An extensive presentation and discussion of these concepts and algorithms can be found in [21,22].

3.1. Tensor-Train Decomposition: Basic Definitions and Properties

A d-order tensor $A \in \mathbb{R}^{n_1\times\cdots n_d}$ is a multidimensional array whose entries are addressed by d indices as

$$A := \big[A(i_1,\ldots,i_d)\big], \quad i_\ell = 1,\ldots,n_\ell, \quad \ell = 1,\ldots,d,$$

according to a notation used in MATLAB@ [40] and the Python programming language [41]. Hereafter, we shall refer to the integers $\underline{n} = (n_1, \ldots, n_d)$ as the *mode sizes*. The set of all d-dimensional arrays with the arithmetic operations of addition and multiplication by a scalar number, which we assume to be defined in the natural way, is a linear space. We endow such a linear space with the inner (dot) product and induced norm

$$\langle A, B \rangle = \sum_{i_1,\ldots,i_d} A(i_1, \ldots, i_d)\, B(i_1, \ldots, i_d), \qquad ||A||_F^2 = \langle A, A \rangle, \tag{15}$$

for all A, $B \in \mathbb{R}^{n_1 \times \ldots n_d}$, and where the summation runs over the set of all admissible values of the multi-index $(i_1, \ldots, i_d)$. The norm $||A||_F$ is usually called the *Frobenius norm* of A.

We say that the d-order tensor A admits a *tensor-train representation* if there exists a $(d+1)$-tuple of strictly positive integers $(r_0, r_1, \ldots r_d)$, with $r_0 = r_d = 1$, and d three-dimensional tensors $G_\ell \in \mathbb{R}^{r_{\ell-1} \times n_\ell \times r_\ell}$, $\ell = 1, \ldots, d$, such that

$$A(i_1, \ldots, i_d) = \sum_{\alpha_0=1}^{r_0} \sum_{\alpha_1=1}^{r_1} \ldots \sum_{\alpha_d=1}^{r_d} G_1(\alpha_0, i_1, \alpha_1) G_2(\alpha_1, i_2, \alpha_2) \ldots G_d(\alpha_{d-1}, i_d, \alpha_d), \tag{16}$$

for all $i_\ell = 1, \ldots, n_\ell$, $\ell = 1, \ldots, d$. The tensors G_ℓ are called the *TT cores* and the integers r_ℓ the *TT ranks* (also known as the *compression ranks*) of the tensor-train representation. Since $r_0 = r_d = 1$ the tensors G_1 and G_d are two-dimensional arrays, i.e., matrices, and the right-hand side of (16) is a scalar number. Moreover, each factor $G_\ell(:, i_\ell, :)$ for any fixed $i_\ell = 1, \ldots, n_\ell$ is a matrix with size $r_{\ell-1} \times r_\ell$, so that we can reformulate (16) in the compact matrix multiplication form given by

$$A(i_1, \ldots, i_d) = G_1(:, i_1, :) \ldots G_d(:, i_d, :),$$

which allows for a faster implementation using BLAS level-3 routines [42].

The storage of the tensor-train representation of A requires $\sum_\ell n_\ell r_{\ell-1} r_\ell$ real numbers. If n and r are convenient upper bounds of all the mode sizes and TT ranks, so that $n_\ell \leq n$ and $r_\ell \leq r$ for all ℓ, the storage is proportional to $\mathcal{O}(dnr^2)$ instead of $\mathcal{O}(n^d)$ of the full format representation.

The tensor-train representation of a tensor A is related to an exact or approximate skeleton decomposition of its unfolding matrices A_ℓ, $\ell = 1, \ldots, d$, cf. [21]. The ℓ-th unfolding matrix A_ℓ is defined by suitably remapping the entries of A into a matrix with size $(n_1 \ldots n_\ell) \times (n_{\ell+1} \ldots n_d)$. To this end, we consider the integer-valued functions $\varphi_\ell(i_1, \ldots, i_\ell)$, $\ell = 1, \ldots, d-1$, which are iteratively defined as

$$\begin{aligned} \varphi_1(i_1) &= i_1, \\ \varphi_\ell(i_1, \ldots, i_\ell) &= \varphi_{\ell-1}(i_1, \ldots, i_{\ell-1}) + \left(\prod_{k=1}^{\ell-1} n_k \right) (i_\ell - 1) \quad \forall \ell = 2, \ldots, d-1. \end{aligned}$$

Then, we set

$$A_\ell(\varphi_\ell(i_1, \ldots i_\ell), \varphi_{d-\ell}(i_{\ell+1}, \ldots i_d)) := A(i_1, \ldots, i_d). \tag{17}$$

The existence of an exact TT decomposition of a tensor $A \in \mathbb{R}^{n_1 \times \ldots n_d}$ with compression ranks $r_1, \ldots, r_{d-1}$ equal to the ranks of its matrix unfoldings $A_1, \ldots, A_{d-1}$ is proved in [21]. Such compression ranks are the smallest possible ones among all possible exact tensor-train decompositions of A. The problem of computing approximate tensor-train decompositions with smaller ranks is addressed in [22]. In particular, if all the unfolding matrices A_ℓ of tensor A can be approximated by rank-r_ℓ matrices $\widetilde{A}_\ell$ through an approximate skeleton decomposition with accuracy ϵ_ℓ so that

$$||A_\ell - \widetilde{A}_\ell||_F \leq \epsilon_\ell \qquad \ell = 1, 2, \ldots, d-1, \tag{18}$$

then, there exists an approximate tensor $\widetilde{A}$ with compression ranks r_ℓ satisfying

$$||A - \widetilde{A}||_F^2 \leq \sqrt{d-1} \sum_{\ell=1}^{d-1} \epsilon_\ell^2. \tag{19}$$

In the numerical experiments of Section 4, we shall consider the low-rank tensor-train approximation of the grid functions given by using the TT cross interpolation of multidimensional arrays proposed in [22]. The TT cross-skeletal decomposition allows us to approximate a multivariate function using only a few entries of the tensor array resulted from the function sampling at the grid nodes, thus avoiding working with the array containing the full multidimensional grid function.

3.2. Numerical Integration of a Low-Rank Tensor-Train Representation of a Multidimensional Grid Function

For any mesh cell $C_{\underline{i}}$, we collect the nodes and weights defining the elemental multidimensional integration rule in the pair $\left(X_{\underline{i}}^m, W_{\underline{i}}^m\right)$, where $X_{\underline{i}}^m = \left\{\mathbf{x}_{\underline{i},\underline{q}}\right\}$ is the set of the quadrature nodes and $W_{\underline{i}}^m = \left\{w_{\underline{i},\underline{q}}\right\}$ is the set of quadrature weights. The global integration rule in (4) is associated with $(X_{\underline{n}}^m, W_{\underline{n}}^m)$, so that, for any cell index $\underline{i} \in \mathcal{I}_{\underline{n}}$, it holds that $X_{\underline{n}\,|C_{\underline{i}}}^m = X_{\underline{i}}^m$ and $W_{\underline{n}\,|C_{\underline{i}}}^m = W_{\underline{i}}^m$, where we denote the restriction of quantity $(*)$ to cell $C_{\underline{i}}$ as $(*)_{|C_{\underline{i}}}$. We represent the grid function given by sampling function f at the nodes of a generic d-dimensional grid X by the d-dimensional tensor $F_X = (F_X(j_1, \ldots, j_d))$. If $X = X_{\underline{n}}^m$, every index j_ℓ runs from 1 to mn_ℓ. The nodes of $X_{\underline{n}}^m$ are not equispaced if the local multidimensional cubature rule is built by taking the tensor product of univariate quadrature rules with non-equispaced nodes such as the Clenshaw–Curtis and Gauss–Legendre formulas. The integration formula (4) can be rewritten as the tensor inner product

$$Q_{\underline{n}}^m(f) = \langle F_X, W_{\underline{n}}^m \rangle. \tag{20}$$

Tensor $W_{\underline{n}}^m$ has compression ranks equal to 1, and if we know a TT decomposition of F_X with compression ranks r_ℓ, we compute the composite cubature rule $Q_{\underline{n}}^m(f)$ on mesh $\Omega_{\underline{n}}$ using Algorithm 1 with a computational cost proportional to $m \sum_{\ell=1}^d n_\ell r_\ell r_{\ell-1}$. Recalling again the uniform upper bounds $n_\ell \leq n$ and $r_\ell \leq r$, we find that the storage scales up like $\mathcal{O}(dmnr^2)$. The computational complexity is thus polynomial in the mode sizes and compression ranks, and under the (strong) assumption that the error ϵ_ℓ in (18) and (19) is small for all ℓ, the computational complexity may depend only linearly on the number of dimensions. We prove this result in the following theorem.

Algorithm 1: Tensor-Train Cubature Algorithm.

Data: Tensor train $\widetilde{F}_X = G_1 G_2 \ldots G_d$ and univariate quadrature weights w_l for $1 \leq l \leq d$
Result: Integral value I
$I = w_1 \times_2 G_1$ where $\times_2$ is the mode-2 tensor product
for $l = 2 : d$ **do**
  $\widehat{G_l} = w_l \times_2 G_l$; `/* ` $\widehat{G_l}$ `is an` $r_{l-1} \times r_l$ `matrix */`
  $I = I \times_2 \widehat{G_l}$
end

Theorem 5. *Let F_X admit the low-rank approximation $\widetilde{F}_X$ in the tensor-train format so that*

$$\widetilde{F}_X(j_1, j_2, \ldots, j_d) = \sum_{\alpha_1=1}^{r_1} \sum_{\alpha_2=1}^{r_2} \ldots \sum_{\alpha_{d-1}=1}^{r_{d-1}} G_1(1, j_1, \alpha_1) G_2(\alpha_1, j_2, \alpha_2) \ldots G_d(\alpha_{d-1}, j_d, 1),$$

where $r_\ell \leq r \lesssim n$, for $\ell = 1, \ldots, d-1$, are the tensor-train ranks and $j_\ell = 1, \ldots, mn_\ell$, for $\ell = 1, \ldots, d$, is the index running through all the mn_ℓ quadrature nodes of the $n_\ell \lesssim n$ intervals along the direction ℓ. Then, to compute the numerical cubature (20) using Algorithm 1 we need $\mathcal{O}\left(dr^2mnr + (d-1)r^2\right)$ saxpy operations.

Proof. Let w_ℓ be the one-dimensional mn_ℓ-sized vector collecting all the weights of all quadrature nodes along the direction ℓ. Then, we write the cubature rule (20) as

$$\begin{aligned} Q_{\underline{n}}^m(f) &= \sum_{\alpha_1=1}^{r_1} \sum_{\alpha_2=1}^{r_2} \ldots \sum_{\alpha_{d-1}=1}^{r_{d-1}} \left(w_1 \times_2 G_1(1, \cdot, \alpha_1)\right) \left(w_2 \times_2 G_2(\alpha_1, \cdot, \alpha_2)\right) \ldots \\ &\qquad \ldots \left(w_d \times_2 G_d(\alpha_{d-1}, \cdot, 1)\right) \\ &= \sum_{\alpha_1=1}^{r_1} \sum_{\alpha_2=1}^{r_2} \ldots \sum_{\alpha_{d-1}=1}^{r_{d-1}} \widetilde{G}_1(1, \alpha_1) \widetilde{G}_2(\alpha_1, \alpha_2) \ldots \widetilde{G}_d(\alpha_{d-1}, 1), \end{aligned}$$

where $\widetilde{G}_\ell(\alpha_{\ell-1}, \alpha_\ell) = w_\ell \times_2 G_\ell(\alpha_{\ell-1}, \cdot, \alpha_\ell)$ is the mode-2 product between vector w_ℓ and the tensor-train core tensor $G_\ell(\alpha_{\ell-1}, i_\ell, \alpha_\ell)$, i.e, the contraction along the ℓ-th mode index. For every pair of indices $(\alpha_{\ell-1}, \alpha_\ell)$ of the contracted core $\widetilde{G}_\ell(\alpha_{\ell-1}, \alpha_\ell)$, the contraction requires mn_ℓ saxpy operations, and since $\widetilde{G}_\ell$ is an $r_{\ell-1} \times r_\ell$-sized matrix, the contraction of all the d cores requires $dmn_\ell r_{\ell-1} r_\ell \leq dmnr^2$ of such operations. Eventually, the calculation proceeds as a sequence of vector-matrix product from left to right between a temporary intermediate one-dimensional $r_{\ell-1}$-sized array and the two-dimensional $r_{\ell-1} \times r_\ell$-sized matrix for $\ell = 2, \ldots d$, thus requiring other $\sum_{\ell=2}^{d} r_{\ell-1} r_\ell \leq (d-1)r^2$ *saxpy* operations. □

Remark 3. *If tensor F_X admits a low-rank approximation $\widetilde{F}_X$ with compression ranks $\widetilde{r}_\ell \leq \widetilde{r} \ll r$, we have the even better scaling proportional to $\mathcal{O}(dmn\widetilde{r}^2)$.*

Remark 4. *Using the* TT cross interpolation algorithm *for multidimensional arrays [22] makes it possible to compute such an approximate TT decomposition of F_X using only a very small fraction of the grid values of $f_{\underline{n}}(X_{\underline{n}}^m)$ and without pre-computing and storing the full tensor F_X. For the details of this algorithm and its practical implementation, we refer again to [22].*

We characterize the accuracy of this approximation by introducing the tolerance factor τ, which is such that $\tau \geq ||F_X - \widetilde{F}_X||_F$. The tolerance factor has clearly an influence on the ranks of the tensor-train approximation of the multidimensional tensor of the grid function values, e.g., F_X, and we can expect that a smaller tolerance implies bigger ranks. The interplay between tolerance τ and the ranks is crucial to obtain an efficient integration algorithm. If instead of computing $Q_{\underline{n}}^m(f)$, we compute $Q_{\underline{n}}^m(\widetilde{f})$, where $\widetilde{f}$ is an approximation of f represented by $\widetilde{F}_X$, and we use the linearity of $Q_{\underline{n}}^m(\cdot)$, we immediately find the bound on the integration error given by

$$\begin{aligned} E_{\underline{n}}(\widetilde{f}) &= |I(f) - Q_{\underline{n}}^m(\widetilde{f})| \leq |I(f) - Q_{\underline{n}}^m(f)| + |Q_{\underline{n}}^m(f) - Q_{\underline{n}}^m(\widetilde{f})| \\ &= E_{\underline{n}}(f) + |Q_{\underline{n}}^m(f - \widetilde{f})|. \end{aligned} \tag{21}$$

Assuming for simplicity that $h_\ell = h$ for $\ell = 1, \ldots, d$, so that we have the same number of points n corresponding to $n - 1 = 1/h$ partitions in every direction, we note that

$$||W_{\underline{n}}^m||_F^2 = \sum_{\underline{i} \in \mathcal{I}_{\underline{n}}} \sum_{\underline{q} \in \mathcal{Q}_m} |w_{\underline{q}}|^2 \leq (n-1)^d h^d = 1.$$

Using this fact, definition (20) and the Cauchy–Schwarz inequality, we find that

$$|Q_{\underline{n}}^m(f - \widetilde{f})| = \langle F_X - \widetilde{F}_X, W_{\underline{n}}^m \rangle \leq ||F_X - \widetilde{F}_X||_F \, ||W_{\underline{n}}^m||_F \leq \tau.$$

If $\tau \ll E_{\underline{n}}(f)$, we can neglect the last term of (21), and we find that $E_{\underline{n}}(\widetilde{f}) \approx E_{\underline{n}}(f)$, i.e., the convergence properties of the cubature remain essentially the same even if integrate $\widetilde{f}$ instead of f.

4. Numerical Experiments

The Newton–Cotes (trapezoidal and Simpson rules), Clenshaw–Curtis, and Gauss formulas are all of the interpolatory type. In fact, they can be derived by integrating the polynomial interpolant supported by the quadrature nodes and are characterized by the *polynomial degree* and the following exactness property [15]. For any integer $m \geq 0$,

- the Newton–Cotes formulas with $m = 2$ points (trapezoidal rule) and $m = 3$ points (Simpson's rule) integrate exactly the polynomials of degree 1 and 3, respectively;
- the m-point Clenshaw–Curtis formula integrates exactly all polynomials of degree up to $m - 1$;
- the m-point Gauss–Legendre formula integrates exactly all polynomials of degree up to $2m - 1$.

These exactness properties readily extend to the multidimensional compound cubature. In this section, we assess the performance of our strategy to build such cubatures by investigating the "h" and the "p" convergence of the method (using a terminology widely adopted in the finite element method). Here, $h = 1/N$ is the (uniform) step size of each univariate partition of the tensor product grid, and $N = n - 1$ is the number of intervals of every partition. Moreover, we set the tolerance factor $\tau = 10^{-16}$, and we note that the observed rank is typically between 5 and 10.

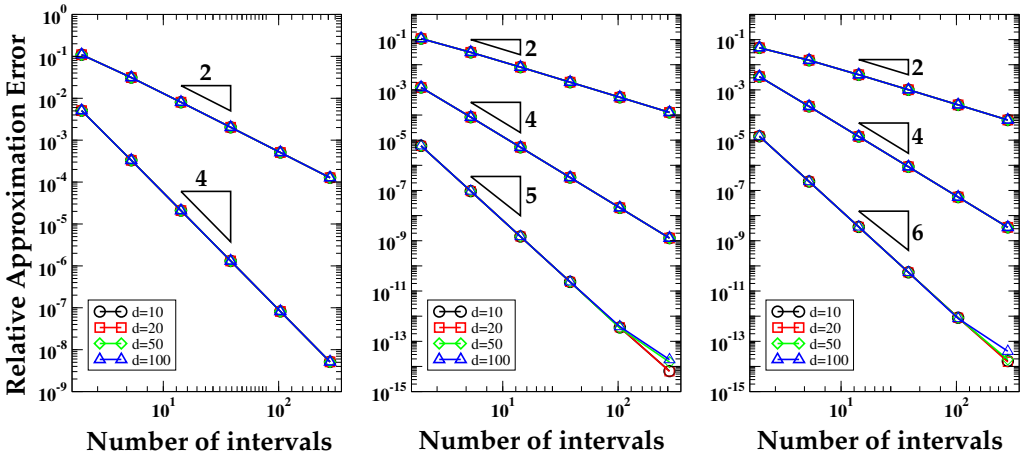
4.1. Test Case 1: h-Convergence of a Smooth Function

The three panels in Figure 2 show the log–log plot of the error curves versus the number of intervals per direction N for the numerical integration of the multidimensional function

- *Test Case 1:* $f(x) = \sin\left(1 + 2\pi \sum_{\ell=1}^{d} x_\ell\right) + \frac{1}{d} \sum_{\ell=1}^{d} x_\ell^{10}; \quad I(f) = 1/2 - 1/99,$

on the integration domain $\Omega = [0, 1]^d$.

We consider $d = 10, 20, 50, 100$. We perform the numerical integration using (i) the Newton–Cotes formulas (trapezoidal and Simpson quadrature rules); (ii) the Clenshaw–Curtis formula with $m = 2, 4, 5$; (iii) the Gauss–Legendre formula with $m = 1, 2, 3$. The convergence rate is expected to be proportional to N^{-2} (trapezoidal rule); N^{-4} (Simpson rule); N^{-m} (Clenshaw–Curtis rule); N^{-2m} (Gauss–Legendre rule), and is shown by the slope of the triangle close to each curve. The slopes of the error curves reflect the numerical orders of convergence, and Figure 2 shows that these slopes are in agreement with the theoretical expectations. We also note that the convergence behavior seems to be unaffected by the number of dimensions.



(i) *Newton–Cotes* ($m = 2, 3$) **(ii)** *Clenshaw–Curtis* ($m = 2, 4, 5$) **(iii)** *Gauss–Legendre* ($m = 1, 2, 3$)

Figure 2. Test Case 1. Relative approximation error curves versus the number of intervals per direction N for the numerical integration of a d-dimensional polynomial of degree 10, $d = 10, 20, 50, 100$, using **(i)** Newton–Cotes formulas (trapezoidal and Simpson quadrature rules); **(ii)** Clenshaw–Curtis formula with $m = 2, 4, 5$; **(iii)** Gauss–Legendre formulas with $m = 1, 2, 3$. The expected convergence rate is shown by the number close to the triangular shapes and is proportional to N^{-2} (trapezoidal rule); N^{-4} (Simpson rule); N^{-m} (Clenshaw–Curtis rule); N^{-2m} (Gauss–Legendre rule).

4.2. Test Cases 2 and 3: h-Convergence and Integration of Multidimensional Exponential Functions

In the second and third test cases, we consider the Gauss bell-shaped function and the exponential function, which we denote as *Genz-4* and *Genz-5* following the nomenclature introduced in [43,44]:

- *Test Case 2: Genz-4:* $f(\mathbf{x}) = \exp\left(-\sum_{\ell=1}^{d}(x_\ell - w_\ell)^2\right)$; $I(f) = \left[\sqrt{\pi}/2\mathrm{erf}(1)\right]^d$;
- *Test Case 3: Genz-5:* $f(\mathbf{x}) = \exp\left(-\sum_{\ell=1}^{d}|x_\ell - w_\ell|\right)$; $I(f) = \left[1 - 1/e\right]^d$.

In Figures 3 and 4, we show the log–log plots of the error curves versus the number of intervals N per direction for the numerical integration of the d-dimensional version of these functions. We consider $d = 10, 20, 50, 100$. We perform the numerical integration using the (i) trapezoidal formula ($m = 2$); (ii) Simpson formula ($m = 4$); (iii) Clenshaw–Curtis formula ($m = 4$); (iv) Gauss–Legendre formula ($m = 2$). The expected convergence rate, still reflected by the slope of the triangles near each curve, is expected to be proportional to N^{-2} (trapezoidal rule) and N^{-4} (Simpson, Clenshaw–Curtis, and Gauss–Legendre rules). As in the previous test case, the numerical slopes are in agreement with the theoretical slopes, still shown by the triangle in each plot. The number of dimensions plays, however, a more significant role than the previous test case, and the constant factor that appears in every error bound, see Theorem 4, changes with d. This behavior is the same for all the cubature schemes that we consider in this test case.

4.3. Test Case 4: p-Convergence and Integration of Multidimensional Exponential Functions

In Figure 5, we show the linear–log plots of the error curves that are obtained by applying the cubature rules built using the univariate Clenshaw–Curtis formula (right column) and the Gauss–Legendre formula (left column). The two panels on top refer to the numerical integration of the *Genz-4* function, while the two panels at the bottom refer to the numerical integration of the *Genz-5* function. We carry out all calculations on a very coarse grid with only two intervals per direction, so the resolution is $2 \times 2 \times \ldots \times 2$ (d times), and we have a total of 2^d multidimensional cells. The numerical integration is performed for $d = 10, 20, 50, 100, 200$. We build the cubature rules using the univariate Clenshaw–Curtis and Gauss–Legendre formulas with m integration nodes with $m = 1, 2, \ldots, 10$. The four

plots of Figure 5 show that the convergence of the Clenshaw–Curtis and Gauss–Legendre formula is (roughly) proportional to $m^{-\gamma}$ for some positive number γ.

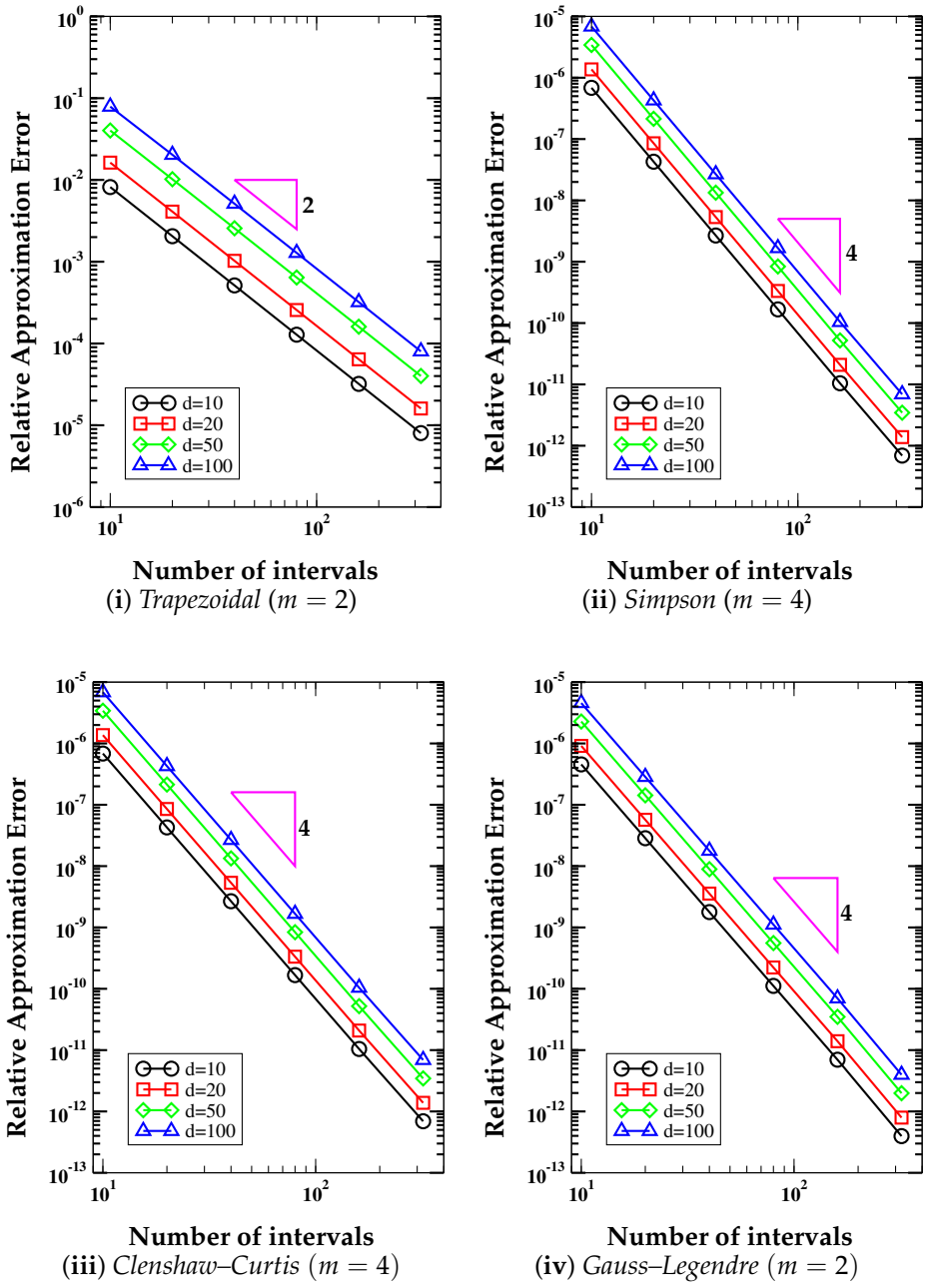


Figure 3. Test Case 2. Relative approximation error curves versus the number of intervals N per direction for the numerical integration of a d-dimensional Gaussian bell-shaped function (Genz-4), $d = 10, 20, 50, 100$, using the **(i)** trapezoidal formula ($m = 2$); **(ii)** Simpson formula ($m = 4$); **(iii)** Clenshaw–Curtis formula ($m = 4$); **(iv)** Gauss–Legendre formulas ($m = 2$). The expected convergence rate is shown by the number close to the triangular shapes and is proportional to N^{-2} (trapezoidal rule) and N^{-4} (Simpson, Clenshaw–Curtis, and Gauss–Legendre rules).

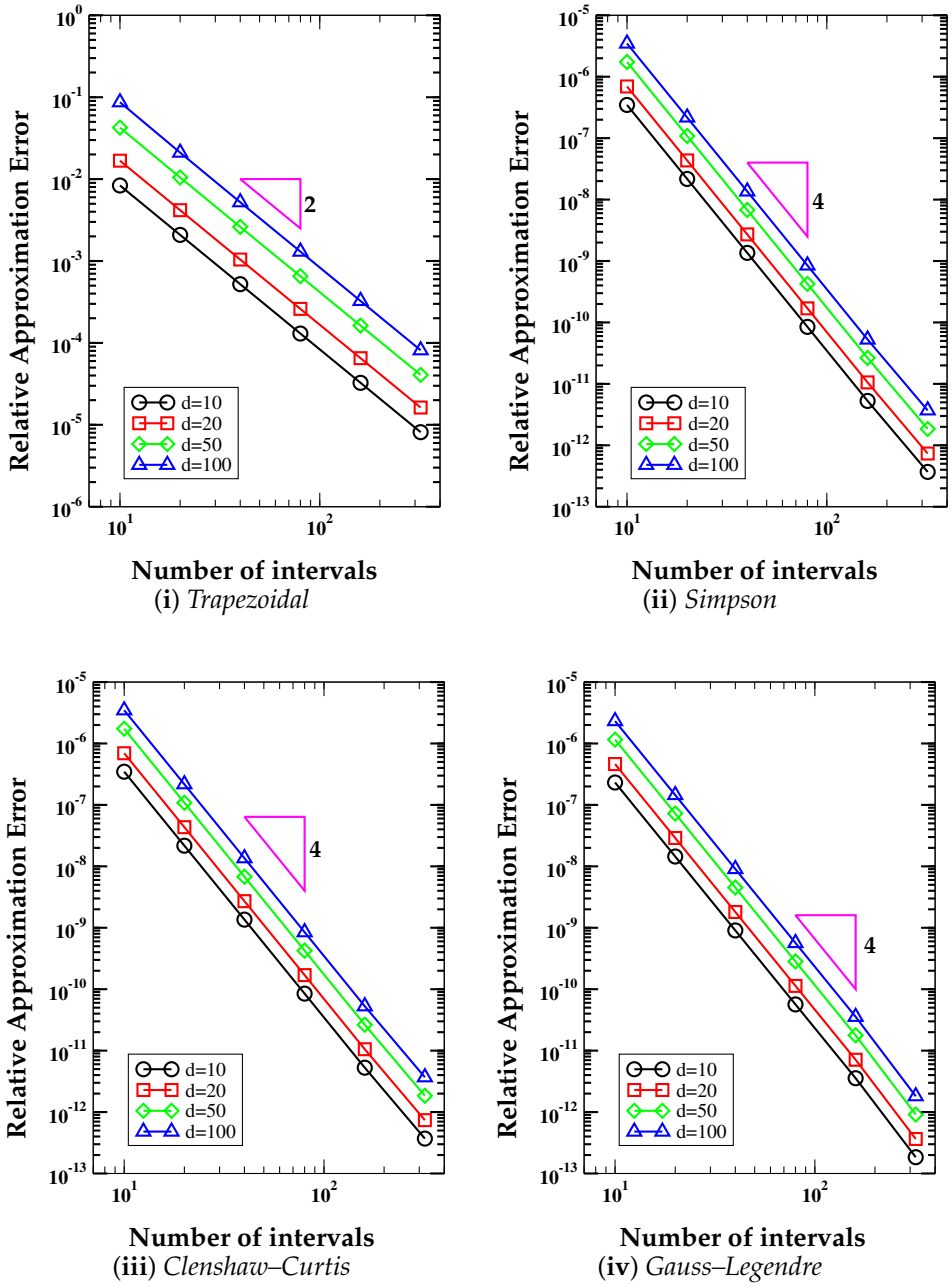


Figure 4. Test Case 3. Relative approximation error curves versus the number of intervals N per direction for the numerical integration of a d-dimensional exponential function (Genz-5), $d = 10, 20, 50, 100$, using the **(i)** trapezoidal formula ($m = 2$); **(ii)** Simpson formula ($m = 4$); **(iii)** Clenshaw–Curtis formula ($m = 4$); **(iv)** Gauss–Legendre formulas ($m = 2$). The expected convergence rate is shown by the number close to the triangular shapes and is proportional to N^{-2} (trapezoidal rule) and N^{-4} (Simpson, Clenshaw–Curtis, and Gauss–Legendre rules).

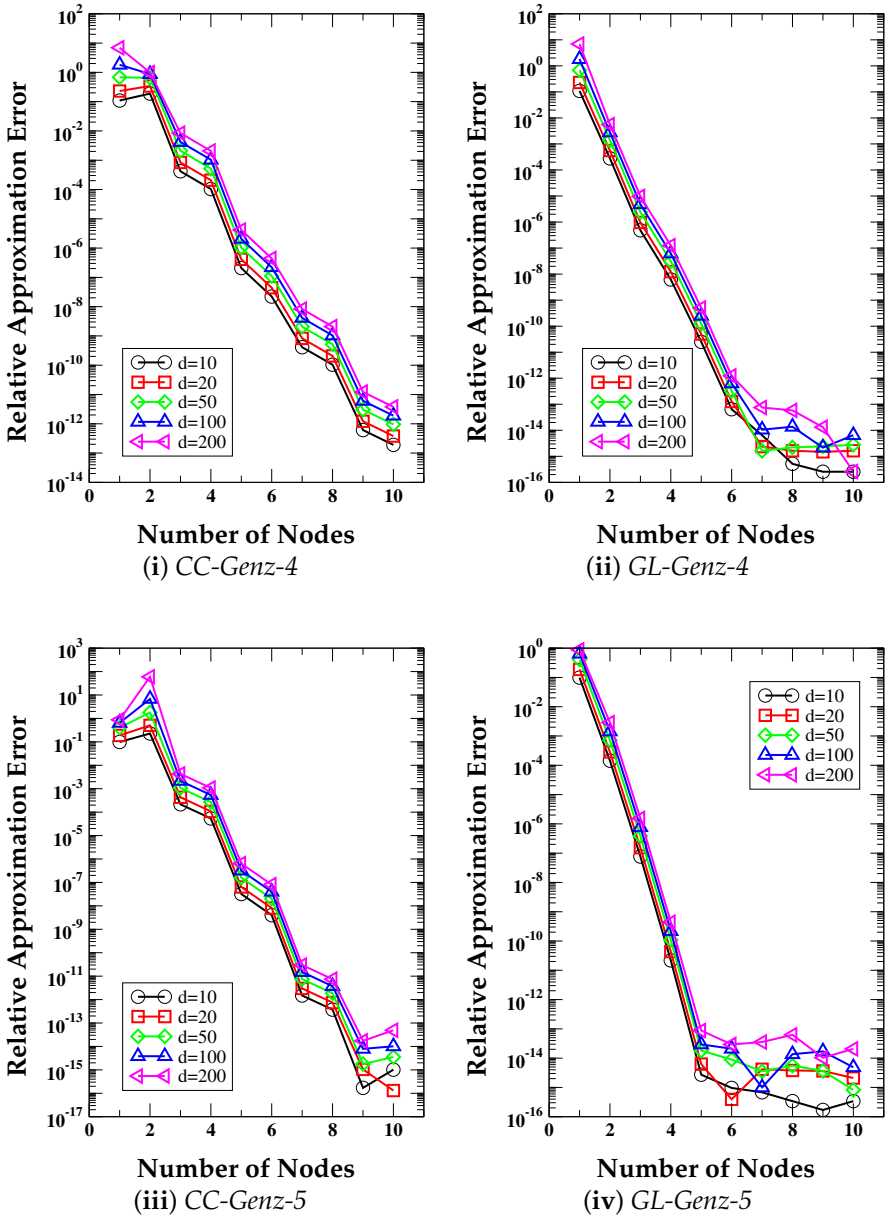


Figure 5. Test Case 4. Relative approximation error curves versus the number of integration nodes m of the univariate quadrature rule for the numerical integration of a d-dimensional Gaussian bell-shaped function (Genz-4, top panels) and the exponential function (Genz-5, bottom panels). We integrate both multidimensional functions on a 2^d grid with $d = 10, 20, 50, 100, 200$. In panels (**i**) and (**iii**), we use the m-points Clenshaw–Curtis (CC) formulas; in panels (**ii**) and (**iv**), we use the m-points Gauss–Legendre (GL) formulas. In all cases, we consider $m = 1, 2, \ldots, m$.

4.4. Test Case 5: Adaptivity for Numerical Integration of Multidimensional, Weakly Singular Functions

In this test case, we show how we can adapt our cubature formulas to the integration of functions with lower regularity. For this test case, we present only the results obtained with Gauss–Legendre quadrature rule. The results obtained with the Clenshaw–Curtis quadrature rule are essentially the same. To this end, we consider the so-called *ANOVA function*, see, for example, [6]:

- *Test Case 5: ANOVA:* $f(x) = \Pi_{\ell=1}^{d} |x_\ell - \overline{x}_\ell|; \quad I(f) = 1.$

This function is only continuous on the integration domain $[0,1]^d$, as all its directional derivatives have a singularity at the point with ℓ-th coordinate equal to $x_\ell = \overline{x}_\ell$. We assume

that $x_\ell \in (0,1)$ for all directional index $\ell = 1, \ldots, d$. We perform the numerical integration versus the number of dimensions, using the Gauss–Legendre formula with 2 nodes, and the equispaced grids with $N = 1, 2, 4$ partitions per direction. When $\overline{x}_\ell = 1/2$, the left panel of Figure 6 shows that the numerical integration with the partition $N = 1$ does not provide a good result. The reason for this behavior is that the singularity of the integrand is inside the (unique) integration cell. However, when we consider the tensor product grid with $N = 2$ and 4 partitions per direction, the cubature rule provides a very accurate result with an error of the order of the machine's precision. In such a case, the multidimensional grid decomposes the computational domain in such a way that the integrand singularity matches with the cell interfaces. Therefore, we are integrating over subdomains where the integrand is actually smooth, and since the Gauss–Legendre is exact on linear polynomials the final error must be zero up to the machine's precision. This same behavior is reflected by the right panel of Figure 6, which refers to the integration of the ANOVA function with $\overline{x}_\ell = 3/4$. In such a case, only the partition with $N = 4$ cells per direction matches the location of the singularity at a cell interface. This is the only situation in this test case that can be expected to provide a numerical value for the integral that is close to the machine precision, as reflected by the numerical results.

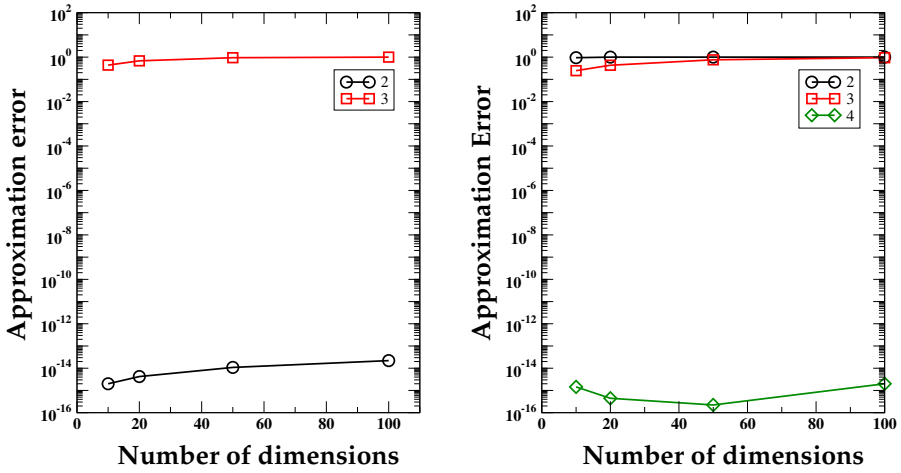


Figure 6. Test Case 5. Approximation error curves versus the number of dimensions d for the numerical integration of a singular function using different partitions of the multidimensional domains. We apply the Gauss–Legendre formula with $m = 4$. The plot on the left shows the errors computed using grids with two and three partitions per direction, and the singularity located at the middle of every direction. The plot on the right shows the same test case with grids with two, three, and four partitions per direction, and the singularity located at the three-quarter mark of every direction. When the singularity is located at an interface between two cells, the integration formula is exact and the error is of the order of the machine's precision, independent of the number of dimensions.

4.5. Test Case 6: Integration of a High-Degree Polynomial and a Function with a Singular High-Order Derivative

We consider the function

- *Test Case 6:* $$f_\mu(x) = \begin{cases} \left(x_1 - \frac{\pi}{4}\right)^\mu + \frac{1}{d}\sum_{\ell=1}^{d} T_\mu(x_\ell) & \text{for } x_1 > \frac{\pi}{4}, \\ \frac{1}{d}\sum_{\ell=1}^{d} T_\mu(x_\ell) & \text{for } x_1 = \frac{\pi}{4}, \\ -\left(x_1 - \frac{\pi}{4}\right)^\mu + \frac{1}{d}\sum_{\ell=1}^{d} T_\mu(x_\ell) & \text{for } x_1 < \frac{\pi}{4}, \end{cases}$$

defined on the integration domain $\Omega = [0,1]^d$, where $T_\mu(x_\ell)$ is the univariate Chebyshev polynomial of the first kind of degree μ along the direction ℓ. The exact value of the integral of the function $f_\mu(x)$ is reported in Table 1. We write the first nine derivatives along the direction x_1 of this function as follows:

$$\text{for } k=1,\dots,\mu: \quad f_\mu^k(x)=\begin{cases} c_{\mu,k}\left(x_1-\frac{\pi}{4}\right)^{\mu-k}+\frac{1}{d}\sum_{\ell=1}^{d}T_\mu^k(x_\ell) & \text{for } x_1>\frac{\pi}{4} \\ \frac{1}{d}\sum_{\ell=1}^{d}T_\mu^k(x_\ell) & \text{for } x_1=\frac{\pi}{4} \\ -c_{\mu,k}\left(x_1-\frac{\pi}{4}\right)^{\mu-k}+\frac{1}{d}\sum_{\ell=1}^{d}T_\mu^k(x_\ell) & \text{for } x_1<\frac{\pi}{4} \end{cases} \tag{22}$$

where $c_{\mu,k}=\Pi_{s=1}^{k}(\mu-s+1)$ and T_μ^k is the k-th derivative of $T_\mu(x_\ell)$. Function $f(x)$ and its first $\mu-1$ derivatives f_μ^k, $1\le k\le\mu-1$, are clearly continuous at the singular point $x_1=\pi/4$ and since the other directions are not affected by the singularity, which is only along the hyperplane $x_1=\pi/4$, all these functions are continuous on Ω. Instead, the μ-th derivative along x_1 is singular at $x_1=\pi/4$, as it reads

$$f_\mu^\mu(x)=\begin{cases} \mu!+\frac{1}{d}\sum_{\ell=1}^{d}T_\mu^\mu(x_\ell) & \text{for } x_1>\frac{\pi}{4}, \\ -\mu!+\frac{1}{d}\sum_{\ell=1}^{d}T_\mu^\mu(x_\ell) & \text{for } x_1<\frac{\pi}{4}. \end{cases} \tag{23}$$

The regularity of f is sufficient to ensure the convergence of the Gauss–Legendre integration formula for $m=1,\dots,\mu$, since according to Theorem 4, we only need a regularity up to f^m. When $m\ge\mu/2$, the Gauss–Legendre quadrature integrates exactly all polynomials of degree up to $\mu-1$, and we expect to see the integration error at the machine precision level. Figure 7 shows the linear–log plot of the error curves versus the number of nodes m. For comparison, we consider a tensor product mesh with two partitions per direction, so that the singularity coincides with a cell interface that is orthogonal to the first spatial direction. In this last situation, the singularity does not affect the accuracy of the numerical integration, as discussed in Test Case 5. The expected behavior is confirmed for $d=10$.

Table 1. Test Case 6: exact value of the integral of the function $f_\mu(x)$ for $\mu=1,\dots,10$.

μ	$I(f_\mu)$
1	0.831452111670637
2	−0.491529937082487
3	−0.404343692999220
4	−0.126345103704993
5	0.205801972908859
6	−0.054903301002036
7	−0.148568167693588
8	−0.028507651909931
9	0.108930994483310
10	−0.016477706416599

4.6. Test Case 7: Efficiency Benchmarks through a Comparison with Monte Carlo Methods

Figure 8 shows the log–log plot of the error curves versus the elapsed time. We compare the performance of the strategy proposed in this paper using the Gauss–Legendre quadrature formula and the performance of two Monte Carlo integration methods (*MC Lattice*, *MC Sobol*, see [35]). These two variants of the Monte Carlo method are provided by the library GAIL [35]. The elapsed time is measured in seconds and the number of dimensions of the integrated function, d, is indicated below each panel. The numerical integration is performed on function *Genz-2* (see [43,44]):

- *Test Case 7: Genz-2:* $f(x)=\Pi_{\ell=1}^{d}\frac{4}{\pi}\left(1+(x_\ell-1)^2\right)^{-1}; \quad I(f)=1,$

on a d-dimensional grid with $d=10,20,50,100,200,500$. These results show that the numerical integration error of the proposed cubature formula can be up to 10 orders of magnitude smaller for comparable elapsed times.

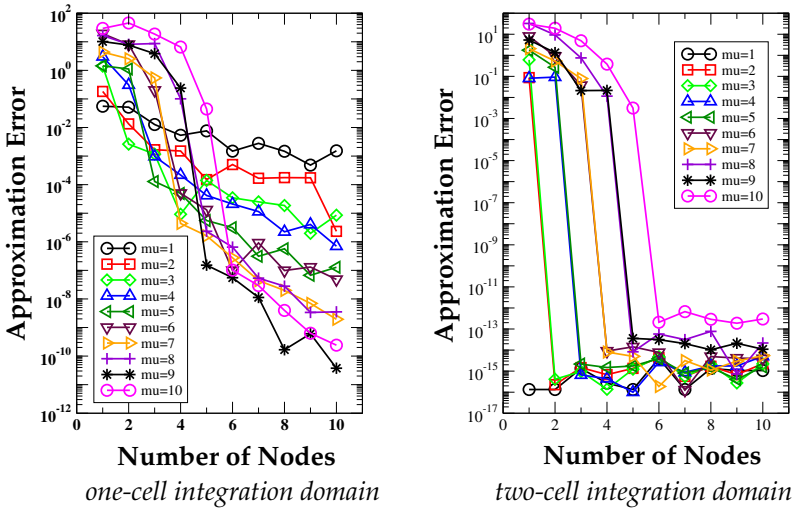


Figure 7. Test Case 6. Approximation error curves versus the number of nodes m for the numerical integration of the combination of a singular function and the Chebyshev degree-10 polynomial of the first kind on a $d = 10$ dimensional domain.

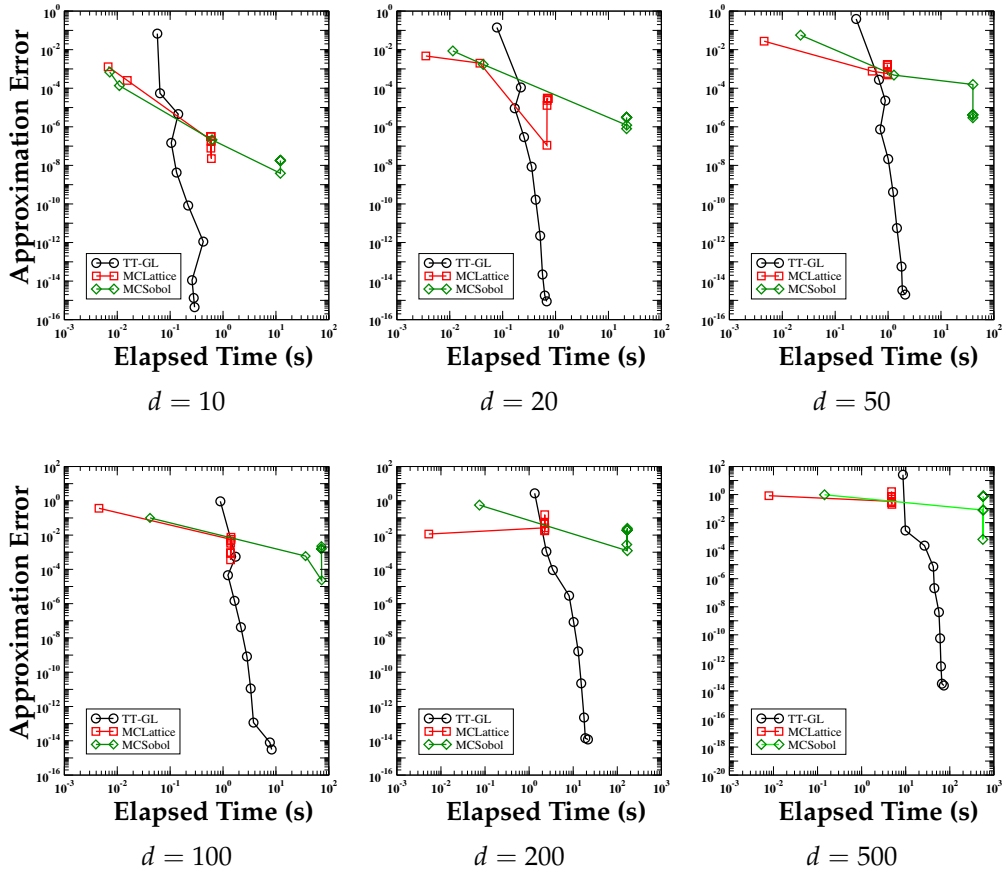


Figure 8. Test Case 7. Approximation error curves versus the elapsed time. We compare the performance of the strategy proposed in this paper and that of two Monte Carlo integration methods (*MC Lattice*, *MC Sobol*). The elapsed time is measured in seconds and the number of dimensions of the integrated function, d, is indicated below each panel. Each panel shows that our integration strategy is more efficient than the Monte Carlo method.

5. Conclusions

In this paper, we discuss an approach to extend the one-dimensional integration rule to a multidimensional grid and presented numerical evidence demonstrating its effectiveness. The straightforward extension of a one-dimensional quadrature to multidimensional grids by the tensor product of the spatial directions is deemed practically impossible for dimensions greater than three or four. The computational burden in terms of storage and floating point operations grows exponentially with the number of dimensions due to the curse of dimensionality. By employing a low-rank tensor-train representation of the integrand functions, however, the tensor product method can be utilized effectively for high-dimensional numerical integration. We assessed the effectiveness of this approach by applying it to the numerical integration of a selected set of multidimensional functions. In particular, the composite integration technique can be competitive with the Monte Carlo method in terms of accuracy and computational costs if the integrand function is sufficiently regular and admits a low-rank tensor train. We obtain such a low-rank representation by applying the TT cross interpolation algorithm, which provides an approximation of the multidimensional tensor collecting the values of the integrand function at the integration nodes. We proved that the accuracy of our composite numerical cubatures is dominated by the "standard" numerical integration error, which depends on the regularity of the integrand function whenever such an error is much bigger than the error introduced by the TT cross approximation algorithm. An open issue is the characterization of the functional interpolation that the TT cross approximate tensor provides. We can assume that this functional interpolation belongs to a subspace of a finite-dimensional approximation space that we can construct, for example, by the tensor product of univariate polynomials, piecewise polynomials interpolations, or similar constructions. If we had a characterization of this subspace, then we could analyze the numerical integration error directly through the function approximation based on the TT cross interpolation algorithm. We will investigate this topic in our future work.

Author Contributions: Conceptualization, B.A., G.M., E.W.S., P.M.D.T. and R.G.V.; Methodology, B.A., G.M., E.W.S., P.M.D.T. and R.G.V.; Validation, B.A., G.M., E.W.S., P.M.D.T. and R.G.V.; Writing—original draft, B.A., G.M., E.W.S., P.M.D.T. and R.G.V.; Writing—review & editing, B.A., G.M., E.W.S., P.M.D.T. and R.G.V. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Laboratory Directory Research and Development (LDRD) grant number 20230067DR and 20210485ER.

Institutional Review Board Statement: This document has been approved for public release as the Los Alamos Technical Report LA-UR-22-29661.

Data Availability Statement: Not applicable.

Acknowledgments: This work was partially supported by the Laboratory Directed Research and Development (LDRD) program of Los Alamos National Laboratory under Grants 20190020DR, 20230067DR and 20210485ER, and in part by LANL Institutional Computing Program. Los Alamos National Laboratory is operated by Triad National Security, LLC, for the National Nuclear Security Administration of U.S. Department of Energy (Contract No. 89233218CNA000001). This article has been authored by an employee of the National Technology & Engineering Solutions of Sandia, LLC under Contract No. DE-NA0003525 with the U.S. Department of Energy (DOE). The employee owns all right, title and interest in and to the article and is solely responsible for its contents. The United States Government retains, and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this article or allows others to do so, for United States Government purposes. The DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan https://www.energy.gov/downloads/doe-public-access-plan.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Clenshaw–Curtis and Gauss–Legendre Quadrature Rules

Clenshaw–Curtis quadratures can be derived through an expansion of the integrand function on Chebyshev polynomials, or, equivalently, through the change of variables $x = \cos\theta$ and, then, applying a discrete cosine transform (DCT) approximation for the cosine series. For the readers' convenience, we report the value of the Clenshaw–Curtis quadrature nodes and weights of order 2, 3, 4, which we used in our numerical experiments, in Table A1.

Table A1. Nodes and weights of the Clenshaw–Curtis quadrature formulas of order 2, 3, and 4 on the interval $[-1, 1]$.

	Clenshaw–Curtiss-2		**Clenshaw–Curtiss-3**		**Clenshaw–Curtiss-4**	
	Nodes	**Weights**	**Nodes**	**Weights**	**Nodes**	**Weights**
1	-1	1	-1	$1/3$	-1	$1/9$
2	1	1	0	$4/3$	$-1/2$	$8/9$
3			1	$1/3$	$1/2$	$8/9$
4					1	$1/9$

The nodes ξ_i, $i = 1, \ldots, n$, of the Gauss–Legendre quadrature of order n are the n roots of $P_n(\xi)$, the univariate Legendre polynomial of degree n defined on the reference interval $[-1, 1]$. The corresponding weight is given by the formula $w_i = 2/\left(\left(1-\xi_i\right)^2\left[P_n'(\xi_i)\right]^2\right)$. For the readers' convenience, we report the Gauss–Legendre quadrature formulas of order 1, 2, and 3, which we used in our numerical experiments, in Table A2.

Table A2. Nodes and weights of the Gauss–Legendre quadrature formulas of order 1, 2, and 3 on the interval $[-1, 1]$.

	Gauss–Legendre-1		**Gauss–Legendre-2**		**Gauss–Legendre-3**	
	Nodes	**Weights**	**Nodes**	**Weights**	**Nodes**	**Weights**
1	0	2	$-1/\sqrt{3}$	1	$-\sqrt{3/5}$	$5/9$
2			$1/\sqrt{3}$	1	0	$8/9$
3					$\sqrt{3/5}$	$5/9$

References

1. Brooks, S.; Gelman, A.; Jones, G.L.; Meng, X.L. *Handbook of Markov Chain Monte Carlo*; Handbooks of Modern Statistical Methods; Chapman & Hall/CRC: Boca Raton, FL, USA, 2011.
2. Stuart, A.M. Inverse problems: A Bayesian perspective. *Acta Numer.* **2010**, *19*, 451–559. [CrossRef]
3. Meyer, H.D.; Manthe, U.; Cederbaum, L.S. The multi-configurational time-dependent Hartree approach. *Chem. Phys. Lett.* **1990**, *165*, 73–78. [CrossRef]
4. Aghanim, N.; Akrami, Y.; Ashdown, M.; Aumont, J.; Baccigalupi, C.; Ballardini, M.; Banday, A.J.; Barreiro, R.B.; Bartolo, N.; Basak, S.; et al. Planck 2018 results. *Astron. Astrophys.* **2020**, *641*, A6. [CrossRef]
5. Barbu, A.; Zhu, S.C. *Monte Carlo Methods*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 1–17.
6. Wang, X.; Fang, K.T. The effective dimension and quasi-Monte Carlo integration. *J. Complex.* **2003**, *19*, 101–124. [CrossRef]
7. Giles, M.B. Multilevel Monte Carlo methods. *Acta Numer.* **2015**, *24*, 259–328. [CrossRef]
8. Gerstner, T.; Griebel, M. Numerical integration using sparse grids. *Numer. Algorithms* **1998**, *18*, 209. [CrossRef]
9. Novak, E.; Ritter, K. High dimensional integration of smooth functions over cubes. *Numer. Math.* **1996**, *75*, 79–97. [CrossRef]
10. Griebel, M.; Schneider, M.; Zenger, C. A combination technique for the solution of sparse grid problems. In *Iterative Methods in Linear Algebra*; de Groen, P., Beauwens, R., Eds.; IMACS, Elsevier: Amsterdam, The Netherlands, 1992; pp. 263–281.
11. Sloan, I.H. Lattice methods for multiple integration. *J. Comput. Appl. Math.* **1985**, *12–13*, 131–143. [CrossRef]
12. Sloan, I.H.; Joe, S. *Lattice Methods for Multiple Integration*; Oxford University Press: Oxford, UK, 1994.
13. Piazzola, C.; Tamellini, L. The Sparse Grids Matlab kit—A Matlab implementation of sparse grids for high-dimensional function approximation and uncertainty quantification. *arXiv* **2022**, arXiv:2203.09314.
14. Jakeman, J.D. PyApprox Software Library. Available online: https://sandialabs.github.io/pyapprox/index.html (accessed on 1 December 2022).

15. Cools, R. Constructing cubature formulae: The science behind the art. *Acta Numer.* **1997**, *6*, 1–54. [CrossRef]
16. Davis, P.J.; Rabinowitz, P.; Rheinbolt, W. *Methods of Numerical Integration*, 2nd ed.; Computer Science and Applied Mathematics; Elsevier Inc.: Amsterdam, The Netherlands; Academic Press: Cambridge, MA, USA, 1984.
17. Brass, H.; Petras, K. *Quadrature Theory*; Mathematical Surveys and Monographs 178; American Mathematical Society: Providence, RI, USA, 2011.
18. Trefethen, L.N. *Approximation Theory and Approximation Practice*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2013. [CrossRef]
19. Hackbusch, W. *Tensor Spaces and Numerical Tensor Calculus*, 1st ed.; Springer Series in Computational Mathematics 42; Springer: Berlin/Heidelberg, Germany, 2012.
20. Hackbusch, W. Numerical tensor calculus. *Acta Numer.* **2014**, *23*, 651–742. [CrossRef]
21. Oseledets, I.V. Tensor-Train decomposition. *SIAM J. Sci. Comput.* **2011**, *33*. [CrossRef]
22. Oseledets, I.; Tyrtyshnikov, E. TT-cross approximation for multidimensional arrays. *Linear Algebra Its Appl.* **2010**, *432*, 70–88. [CrossRef]
23. Khoromskij, B.N. O(dlog n)-quantics approximation of N-d tensors in high-dimensional numerical modeling. *Constr. Approx.* **2011**, *34*, 257–280. [CrossRef]
24. Savostyanov, D.V. Fast revealing of mode ranks of tensor in canonical form. *Numer. Math. Theory Methods Appl.* **2009**, *2*, 439–444. [CrossRef]
25. Oseledets, I.V.; Savostianov, D.V.; Tyrtyshnikov, E.E. Tucker Dimensionality Reduction of Three-Dimensional Arrays in Linear Time. *SIAM J. Matrix Anal. Appl.* **2008**, *30*, 939–956. [CrossRef]
26. Hackbusch, W.; Kühn, S. A new scheme for the tensor representation. *J. Fourier Anal. Appl.* **2009**, *15*, 706–722. [CrossRef]
27. Oseledets, I.V.; Savostyanov, D.V.; Tyrtyshnikov, E.E. Cross approximation in tensor electron density computations. *Numer. Linear Algebra Appl.* **2010**, *17*, 935–952. [CrossRef]
28. Dolgov, S.; Khoromskij, B. Simultaneous state-time approximation of the chemical master equation using tensor product formats. *Numer. Linear Algebra Appl.* **2014**, *22*, 197–219. [CrossRef]
29. Dolgov, S.; Khoromskij, B.; Savostyanov, D. Superfast Fourier transform using QTT approximation. *J. Fourier Anal. Appl.* **2012**, *18*, 915–953. [CrossRef]
30. Zhang, Z.; Yang, X.; Oseledets, I.V.; Karniadakis, G.E.; Daniel, L. Enabling high-dimensional hierarchical uncertainty quantification by ANOVA and tensor-train decomposition. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2015**, *34*, 63–76. [CrossRef]
31. Richter, L.; Sallandt, L.; Nüsken, N. Solving high-dimensional parabolic PDEs using the tensor-train format. In Proceedings of the 38th International Conference on Machine Learning, Virtual, 18–24 July 2021.
32. Dolgov, S.V. A tensor decomposition algorithm for large ODEs with conservation laws. *Comput. Methods Appl. Math.* **2019**, *19*, 23–38. [CrossRef]
33. Dolgov, S.; Savostyanov, D. Parallel cross interpolation for high-precision calculation of high-dimensional integrals. *Comput. Phys. Commun.* **2019**, *246*, 106869. [CrossRef]
34. Vysotsky, L.I.; Smirnov, A.V.; Tyrtyshnikov, E.E. Tensor-train numerical integration of multivariate functions with singularities. *Lobachevskii J. Math.* **2021**, *42*, 1608–1621. [CrossRef]
35. Choi, S.C.T.; Ding, Y.; Hickernell, F.J.; Jiang, L.; Jiménez Rugama, L.A.; Tong, X.; Zhang, Y.; Zhou, X. GAIL: Guaranteed Automatic Integration Library (Versions 1.0–2.2). MATLAB Software (2013–2017). 2021. Available online: http://gailgithub.github.io/GAIL_Dev/ (accessed on 1 June 2021).
36. Imhof, J.P. On the method for numerical integration of Clenshaw and Curtis. *Numer. Math.* **1963**, *5*, 138–141. [CrossRef]
37. Quarteroni, A.; Sacco, R.; Saleri, F. *Numerical Mathematics*, 2nd ed.; Texts in Applied Mathematics; Springer: Berlin/Heidelberg, Germany, 2007; Volume 37.
38. Trefethen, L.N. Is Gauss Quadrature Better than Clenshaw–Curtis? *SIAM Rev.* **2008**, *50*. [CrossRef]
39. Kahaner, D.; Moler, C.; Nash, S. *Numerical Methods and Software*; Prentice Hall: Hoboken, NJ, USA, 1988.
40. *MATLAB*, Version 9.8.0 (R2020a); The MathWorks Inc.: Natick, MA, USA, 2020.
41. Van Rossum, G.; Drake, F.L., Jr. *Python Reference Manual*; Centrum Voor Wiskunde en Informatica: Amsterdam, The Netherlands, 1995.
42. Blackford, L.S.; Demmel, J.; Dongarra, J.; Du, I.; Hammarling, S.; Henry, G.; Heroux, M.; Kaufman, L.; Lumsdaine, A.; Petitet, A.; et al. An updated set of basic linear algebra subprograms (BLAS). *ACM Trans. Math. Softw.* **2002**, *28*, 135–151.
43. Genz, A. Testing multidimensional integration routines. In *Tools, Methods, and Languages for Scientific and Engineering Computation*; Ford, B., Rault, J.C., Eds.; Elsevier: Amsterdam, The Netherlands, 1984; pp. 81–94.
44. Genz, A. A package for testing multiple integration subroutines. In *Merical Integration: Recent Developments, Software and Applications*; Keast, P., Reidel, G.F., Eds.; Springer: Berlin/Heidelberg, Germany, 1987; pp. 337–340.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.