Journal of Computational Physics 400 (2020) 108979

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Generation of nested quadrature rules for generic weight functions via numerical optimization: Application to sparse grids

Vahid Keshavarzzadeh[a,*], Robert M. Kirby[a,b], Akil Narayan[a,c]

[a] *Scientific Computing and Imaging Institute, University of Utah, United States*
[b] *School of Computing, University of Utah, United States*
[c] *Department of Mathematics, University of Utah, United States*

A R T I C L E   I N F O

Article history:
Received 3 August 2018
Received in revised form 30 August 2019
Accepted 17 September 2019
Available online 27 September 2019

Keywords:
Nested quadrature
Optimal quadrature
Sparse grids
Numerical integration
Polynomial approximation

A B S T R A C T

We present a numerical framework for computing nested quadrature rules for various weight functions. The well-known Kronrod method extends the Gauss-Legendre quadrature by adding new optimal nodes to the existing Gauss nodes for integration of higher order polynomials. Our numerical method generalizes the Kronrod rule for any continuous probability density function on real line with finite moments. We develop a bi-level optimization scheme to solve moment-matching conditions for two levels of main and nested rule and use a penalty method to enforce the constraints on the limits of the nodes and weights. We demonstrate our nested quadrature rule for probability measures on finite/infinite and symmetric/asymmetric supports. We generate Gauss-Kronrod-Patterson rules by slightly modifying our algorithm and present results associated with Chebyshev polynomials which are not reported elsewhere. We finally show the application of our nested rules in construction of sparse grids where we validate the accuracy and efficiency of such nested quadrature-based sparse grids on parameterized boundary and initial value problems in multiple dimensions.

1. Introduction

A quadrature formula for integration takes the form

$$\int_{\Gamma} f(x)\omega(x)dx = \sum_{i=1}^{n} w_i f(x_i),$$

where the weight $\omega : \Gamma \to \mathbb{R}$ is a positive measurable function with finite moments. The weights w_i and nodes x_i are selected to maximize the order p for which the above integral is exact for all polynomials f with degree up to p. The well-known optimal rule for integration in one variable is Gauss quadrature which integrates polynomials of degree $p = 2n - 1$ or less with n nodes.

* Corresponding author.
E-mail addresses: vkeshava@sci.utah.edu (V. Keshavarzzadeh), kirby@sci.utah.edu (R.M. Kirby), akil@sci.utah.edu (A. Narayan).

https://doi.org/10.1016/j.jcp.2019.108979
0021-9991/© 2019 Elsevier Inc. All rights reserved.

In many scientific computing applications such as uncertainty quantification (UQ), integrals are evaluated via quadrature rules where each quadrature node typically corresponds to an expensive simulation. When integrals with differing accuracies (i.e., polynomial exactness) are desired for error estimation or extrapolation, it is therefore desirable to use a nested quadrature rule where each formula is a subset of a node set with higher degree of exactness. A generic strategy is to start with a size-n_1 rule and enrich it with $n_2 - n_1 > 0$ nodes where the resulting n_2 nodes with a new set of weights integrates higher order polynomials. In all that follows, we refer to the smaller size-n_1 rule as the "nested" rule and the rule with the larger number n_2 of points as the "main" rule.

Kronrod [1] extended the well-known Gauss-Legendre formulas by adding $n+1$ points to existing n Gauss points in some cases. The resulting nodes integrate with accuracy $p = 3n+1$ for n even, and $p = 3n+2$ for n odd. He showed that this is the best possible extension in terms of the maximum degree of exactness and provided tables for up to $n = 40$ points.

In this paper we propose a systematic optimization algorithm that generates nested quadrature formulas for general continuous univariate distributions. The numerical procedure in this paper utilizes the building blocks of our previous algorithm [2] for finding multidimensional quadrature. However, these two algorithms are different in several mathematical details and their implementations as the original algorithm solves a polynomial system in multiple dimensions whereas the algorithm in this paper solves a bi-level optimization problem which involves both main and nested quadrature rules.

There are four main ingredients in our quadrature optimization both in multiple dimensions [2] and the current paper: penalization, iteration, regularization and initialization. The algorithm in this paper is substantially different from the one in [2] in the steps of iteration and initialization. There is also a major difference between these algorithms in the strategy for defining the optimization problems: the original algorithm fixes the order (degree) of a multivariate polynomial space and searches through the number of points until the tolerance is reached. In contrast, the algorithm in this paper fixes the number of points in both main and nested rules in a preset configuration, and then searches through optimal order until the tolerance is reached. Thus, the algorithm from previous work [2] cannot be directly applied in a simple way to obtain the results from this paper.

A more technical description of the differences between the two algorithms are summarized below:

- Unlike the original algorithm which initializes points randomly in multiple dimensions, in this work we consider a predefined initial configuration for main and nested rules as depicted in Fig. 1. This more effective initialization is possible due to exploiting of geometry in one dimension. We also consider shrinking the range of main rules in the initialization step for unbounded domains as the current algorithm involves high polynomial orders. This issue was not a challenge in the original algorithm as it only involved low to moderate order polynomials. The details of range shrinkage for unbounded domains are discussed in Remark 3.1 and demonstrated in Fig. 11. The details of initialization, especially the preset configuration of main and nested rules, are discussed in Section 3.4.
- To accommodate the nested rule as a subset of main rule i.e. $\boldsymbol{x}_1 \subset \boldsymbol{x}_2$ where $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are the nodes for nested and main rules, respectively, we form the residual and Jacobian differently from the original algorithm. In this algorithm we consider two residual vectors $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$ which take into account the violation in moment-matching conditions for nested and main rules respectively. These two residuals are parametrized independently i.e. $\boldsymbol{R}_1$ is dependent on the nodes and weights associated with the nested rule and $\boldsymbol{R}_2$ is dependent on the nodes of the main rule which contains the nested nodes, and weights of the main rule. To minimize these residuals simultaneously we form the Jacobian as follows: the derivative of $\boldsymbol{R}_1$ with respect to interlacing points (those in the main rule and not the nested rule) and weights of main rule is set to zero; we compute the derivative for the rest of decision variables analytically as discussed in Section 3.2. Similarly for $\boldsymbol{R}_2$ we only set the derivative with respect to the weights of the nested rule zero and compute the rest of derivatives analytically. We note that this innovation in the definition of residuals is one of the novel ingredients that makes this procedure for computing hierarchical rules successful; alternative formulations that we tested resulted in failure of the algorithm to converge.
- Once we have an algorithm which minimizes both $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$ simultaneously we fix the number of points as well as the order of the nested rule and then search for the optimal order as outlined in Algorithm 1. Using this procedure we have been able to find nested quadrature rules in one dimension for several different weight functions including those on bounded and unbounded domains.

Our algorithm is simple and easily implementable. The complete pseudocode is provided in this paper; we plan to distribute the MATLAB implementation of our algorithm in a public repository in the future.

The organization of this paper is as follows. In Section 2 we briefly discuss mathematical setting of Gauss quadrature, Kronrod's extension for nested quadrature and our nested rule. We describe in detail the computational framework for generating the proposed nested quadrature rule in Section 3, and present numerical results in Section 4. Finally, Section 5 discusses the concluding remarks.

2. Univariate quadrature

2.1. Notation

Let $\omega(\boldsymbol{x})$ be a given non-negative weight function (or a probability density function) whose support is $\Gamma \subset \mathbb{R}$ where Γ need not be compact. We assume ω is a measurable function. The space $L^2_\omega(\Gamma)$ is the set of functions f defined by

$$L^2_\omega(\Gamma) = \left\{ f : \Gamma \to \mathbb{R} \;\middle|\; \|f\| < \infty \right\}, \qquad \|f\|^2 = (f, f), \qquad (f, g) = \int_\Gamma f(x) g(x) \omega(x) \mathrm{d}x.$$

We assume that the weight function has finite moments for squared monomials of all orders, i.e.,

$$\int_\Gamma \left(x^\alpha\right)^2 \omega(x) < \infty, \qquad \alpha \in \mathbb{N}_0 := \{0, 1, \ldots\}. \tag{1}$$

Note that these squared polynomials must also have non-vanishing norms since ω is single-signed, measurable, and has non-zero norm; and hence ω has an infinite number of points of support. The assumption above ensures that monomials are linearly independent set of functions in L^2_ω.

Lemma 2.1. *Under the assumption (1), then $\{x^j\}_{j\geq 0}$ is a linearly independent set in L^2_ω.*

Proof. The upper inequality in (1) along with the Cauchy-Schwarz inequality imply that for any constants $c_1, \ldots, c_J$ and degrees $\{\alpha_1, \ldots, \alpha_J\} \subset \mathbb{N}_0$, we have

$$\begin{aligned} \int_\Gamma \left[\sum_{j=1}^J c_J x^{\alpha_j} \right]^2 \omega(x) \mathrm{d}x &= \sum_{j=1}^J c_j^2 \int_\Gamma \left(x^{\alpha_j}\right)^2 \omega(x) \mathrm{d}x + \sum_{\substack{j,k=1 \\ j \neq k}}^N 2 c_j c_k \int_\Gamma x^{\alpha_j} x^{\alpha_k} \omega(x) \mathrm{d}x \\ &\leq \sum_{j=1}^J c_j^2 \|x^{\alpha_j}\|^2 + \sum_{\substack{j,k=1 \\ j \neq k}}^J 2 |c_j| |c_k| \|x^{\alpha_j}\| \|x^{\alpha_k}\| < \infty, \end{aligned}$$

showing that any polynomial with a finite expansion in monomials is in L^2_ω. An iteration of this process shows that any finite linear combination of monomials is in L^2_ω.

We now show that two monomials of different degrees are linearly independent. Consider $\alpha \neq \beta$ with $\alpha, \beta \in \mathbb{N}_0$, and let $c_\alpha, c_\beta \in \mathbb{R}$ be constants that are both not 0. The polynomial

$$c_\alpha x^\alpha + c_\beta x^\beta, \tag{2}$$

is thus a nontrivial polynomial of degree at most $|\alpha + \beta|$, and so it can have at most $|\alpha + \beta| < \infty$ real-valued solutions by the Fundamental Theorem of Algebra. Let γ be the subset of $\mathbb{R}$ where (2) vanishes, which is a finite set in $\mathbb{R}$. Since γ is finite, then $\int_{\Gamma \backslash \gamma} \omega(x) \mathrm{d}x > 0$ since removing a finite number of points from the integral does not affect its value. Then

$$\|c_\alpha x^\alpha + c_\beta x^\beta\|^2 = \int_\Gamma \left(c_\alpha x^\alpha + c_\beta x^\beta\right)^2 \omega(x) \mathrm{d}x = \int_{\Gamma \backslash \gamma} \left(c_\alpha x^\alpha + c_\beta x^\beta\right)^2 \omega(x) \mathrm{d}x > 0,$$

where the inequality holds since the integrand is strictly positive on $\Gamma \backslash \gamma$, and the second equality holds since finite sets have zero measure. This shows that x^α and x^β are linearly independent in L^2_ω. The arguments above can be repeated to show that any finite set of monomials of differing degrees is linearly independent. □

Throughout we use α to denote the degree of a polynomial. Given an integrable function f, we will also use the notation

$$I(f) = \int_\Gamma f(x) \omega(x) \mathrm{d}x. \tag{3}$$

In this paper we seek to construct two sets of n_1 and n_2 points $\left\{x_1^{(q)}\right\}_{q=1}^{n_1} \subset \left\{x_2^{(q)}\right\}_{q=1}^{n_2} \subset \Gamma$ with positive weights $w_1^{(q)}, w_2^{(q)} > 0$ such that

$$I(f_1) = \sum_{q=1}^{n_1} w_1^{(q)} f_1(x_1^{(q)}), \qquad f_1 \in \Pi_{\alpha_1} \tag{4a}$$

$$I(f_2) = \sum_{q=1}^{n_2} w_2^{(q)} f_2(x_2^{(q)}), \qquad f_2 \in \Pi_{\alpha_2}, \tag{4b}$$

where Π_α is the space of polynomials up to degree α:

$$\Pi_\alpha = \text{span}\left\{x^j \mid 0 \le j \le \alpha\right\}. \tag{5}$$

We emphasize that we desire positive weights so that integrating a positive function (such as a probability density function) will always result in a positive value. In (4) we assume $\alpha_1 < \alpha_2$; throughout this paper we use α_1, α_2 to denote the polynomials degrees for f_1 and f_2.

In many applications, the integrand f in (3) exhibits smoothness (e.g., integrable high-order derivatives), which implies high-order convergence when approximating f by a polynomial. Assuming f is smooth, we expect the quadrature rules in (4) applied to f to be good approximations to $I(f)$ if Π_{α_1} is a large enough subspace. Our main goal in this paper is then to make Π_{α_1} and Π_{α_2} as large as possible while keeping n_1, n_2 as small as possible.

In principle, our numerical method applies to general weight functions in multiple dimensions, but will suffer from the standard complications associated with the curse of dimensionality. In this paper, we restrict our attention and numerical examples to computation of univariate nested quadrature rules for a number of standard weight functions.

2.2. Gauss quadrature

It is well known that the ω-Gauss quadrature rule is optimal quadrature rule for univariate integration, in terms of polynomial accuracy. To define this rule, we first prescribe an orthonormal basis for polynomials. Such a basis of orthonormal polynomials can be constructed via a Gram-Schmidt procedure, with elements $p_n(\cdot)$, where $\deg p_n = n$. Orthonormal polynomials are unique up to a multipicative sign, and satisfy the three-term recurrence relation

$$xp_n(x) = \sqrt{b_n}p_{n-1}(x) + a_n p_n(x) + \sqrt{b_{n+1}}p_{n+1}(x), \tag{6}$$

for $n \ge 0$, with $p_{-1} \equiv 0$ and $p_0 \equiv 1/\sqrt{b_0}$ to seed the recurrence. The recurrence coefficients are given by

$$a_n = (xp_n, p_n), \qquad b_n^2 = \frac{(p_n, p_n)}{(p_{n-1}, p_{n-1})},$$

for $n \ge 0$, where $b_0 = (1, 1)$ the integral of ω over D. Explicit formulas for the a_n and b_n coefficients are available for various classical orthogonal polynomial families, such as the Legendre and Hermite polynomials [3]. Gaussian quadrature rules are n-point rules that exactly integrate polynomials in Π_{2n-1} [4,5], and have essentially complete characterizations.

Theorem 2.1 *(Gaussian quadrature). Let $x_1, \ldots, x_n$ be the roots of the nth orthogonal polynomial $p_n(x)$ and let $w_1, \ldots, w_n$ be the solution of the system of equations*

$$\sum_{q=1}^{n} p_j(x^{(q)})w^{(q)} = \begin{cases} \sqrt{b_0}, & \textit{if } j = 0 \\ 0, & \textit{if } j = 1, \ldots, n-1. \end{cases} \tag{7}$$

Then $x^{(q)} \in \Gamma$ and $w^{(q)} > 0$ for $q = 1, 2, \ldots, n$ and

$$\int_\Gamma \omega(x)p(x)dx = \sum_{q=1}^{n} p(x^{(q)})w^{(q)} \tag{8}$$

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

Having knowledge of only a finite number of recurrence coefficients a_n, b_n, one can compute the Gauss quadrature rule via well established algorithmic strategies in [6,7].

2.3. Kronrod nested rule

Two different Gauss quadrature sets with e.g. n and $n+1$ nodes can estimate functions with different degrees e.g. $2n-1$ and $2n+1$. The natural question is: what is the maximum degree that the combination of these two sets i.e. $2n+1$ nodes can integrate. Kronrod showed that for the same amount of labor i.e. $2n+1$ nodes in conjunction with n-node Gaussian rule one can integrate $3n+1$ (n even) and $3n+2$ (n odd) polynomials.

Given the Gaussian quadrature A with n points Kronrod found quadrature B different than A that has the maximum possible accuracy. To construct quadrature B, let $l_n(x)$ denote the degree-n Legendre polynomial, and define $k_{2n+1}(x) = l_n(x)p_{n+1}(x)$ as a polynomial of degree $2n+1$ where p_n is a polynomial of degree n. The polynomial k_{2n+1} is defined such that it is orthogonal to all powers x^i for $i = 0, \dots, n$. Let the roots of k_{2n+1} be $x_1, \dots, x_{2n+1}$. The $(2n+1)$-point interpolatory quadrature rule of the form

$$B(f) = \sum_{i=1}^{2n+1} w_i f(x_i),$$

has an accuracy no less than $2n$. However, the accuracy of this rule is substantially larger than $2n$.

Theorem 2.2 *(Kronrod Rule [1]). Quadrature $B(f)$ has an accuracy of $3n+1$ for even n and of $3n+2$ for odd n.*

Kronrod found these quadrature sets for Legendre polynomials up to 40 points. We show that our numerical method is applicable to various weight functions and can generate high order nested quadrature rules up to a certain precision.

2.4. Nested rule via numerical optimization

The Kronrod rule revolves around the ability to find the root of the polynomial k_{2n+1} and as mentioned it has been proven for certain number of nodes and polynomial degrees and weight function. Similarly to the Kronrod rule we construct a nested rule which centers around the direct moment-matching conditions for the lower and the upper rule. The following result states the essence of our nested numerical quadrature:

Proposition 2.1. *Let $n_1, n_2, \alpha_1, \alpha_2 \in \mathbb{N}$ be given, with $n_1 < n_2$ and $\alpha_1 < \alpha_2$. Suppose that $\left\{x_1^{(q)}\right\}_{q=1}^{n_1} \subset \left\{x_2^{(q)}\right\}_{q=1}^{n_2}$ and $\left\{w_1^{(q)}\right\}_{q=1}^{n_1}$, $\left\{w_2^{(q)}\right\}_{q=1}^{n_2}$ are the solution of two systems of equations*

$$\sum_{q=1}^{n_1} p_j(x_1^{(q)}) w_1^{(q)} = \begin{cases} \sqrt{b_0}, & \text{if } j = 0 \\ 0, & \text{if } j = 1, \dots, \alpha_1 \end{cases} \qquad \sum_{q=1}^{n_2} p_j(x_2^{(q)}) w_2^{(q)} = \begin{cases} \sqrt{b_0}, & \text{if } j = 0 \\ 0, & \text{if } j = 1, \dots, \alpha_2 \end{cases} \tag{9}$$

then

$$\int_{\Gamma} \omega(x)\pi_1(x)dx = \sum_{q=1}^{n_1} \pi_1(x_1^{(q)}) w_1^{(q)}, \qquad \int_{\Gamma} \omega(x)\pi_2(x)dx = \sum_{q=1}^{n_2} \pi_2(x_2^{(q)}) w_2^{(q)} \tag{10}$$

holds for all polynomials $\pi_1 \in \Pi_{\alpha_1}, \pi_2 \in \Pi_{\alpha_2}$.

Proof. The conclusion follows from linearity. With α_1 and α_2 given, the polynomial subspaces Π_{α_1} and Π_{α_2} are uniquely defined. For any $\pi_i \in \Pi_{\alpha_i}$ for $i = 1, 2$, there are coefficients $\{c_{i,j}\}_{j=0}^{\alpha_i}$ for $i = 1, 2$, such that

$$\pi_i(x) = \sum_{j=0}^{\alpha_i} c_{i,j} p_j(x), \qquad i = 1, 2.$$

The left hand sides in Equation (10) can therefore be written as

$$\begin{aligned} \int_{\Gamma} \omega(x)\pi_1(x)dx &= \sqrt{b_0}(\pi_1, p_0) = c_{0,1}\sqrt{b_0}(p_0, p_0) = \sqrt{b_0}c_{1,0}, \\ \int_{\Gamma} \omega(x)\pi_2(x)dx &= \sqrt{b_0}(\pi_2, p_0) = c_{0,2}\sqrt{b_0}(p_0, p_0) = \sqrt{b_0}c_{2,0}. \end{aligned} \tag{11}$$

Similarly the right hand sides of (10) in view of Equation (9) is given by

$$\begin{aligned} \sum_{q=1}^{n_1} \pi_1(x_1^{(q)}) w_1^{(q)} &= \sum_{j=0}^{\alpha_1} c_{1,j} \Big(\sum_{q=1}^{n_1} w^{(q)} p_j(x^{(q)}) \Big) = \sqrt{b_0}c_{1,0}, \\ \sum_{q=1}^{n_2} \pi_2(x_2^{(q)}) w_2^{(q)} &= \sum_{j=0}^{\alpha_2} c_{2,j} \Big(\sum_{q=1}^{n_2} w^{(q)} p_j(x^{(q)}) \Big) = \sqrt{b_0}c_{2,0}. \end{aligned} \tag{12}$$

Combining (11) and (12) proves (10). □

We note that $\int_\Gamma p_j(x)\omega(x)dx = 0$ when $j \neq 0$ due to orthogonality, and hence Equations (9) are simply moment-matching conditions over two different polynomial spaces Π_{α_1} and Π_{α_2}. The existence of quadrature rules satisfying the above conditions does not guarantee the positivity of weights nor it ensures that the nodes lie in Γ. We enforce these conditions in our numerical method, which is explained in the next section. We also provide a general guideline for the choices of n_1, n_2 and α_1, α_2.

We finally note that Proposition 2.1 can be generalized for multi-dimensional polynomials however the relationship between $n_1, n_2, \alpha_1, \alpha_2$ is less obvious, and the computational cost becomes prohibitive. Hence we focus only on computation of univariate nested rules in this paper. Next, we briefly discuss the multivariate construction of quadrature rules based on the Smolyak algorithm which is known to enjoy efficiency gains from using univariate nested rules.

In the next section we provide a computational framework that solves a constrained version of (9) via optimization.

3. Numerical method

We aim to compute nodes $\boldsymbol{x_1} = \left\{x_1^{(1)}, \dots, x_1^{(n_1)}\right\} \subset \boldsymbol{x_2} = \left\{x_2^{(1)}, \dots, x_2^{(n_2)}\right\} \in \Gamma$ and positive weights $\boldsymbol{w_1} = \left\{w_1^{(1)}, \dots, w_1^{(n_1)}\right\}$, $\boldsymbol{w_2} = \left\{w_2^{(1)}, \dots, w_2^{(n_2)}\right\} \in (0, \infty)$ that satisfies (9) up to a tolerance of a prescribed $\epsilon > 0$. We directly formulate (9) as

$$
\begin{aligned}
&\boldsymbol{R}_1(\boldsymbol{d}_1) = \boldsymbol{V}_{\alpha_1}(\boldsymbol{x}_1)\boldsymbol{w}_1 - \sqrt{b_0}\boldsymbol{e}_{1,\alpha_1+1} = \boldsymbol{0}, \\
&\boldsymbol{R}_2(\boldsymbol{d}_2) = \boldsymbol{V}_{\alpha_2}(\boldsymbol{x}_2)\boldsymbol{w}_2 - \sqrt{b_0}\boldsymbol{e}_{1,\alpha_2+1} = \boldsymbol{0}, \\
&\boldsymbol{x}_1 \subset \boldsymbol{x}_2 \in \Gamma, \\
&\boldsymbol{w}_1, \boldsymbol{w}_2 > \boldsymbol{0},
\end{aligned}
\tag{13}
$$

where $\boldsymbol{V}$ denotes the Vandermonde matrix, e.g. $\boldsymbol{V}_{\alpha_1}$ is the Vandermonde matrix that consists of polynomial evaluations up to degree α_1, $\boldsymbol{e}_{1,\alpha_i+1}$ is the first unit vector with size $\alpha_i + 1$ and $\boldsymbol{d}_1 = (\boldsymbol{x}_1, \boldsymbol{w}_1), \boldsymbol{d}_2 = (\boldsymbol{x}_2, \boldsymbol{w}_2)$ are decision variables. Instead of solving this constrained root finding problem, we introduce a closely related constrained minimization problem on decision variables $\boldsymbol{d} = (\boldsymbol{x}_2, \boldsymbol{w})$:

$$
\begin{aligned}
\min_{\boldsymbol{x}_2, \boldsymbol{w}} \quad & ||\boldsymbol{R}||_2 \\
\text{subject to} \quad & \boldsymbol{x}_2 \in \Gamma, \\
& \boldsymbol{w} > \boldsymbol{0}.
\end{aligned}
\tag{14}
$$

Above we have eliminated the decision variable $\boldsymbol{x}_1$, since $\boldsymbol{x}_1 \subset \boldsymbol{x}_2$. We have also introduced the vectors $\boldsymbol{R}$ and $\boldsymbol{w}$, defined as

$$
\boldsymbol{R} = \begin{bmatrix} \boldsymbol{R_1} \\ \boldsymbol{R_2} \end{bmatrix}, \quad \boldsymbol{w} = \begin{bmatrix} \boldsymbol{w_1} \\ \boldsymbol{w_2} \end{bmatrix}.
\tag{15}
$$

The total number of decision variables above is $n_1 + 2n_2$. Ideally we need the solution to (13), which also solves (14), but the reverse is not necessarily true. In practice we fix the polynomial degree α_2, solve (14), and when the solution exhibits nonzero values of $\|\boldsymbol{R}\|$, we decrease α_2 and repeat. Using this strategy, we empirically find that we can satisfy $\|\boldsymbol{R}\| \leq \epsilon$ in all situations we have tried i.e. we have been able to re-generate available Kronrod rules with very small tolerances, and we have been able to find new nested quadrature rules.

Our approach therefore effectively solves (13) via repeated applications of (14). Our numerical approach to solve (14) is a modification of the algorithm in [2]. We describe in brief the ingredients of this algorithm, and in more detail the portions that are specific to our construction. The overall algorithm has four major steps, each of which are described in the subsequent sections:

1. Section 3.1 Penalization: transforming constrained root finding into unconstrained minimization problem by augmenting the objective with penalty functions. This step is nearly identical to the procedure in [2].
2. Section 3.2 Iteration: Gauss-Newton algorithm for unconstrained minimization.
3. Section 3.3 Regularization: numerical regularization to address ill-conditioned Gauss-Newton update steps. This step is nearly identical to the procedure in [2].
4. Section 3.4 Initialization: specification of an initial guess.

Steps 2 and 4 above differ substantially from the approach in [2]; in addition the way we formulate the optimization problem (as shown in Algorithm 1) differs from the previous approach.

3.1. Penalty method

We use penalty methods to solve the constrained optimization problem (14) by adding a high cost for violated constraints to the objective function. We subsequently solve an unconstrained minimization problem on the augmented objective.

We choose a popular penalty function, the non-negative and smooth quadratic function. Taking as an example $\Gamma = [-1, 1]$, the constraints and corresponding penalties P_j, $j = 1, \dots, 2n_2 + n_1$ as a function of the $2n_2 + n_1$ decision variables $\boldsymbol{d} = (\boldsymbol{x_2}, \boldsymbol{w})$ can be expressed as

$$\begin{aligned} -1 \le x_2^{(j)} \le 1 &\implies P_j(\boldsymbol{d}) = \left(\max[0, x_j - 1, -1 - x_j]\right)^2, j = 1, \dots, n_2 \\ w_2^{(j)} \ge 0 &\implies P_{n_2+j}(\boldsymbol{d}) = \left(\max[0, -w_j]\right)^2, j = 1, \dots, n_2 \\ w_1^{(j)} \ge 0 &\implies P_{2n_2+j}(\boldsymbol{d}) = \left(\max[0, -w_j]\right)^2, j = 1, \dots, n_1. \end{aligned}$$

The total penalty in this case is then expressed as

$$P^2(\boldsymbol{d}) = \sum_{j=1}^{2n_2+n_1} P_j^2(\boldsymbol{d}).$$

The penalty approach solves the constrained problem (14) by using a sequence of unconstrained problems indexed by $k \in \mathbb{N}$ with objective functions

$$g(c_k, \boldsymbol{d}) = \left\|\widetilde{\boldsymbol{R}}_k\right\|_2^2 = \|\boldsymbol{R}\|_2^2 + c_k^2 P^2(\boldsymbol{d}), \tag{16}$$

where

$$\widetilde{\boldsymbol{R}}_k = \begin{bmatrix} \boldsymbol{R} \\ c_k P_1 \\ c_k P_2 \\ \vdots \\ c_k P_{2n_2+n_1} \end{bmatrix}.$$

The positive constants c_k are monotonically increasing with k, i.e., $c_{k+1} > c_k$. Each unconstrained optimization yields an updated solution point $\boldsymbol{d}^k$, and as $c_k \to \infty$ the solution point of the unconstrained problem will converge to the solution of constrained problem.

We now formulate an unconstrained minimization problem with sequence of increasing c_k associated with the objectives g in (16) for the decision variables $\boldsymbol{d} = (\boldsymbol{x_2}, \boldsymbol{w})$,

$$\min_{\boldsymbol{d}} g(c_k, \boldsymbol{d}) \tag{17}$$

which provides an approximation to the solution of the constrained root-finding problem (13). We specify the constants c_k as follows: if $\boldsymbol{d}$ is the current iterate for the decision variables, we use the formula

$$c_k = \max\left\{A, \frac{1}{||\boldsymbol{R}(\boldsymbol{d})||_2}\right\},$$

where A is a tunable parameter that is meant to be large and set to $A = 10^3$ in our numerical examples. We also note that we never have $c_k = \infty$ so that our iterations cannot exactly constrain the computed solution to lie in the feasible set. In practice we reformulate constraints to have non-zero penalty within a small radius inside the feasible set. For example, instead of enforcing $w_j > 0$, we enforce $w_j > 10^{-6}$.

3.2. The Gauss-Newton minimization

Now that the constrained problem (14) is transformed to a sequence of unconstrained problems (17), we can use standard unconstrained optimization tools such as gradient descent or Newton's method.

The gradient-based approaches require the Jacobian of the objective function with respect to the decision variables. We define

$$\boldsymbol{J}(\boldsymbol{d}) = \begin{bmatrix} \boldsymbol{J}_1 \\ \boldsymbol{J}_2 \end{bmatrix} = \begin{bmatrix} \partial \boldsymbol{R}_1 / \partial \boldsymbol{d} \\ \partial \boldsymbol{R}_2 / \partial \boldsymbol{d} \end{bmatrix} \in \mathbb{R}^{(\alpha_1+\alpha_2+2)\times(2n_2+n_1)}, \qquad \widetilde{\boldsymbol{J}}_k = \frac{\partial \widetilde{\boldsymbol{R}}_k}{\partial \boldsymbol{d}} = \begin{bmatrix} \boldsymbol{J} \\ c_k \partial P_1 / \partial \boldsymbol{d} \\ c_k \partial P_2 / \partial \boldsymbol{d} \\ \vdots \\ c_k \partial P_{2n_2+n_1} / \partial \boldsymbol{d} \end{bmatrix}, \tag{18}$$

where $\frac{\partial P_j}{\partial \boldsymbol{d}} \in \mathbb{R}^{1\times(2n_2+n_1)}$ is the Jacobian of P_j with respect to the decision variables. We note that the Lipschitz continuity as well as easy evaluation of Jacobians are ensured using the quadratic penalty function. For instance, the first part of $\boldsymbol{J}$ has entries

$$(J_1)_{m,i} = \frac{\partial p_m\left(x_1^{(i)}\right)}{\partial x_1^{(i)}} w_i, \qquad (J_1)_{m,n_1+i} = p_m({x_1}^{(i)}), \tag{19}$$

for $m = 0, \ldots, \alpha_1$, $i = 1, \ldots, n_1$ where we define the polynomial sequence $\{p_m\}_{m\geq 0}$ as in Section 2.4. The formula is similar for entries of $\boldsymbol{J}_2$. With respect to forming the Jacobian, it should be noted that $\boldsymbol{R}_1$ is dependent on $\boldsymbol{x}_1 \subset \boldsymbol{x}_2$ and $\boldsymbol{w}_1$, and its derivative with respect to the rest of decision variables i.e. $\boldsymbol{x}_2 \backslash \boldsymbol{x}_1$, and $\boldsymbol{w}_2$ is set to zero. Similarly the derivative of $\boldsymbol{R}_2$ with respect to $\boldsymbol{w}_1$ is set to zero. Fig. 1 shows an example of decision variables and their initial configuration.

Computing entries of the Jacobian matrix $\boldsymbol{J}$ is straightforward since we only need to compute derivatives of univariate polynomials. A manipulation of the three-term recurrence relation (6) yields the recurrence

$$\sqrt{b_{m+1}}\, p'_{m+1}(x) = (x - a_m)\, p'_m(x) - \sqrt{b_m}\, p'_{m-1}(x) + p_m(x),$$

which is used to evaluate the partial derivatives in $\boldsymbol{J}$.

We use the same index iterations k as that defining the sequence of unconstrained problems (17); Thus, our choice of c_k changes at each iteration. A simple gradient descent is based on the update with the form

$$\boldsymbol{d}^{k+1} = \boldsymbol{d}^k - \alpha \frac{\partial \|\widetilde{\boldsymbol{R}}_k\|_2}{\partial \boldsymbol{d}}, \qquad \frac{\partial \|\widetilde{\boldsymbol{R}}_k\|_2}{\partial \boldsymbol{d}} = \frac{\widetilde{\boldsymbol{J}}_k^T \widetilde{\boldsymbol{R}}_k}{\|\widetilde{\boldsymbol{R}}_k\|_2},$$

with α a customizable step length that is frequently optimized via, e.g., a line-search algorithm. In contrast, a variant of Newton's method applied to rectangular systems is the Gauss-Newton method [4], with update iteration

$$\boldsymbol{d}^{k+1} = \boldsymbol{d}^k - \Delta \boldsymbol{d}, \qquad \Delta \boldsymbol{d} = \left(\widetilde{\boldsymbol{J}}_k^T \widetilde{\boldsymbol{J}}_k\right)^{-1} \widetilde{\boldsymbol{J}}_k^T \widetilde{\boldsymbol{R}}_k, \tag{20}$$

where both $\widetilde{\boldsymbol{J}}_k$ and $\widetilde{\boldsymbol{R}}_k$ are evaluated at $\boldsymbol{d}^k$. The iteration above reduces to the standard Newton's method when the system is square, i.e., $\alpha_1 + \alpha_2 + 2 = 2n_2 + n_1$. It is well known that Newton's method converges quadratically to a local solution for a sufficiently close initial guess $\boldsymbol{d}^0$ versus the gradient descent which has linear convergence [8]. We use Gauss-Newton iterations that we find robust for our numerical algorithm.

Starting with an initial guess $\boldsymbol{d}^0$, we repeatedly apply the Gauss-Newton iteration (20) until a stopping criterion is met. We terminate our iterations when the residual norm falls below a user-defined threshold ϵ, i.e., when $||\widetilde{\boldsymbol{R}}||_2 < \epsilon$.

We also define another useful quantity to monitor during the iteration process which is the magnitude of the Newton decrement. This measure often reflects quantitative proximity to the optimal point [9]. Based on its definition, the Newton decrement is the norm of the Newton step in the quadratic norm defined by the Hessian. I.e., the Newton decrement norm for a function $f(\boldsymbol{x})$, is $||\Delta \boldsymbol{d}||_{\nabla^2 f(\boldsymbol{x})} = (\Delta \boldsymbol{d}^T \nabla^2 f(\boldsymbol{x}) \Delta \boldsymbol{d})^{1/2}$, where $\nabla^2 f$ is the Hessian of f. In our minimization procedure with non-squared systems we define

$$\eta = \left(\Delta \boldsymbol{d}^T (\widetilde{\boldsymbol{J}}_k^T \widetilde{\boldsymbol{R}}_k)\right)^{1/2}, \tag{21}$$

as a surrogate for a Hessian-based Newton decrement, which decreases as $\boldsymbol{d} \to \boldsymbol{d}^*$.

Finally we note that, for a given nested quadrature rule with $(n_1, n_2, \alpha_1, \alpha_2)$ we cannot guarantee that a solution to (13) exists. In this case our Gauss-Newton iterations will exhibit residual norms stagnating at some positive value while the Newton decrement is almost zero. When this occurs, we either re-initialize our decision variables or decrease the magnitude of the higher order α_2 and continue the optimization procedure. By gradually decreasing α_2, we find a successful combination of $(n_1, n_2, \alpha_1, \alpha_2)$ that meet the residual tolerance criterion.

3.3. Tikhonov regularization

The evaluation of the Newton update (20) is a critical part of our scheme. In our rectangular system, the update is the least-squares solution $\Delta \boldsymbol{d}$ to the linear system

$$\widetilde{\boldsymbol{J}} \Delta \boldsymbol{d} = \widetilde{\boldsymbol{R}},$$

where $\widetilde{\boldsymbol{J}} = \widetilde{\boldsymbol{J}}_k\left(\boldsymbol{d}^k\right)$, and $\widetilde{\boldsymbol{R}} = \widetilde{\boldsymbol{R}}_k\left(\boldsymbol{d}^k\right)$ and in this section we omit explicit notational dependence on the iteration index k. Based on our numerical experience, the Jacobian $\widetilde{\boldsymbol{J}}$ can be ill-conditioned. Therefore, to effectively solve the above least-squares problem we consider a generic regularization of the equality:

$$\underset{\Delta \boldsymbol{d}}{\text{minimize}} \quad ||\widetilde{\boldsymbol{J}} \Delta \boldsymbol{d} - \widetilde{\boldsymbol{R}}||_p \quad \text{subject to} \quad ||\Delta \boldsymbol{d}||_q < \tau, \tag{22}$$

where p, q, and τ are free parameters. The Pareto curve that characterizes the trade off between the objective norm and solution norm is shown to be convex in [10,11] for generic norms $1 \leq (p, q) < \infty$. In this paper we utilize a 2-norm regularization i.e. $p = q = 2$ which can be solved via an approach in [12]; however, this procedure provides no clear guideline on choosing τ. Thus we adopt an alternative approach based on Tikhonov regularization. This approach is a penalized version of the $p = q = 2$ optimization (22):

$$\Delta \boldsymbol{d}_\lambda = \operatorname{argmin}\left\{ ||\widetilde{\boldsymbol{J}}\Delta \boldsymbol{d} - \widetilde{\boldsymbol{R}}||_2^2 + \lambda ||\Delta \boldsymbol{d}||_2^2 \right\}, \tag{23}$$

where λ is a regularization parameter that can be selected by the user. However, this parameter can significantly impact the quality of the solution with respect to the original least-squares problem. Assuming that a value for λ is prescribed, the solution to (23) can be obtained via the singular value decomposition (SVD) of $\widetilde{\boldsymbol{J}}$. The SVD of matrix $\widetilde{\boldsymbol{J}}_{\mathrm{N}\times\mathrm{M}}$ (for N < M) is given by

$$\widetilde{\boldsymbol{J}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T = \sum_{i=1}^{\mathrm{N}} \boldsymbol{u}_i \sigma_i \boldsymbol{v}_i^T, \tag{24}$$

where σ_i are singular values (in decreasing order), and $\boldsymbol{u}_i$ and $\boldsymbol{v}_k$ are the corresponding left- and right-singular vectors, respectively and U, Σ and V are the aforementioned quantities in the matrix form. It is easy to show that the regularized solution in the matrix form is

$$\Delta \boldsymbol{d}_\lambda = (\widetilde{\boldsymbol{J}}^T\widetilde{\boldsymbol{J}} + \lambda^2 \boldsymbol{I})^{-1}\widetilde{\boldsymbol{J}}^T\widetilde{\boldsymbol{R}}, \tag{25}$$

where $\boldsymbol{I}$ is the identity matrix. Using the SVD of $\widetilde{\boldsymbol{J}}$ the regularized solution is written as

$$\begin{aligned} \Delta \boldsymbol{d}_\lambda &= (\boldsymbol{V}\boldsymbol{\Sigma}^2\boldsymbol{V}^T + \lambda^2 \boldsymbol{V}\boldsymbol{V}^T)^{-1}\boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^T\widetilde{\boldsymbol{R}} \\ &= \boldsymbol{V}(\boldsymbol{\Sigma}^2 + \lambda^2\boldsymbol{I})^{-1}\boldsymbol{V}\boldsymbol{V}^T\boldsymbol{\Sigma}\boldsymbol{U}^T\widetilde{\boldsymbol{R}} \\ &= \boldsymbol{V}(\boldsymbol{\Sigma}^2 + \lambda^2\boldsymbol{I})^{-1}\boldsymbol{\Sigma}\boldsymbol{U}^T\widetilde{\boldsymbol{R}} \end{aligned} \tag{26}$$

which leads to the following expression when the singular values and vectors are inserted:

$$\Delta \boldsymbol{d}_\lambda = \sum_{i=1}^{\mathrm{N}} \rho_i \frac{\boldsymbol{u}_i^T\widetilde{\boldsymbol{R}}}{\sigma_i}\boldsymbol{v}_i. \tag{27}$$

In the above equation ρ_i are Tikhonov filter factors denoted by

$$\rho_i = \frac{\sigma_i^2}{\sigma_i^2 + \lambda^2} \simeq \begin{cases} 1 & \sigma_i \gg \lambda, \\ \sigma_i^2/\lambda^2 & \sigma_i \ll \lambda. \end{cases} \tag{28}$$

Tikhonov regularization filters singular values that are below the threshold λ. Therefore a suitable λ is bounded by the extremal singular values of $\widetilde{\boldsymbol{J}}$. One approach to select the regularization parameter is to analyze the "L-curve" of singular values [13,14] and choose λ that corresponds to the corner of L-curve that has the maximum curvature. Approximation of the curvature with respect to singular value index can be used to find the index with maximum curvature, and the singular value corresponding to this index prescribes λ.

In practice, we evaluate the curvature of the singular value spectrum via finite differences on $\log(\sigma_i)$ (where the singular values are directly computed) and choose the singular value that corresponds to the first spike in the spectrum. The regularization parameter can be updated after several, e.g., $30 - 50$, Gauss-Newton iterations. In our small size problems, i.e. univariate problems a small number of fixed appropriate λ throughout the Gauss-Newton scheme also yields the desirable solutions.

In our experiments we also find that adding a regularization parameter to all singular values and computing the regularized Newton step as $\Delta \boldsymbol{d}_\lambda = \sum_{i=1}^{\mathrm{N}}[(\boldsymbol{u}_i^T\widetilde{\boldsymbol{R}})/(\sigma_i + \lambda)]\boldsymbol{v}_i$ (instead of using Equations (27) and (28)) enhances the convergence when $\boldsymbol{d}$ is close to the root i.e. $||\widetilde{\boldsymbol{R}}||$ is sufficiently small. We demonstrate this point via a numerical study in Section 4.1.1.

3.4. Initialization

The first step of the algorithm requires an initial guess $\boldsymbol{d}^0$ for nodes and weights. Mimicking the Kronrod approach, we choose n_1 for the number of nested nodes and $n_2 = 2n_1 + 1$ for the number of main nodes. We also fix the order for the nested rule based on Gauss quadrature (similarly to Kronrod rule) i.e. $\alpha_1 = 2n_1 - 1$. The order for the main rule α_2 is varied until a desired residual norm is achieved. We initialize the main and nested nodes in an interlacing order as depicted in the Fig. 1.

We consider ω a probability density in all cases (i.e., $b_0 = 1$) so that the weights for main and nested rules are normalized to yield $\sum_i w^{(i)} = 1$. We initialize nodes based on the Gauss quadrature nodes for the main rule. Once the main rule

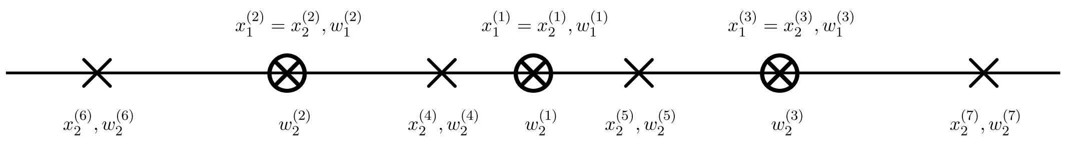

Fig. 1. Configuration of initial nested and main nodes for $n_1 = 3,\ n_2 = 2n_1 + 1 = 7$. The nested nodes $\boldsymbol{x}_1$ are shown with circumscribed circles ($\circ$), whereas the main nodes $\boldsymbol{x}_2$ are marked with $\times$. Labels with subscript $\cdot_1$ indicate decision variables whose variation contributes to the Jacobian of $\boldsymbol{R}_1$, and labels with subscript $\cdot_2$ indicate decision variables whose variation contributes to the Jacobian of $\boldsymbol{R}_2$. In this particular case the first residual $\boldsymbol{R}_1$ is dependent on $\left\{x_2^{(i)}, w_1^{(i)}\right\}_{i=1}^{3}$ and the second residual $\boldsymbol{R}_2$ is dependent on $\left\{x_2^{(i)}\right\}_{i=1}^{7}$ and $\left\{w_2^{(i)}\right\}_{i=1}^{7}$. Note that $x_1^{(i)} = x_2^{(i)},\ i = 1, 2, 3$.

is initialized the nested nodes are readily available as shown in Fig. 1. The weights for main and nested rules are simply considered as uniform values i.e. $\boldsymbol{w}_1 = 1/n_1, \boldsymbol{w}_2 = 1/n_2$.

Remark 3.1. For unbounded weights e.g. Gaussian weight function the optimal main rule with n_2 points has an order much smaller than the optimal Gauss quadrature order with n_2 points. That means the nodes are located in a much shorter range compared to original Gauss quadrature. To generate an initial guess for unbounded domains we use the ratio between the maximum value of the Gaussian quadrature with α_2 degree (corresponding to $\lfloor(\alpha_2 + 1)/2\rfloor$ nodes where $\lfloor . \rfloor$ is the floor function) i.e. $\max\{\boldsymbol{x}_2\}$ and the maximum value of the Gaussian quadrature with n_2 nodes. We use this ratio to shrink the range of Gaussian quadrature with n_2 nodes. We illustrate this shrinkage via a numerical example in Section 4.1.4.

Algorithm 1 summarizes our numerical method for nested quadrature, including the steps in Sections 3.1–3.4.

Algorithm 1 Nested Quadrature Generation.

```
1: Initialize nodes and weights, $n_1$ for the nested nodes and $n_2 = 2n_1 + 1$ for the main nodes associated with the given orthogonal polynomial system
   cf. Section 3.4.
2: Specify the residual tolerance e.g. $\epsilon = 10^{-12}$
3: Set $\alpha_2^* = 0$
4: while $||\widetilde{\boldsymbol{R}}|| > \epsilon$ do
5:     Compute $\widetilde{\boldsymbol{R}}$ and $\widetilde{\boldsymbol{J}}$ using (16), (13), (18), and (19).
6:     Determine the Tikhonov parameter $\lambda$ from the SVD of $\widetilde{\boldsymbol{J}}$
7:     Compute the Newton step $\Delta\boldsymbol{d}$ from (27)
8:     Update the decision variables $\boldsymbol{d}^{k+1} = \boldsymbol{d}^k - \Delta\boldsymbol{d}$.
9:     Compute the residual norm $||\widetilde{\boldsymbol{R}}||_2$ and Newton decrement $\eta$ from (21).
10:    if $\eta < \epsilon$ and $||\widetilde{\boldsymbol{R}}||_2 \gg \epsilon$ then
11:        $\alpha_2 \leftarrow \alpha_2 - 1$ and go to line 4.
12:    end if
13: end while
14: if $\alpha_2 = \alpha_2^*$ then
15:     Return
16: else
17:     $\alpha_2^* \leftarrow \alpha_2$, $\alpha_2 \leftarrow \alpha_2 + 1$ and go to line 4
18: end if
```

4. Numerical examples

4.1. Univariate examples

In this section we investigate the effectiveness of the nested quadrature rules on evaluating univariate integrals. Some of these experiments serve as validation where we verify existing nested rules. The remaining experiments focus on highlighting the computation of new rules using the general methodology we have introduced.

In addition to quadrature corresponding to Kronrod rules, we have also generated existing and new quadrature for Gauss-Kronrod-Patterson rules which we briefly describe at this juncture.

Patterson [15] extended Kronrod rule by generating nested sequences of univariate quadrature rules. In other words, a Kronrod-Patterson rule with accuracy level i adds a number of points to the preceding level set $\mathbb{X}_i$ sequentially and updates the weights accordingly. As such this procedure results in $\mathbb{X}_i \subset \mathbb{X}_{i+1} \subset \ldots$. These nodes have been specifically obtained for Legendre polynomials [15] and later in a similar fashion for Hermite polynomials [16]. To investigate these rules we slightly amend our algorithm and generate some of these rules in addition to sequential nested rules for Chebyshev polynomials, which to our knowledge are new.

To generate Kronrod-Patterson sequential nested rules, we use an initial guess with fixed n_1 number of points and optimize $n_1 + 1$ additional points. The sequence starts with $n_1 = 1$ and $x_1^{(1)} = 0$ which integrates $\alpha_1 = 1$ order. The next step is to add $1 + 1 = 2$ points and optimize their locations and weights (while the initial 1 point is fixed) to achieve an optimal order e.g. $\alpha_2 = 5$ in the case of Legendre and Chebyshev. At this point the new rule with $n_2 = 3$ and $\alpha_2 = 5$ is considered

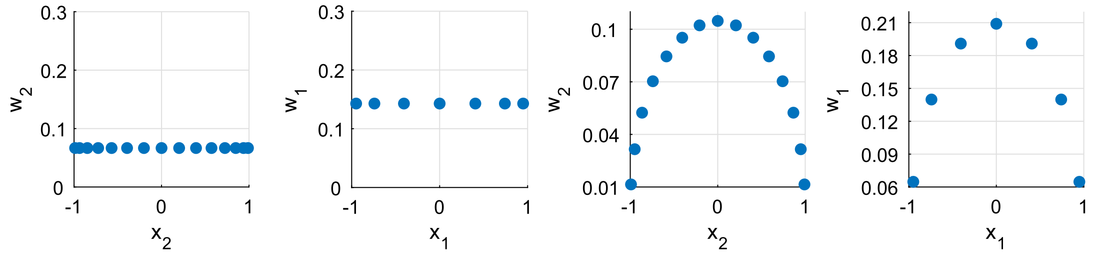

Fig. 2. Initial nodes and weights (two left plots), final nodes and weights (two right plots) for Legendre polynomial associated with $n_1 = 7, n_2 = 15, \alpha_1 = 13, \alpha_2 = 23$.

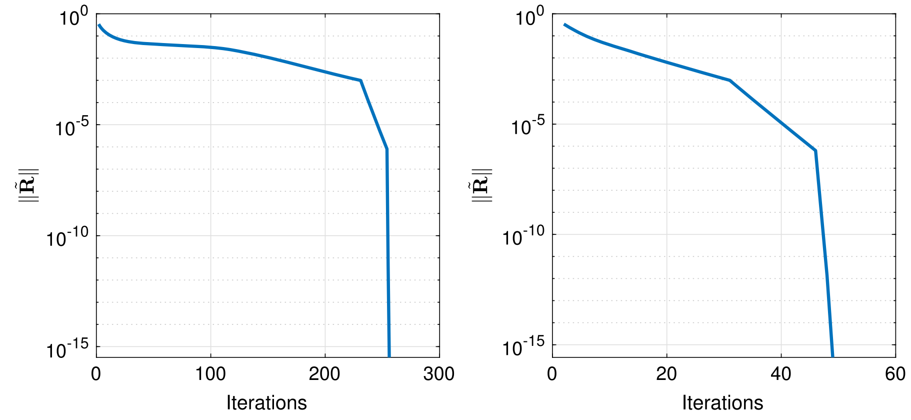

Fig. 3. Convergence speed using different regularization procedures: Tikhonov filter factors (left) and modified regularization discussed in Section 3.3 (right).

as the nested rule for the next step where $3 + 1 = 4$ points are added to find a new optimal 7-point rule. This procedure is continued for larger number of points.

It is noted that we only consider $\boldsymbol{R}_2$ residual and its associated constraints' residual (since $\boldsymbol{R}_1 = \boldsymbol{0}$ already) at each sequential optimization and compute $\boldsymbol{R}_2$ as a function of $(\boldsymbol{x}_2, \boldsymbol{w}_2)$. However, the Jacobian only includes the derivatives of $\boldsymbol{R}_2$ with respect to additional $\boldsymbol{x}_2 \backslash \boldsymbol{x}_1$ nodes and the weights $\boldsymbol{w}_2$. As such the resulting Newton update has $n_2 - n_1 + n_2 = 3n_1 + 2$ entries. These values are concatenated to n_1 zeros to update the current iteration $(\boldsymbol{x}_2, \boldsymbol{w}_2)$.

4.1.1. Verification of existing rules: bounded domains

In our numerical experiments for integration on bounded domains we always find the Kronrod relationship given n_1, i.e.,

$$n_2 = 2n_1 + 1, \qquad \alpha_1 = 2n_1 - 1, \qquad \alpha_2 = \begin{cases} 3n_1 + 1, & n_1 \text{ even} \\ 3n_1 + 2, & n_1 \text{ odd} \end{cases} \tag{29}$$

for all tested values of n_1. We will consider three different polynomial families: Legendre, Chebyshev and asymmetric Jacobi. We report existing results on Legendre Polynomial in this section and show the results for Chebyshev and Jacobi in section 4.1.3. Fig. 2 shows the initial and final nodes and weights for the Legendre polynomial associated with $n_1 = 7, n_2 = 15, \alpha_1 = 13, \alpha_2 = 23$.

To quantify the difference between the generated and existing rules in the literature, we compute the error for nodes and weights using $e_x = \|x_{opt} - x_{exs}\|_2 / \|x_{exs}\|_2$ and $e_w = \|w_{opt} - w_{exs}\|_2 / \|w_{exs}\|_2$ where subscript $._{exs}$ denotes existing for both main and nested rules. The existing values for the above example is from [1]. We find the error for the main and nested rules as $e_x = 4.43\text{e}{-10},\ 2.83\text{e}{-10}$ and $e_w = 4.98\text{e}{-9},\ 1.51\text{e}{-9}$ respectively.

In this example we also investigate the effect of different ways of regularization on the convergence speed. To this end in the first experiment we use Equations (27) and (28), and in the second experiment we use the regularization expression as provided at the end of Section 3.3. In this second experiment we add the regularization parameter directly to singular values as the residual falls below 0.5 i.e. $\|\widetilde{\boldsymbol{R}} \leq 0.5\|$. Fig. 3 shows the convergence in both cases. It is apparent that the optimization in the second case is faster as it only takes 49 iterations to converge while the fist case takes 256 iterations.

Fig. 4 shows the sequence of nested quadrature for Gauss-Kronrod-Patterson rules. These rules are associated with $n_2 = 3, 7, 15, 31, 63$ and $\alpha_2 = 5, 11, 23, 47, 95$. Our algorithm found a lower order ($\alpha_2 = 91$ instead of $\alpha_2 = 95$) for the 63-point rule which we report in the next section however we were able to recover these existing points when we used an initial guess close to the optimal points reported in [17]. The errors between our generated rules and the ones available in [17] are $e_x = 8.93\text{e}{-08},\ 8.40\text{e}{-08},\ 4.23\text{e}{-08},\ 5.86\text{e}{-08},\ 3.12\text{e}{-08}$ and $e_w = 9.17\text{e}{-08},\ 1.74\text{e}{-07}, 1.12\text{e}{-07}, 8.21\text{e}{-08}, 9.84\text{e}{-08}$ which are appreciably small.

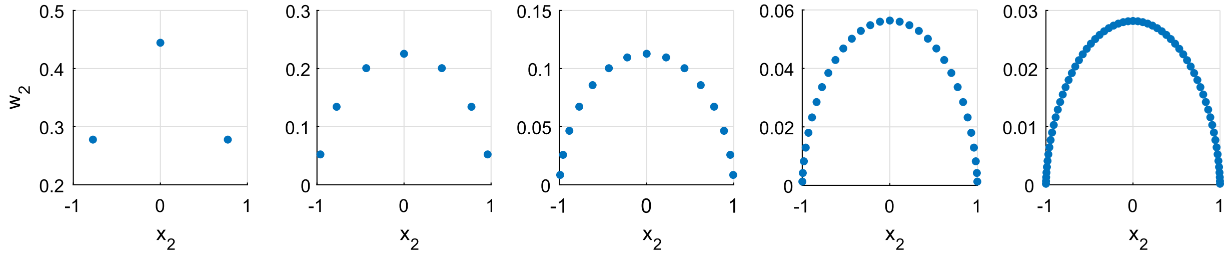

Fig. 4. Optimized nested sequence of quadrature rules $n_2 = 3, 7, 15, 31, 63$ for Legendre polynomial with orders $\alpha_2 = 5, 11, 23, 47, 95$.

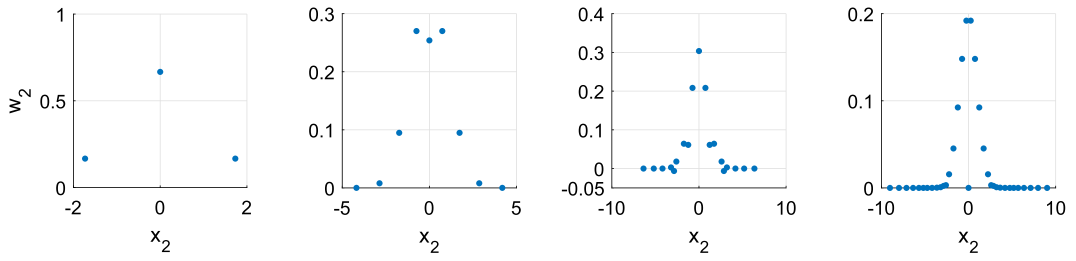

Fig. 5. Optimized nested sequence of quadrature rules $n_2 = 3, 9, 19, 35$ for weight function $\omega(x) \propto e^{-x^2}$.

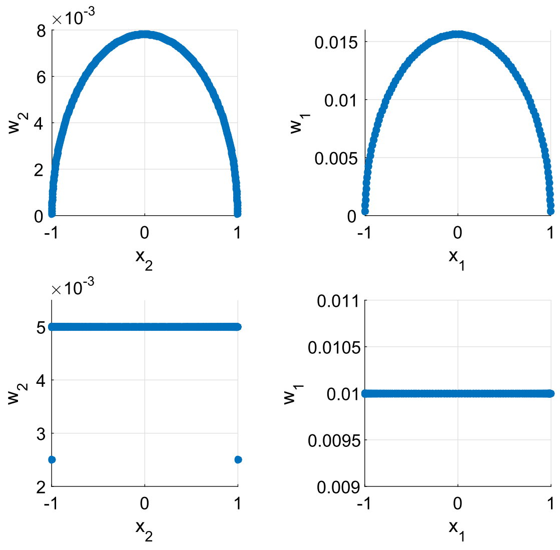

Fig. 6. Optimal main and nested nodes and weights for Legendre (top) and Chebyshev (bottom) polynomials associated with $n_1 = 100, n_2 = 201, \alpha_1 = 199, \alpha_2 = 301$.

4.1.2. *Verification of existing rules: unbounded domains*

To generate existing rules for unbounded domains we consider the sequence of nested rules associated with the Gaussian weight function $\omega(x) \propto e^{-x^2}$. We generate $n_2 = 3, 9, 19, 35$ nodes corresponding to polynomial orders $\alpha_2 = 5, 15, 29, 51$ for tolerance $\epsilon = 10^{-12}$. For $n_2 = 19, 35$ we used initial guesses close to those reported in [17] and were able to recover similar nodes cf. Fig. 5. The computed 19-point rule has two (symmetric) nodes with negative weights. To recover this rule we relaxed our optimization with no constraint on positive weights. This rule is the only quadrature with negative weights throughout this paper and we generated it to solely verify the existing rule.

Similarly to the previous example we compute the errors between the generated and existing rules which are $e_x =$ 4.79e–14, 1.23e–14, 7.82e–15, 5.12e–15 and $e_w =$ 1.25e–13, 8.56e–14, 3.47e–14, 2.21e–14.

4.1.3. *Generation of new rules: bounded domains*

The optimal quadrature for high polynomial order (and high number of nodes) i.e. $n_1 = 100, n_2 = 201, \alpha_1 = 199, \alpha_2 = 301$ for both cases of Legendre and Chebyshev is shown in Fig. 6. The MATLAB scheme takes 173 and 261 iterations which amount to 36.17 *sec* and 53.18 *sec* respectively for these cases on a personal desktop computer with Intel Core $i7 - 5930K$ @3.5 GHz CPU.

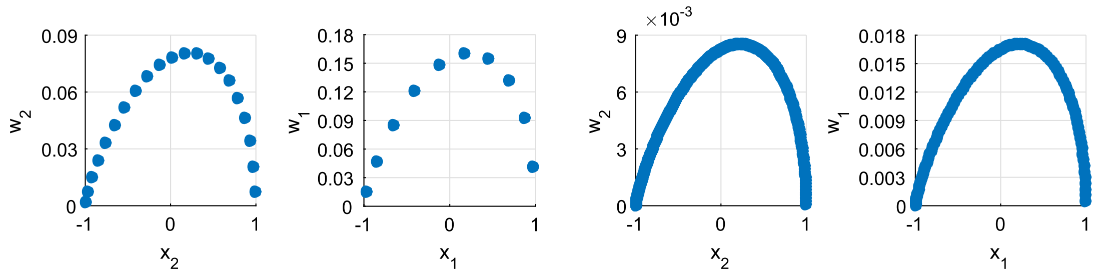

Fig. 7. Optimal main and nested rules for asymmetric Jacobi with $\alpha = 0, \beta = 0.3$ for $n_1 = 10, n_2 = 21, \alpha_1 = 19, \alpha_2 = 31$ (two left figures) and $n_1 = 100, n_2 = 201, \alpha_1 = 199, \alpha_2 = 301$ (two right figures).

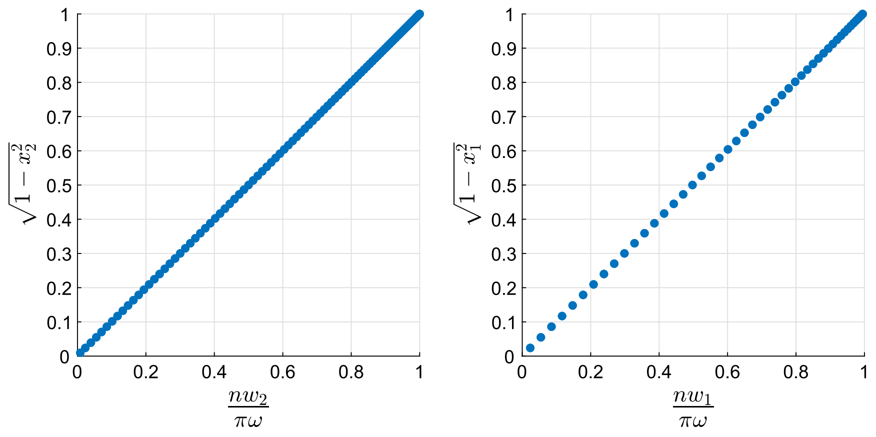

Fig. 8. Circle Theorem for main and nested rule associated with Legendre polynomial.

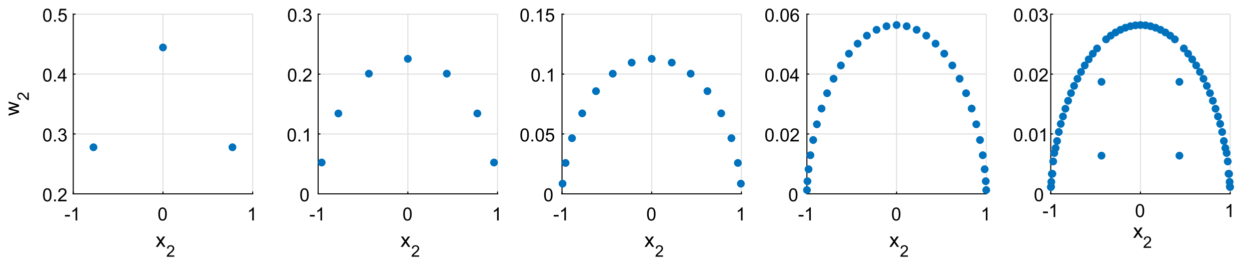

Fig. 9. Optimized nested sequence of quadrature rules $n_2 = 3, 7, 15, 31, 63$ for Legendre polynomial with orders $\alpha_2 = 5, 11, 23, 47, 91$.

The experiment for asymmetric Jacobi is associated with $\alpha = 0, \beta = 0.3$. Fig. 7 shows the optimal nodes and weights for $n_1 = 10$ and $n_1 = 100$. Again we emphasize that we find the Kronrod rule cf. Equation (29).

Now that we have quadrature nodes for high polynomial order we can test the well-documented *Circle Theorem*. The theorem states that the Gaussian weights, suitably normalized and plotted against the Gaussian nodes, lie asymptotically for large orders on the upper half of the unit circle centered at the origin in the case of Jacobi weight functions[18]. In other words,

$$\frac{nw}{\pi\,\omega(x)} \sim \sqrt{1-x^2} \quad \text{as} \quad n \to \infty$$

where x, w are the quadrature node and weight and $\omega(x)$ is the weight function evaluated at node x. Fig. 8 shows the above relationship for both main $n_2 = 201$ and nested $n_1 = 100$ rules associated with Legendre polynomial cf. Fig. 6.

The Gauss-Kronrod-Patterson rule associated with the Legendre polynomial is shown in Fig. 9. As mentioned earlier we find the nodes for highest order in this case without using the initial guess that we used to generate nodes in Fig. 4.

We also used our method to generate a sequence of nested points for quadrature under the Chebyshev weight. We generated these points for tolerance $\epsilon = 10^{-12}$ which attains polynomial accuracy similar to the Legendre case, cf. Fig. 10. For the last rule we started with 63 points, and we found after optimization that six of these points have negligible weights ($w \sim 10^{-15}$ which is comparable to our constraint tolerance). These points were automatically flagged for removal, so the final set has 57 points. The residual norm with these 57 points is $||\boldsymbol{R}_2||_2 = 1.51\text{e}{-14}$.

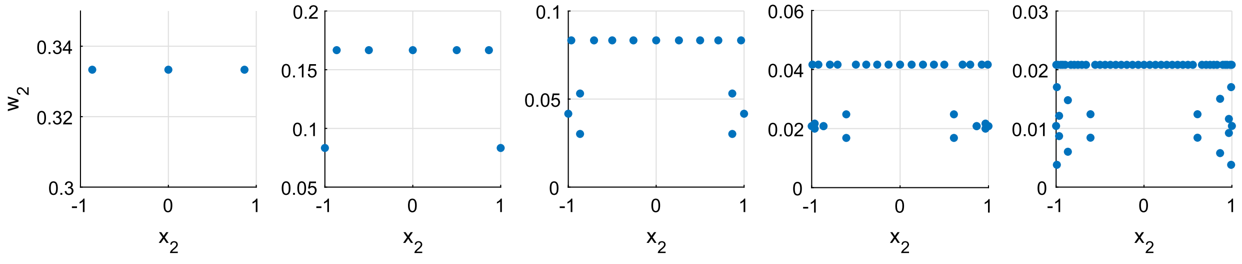

Fig. 10. Optimized nested sequence of quadrature rules $n_2 = 3, 7, 15, 31, 57$ for Chebyshev polynomial with orders $\alpha_2 = 5, 11, 23, 47, 95$.

Table 1
Nested rule for Hermite polynomials with $\rho_G = 0$.

	$i=1$	$i=2$	$i=1$	$i=2$	$i=1$	$i=2$	$i=1$	$i=2$
n_i	1	3	2	5	3	7	4	9
α_i	1	5	3	7	5	9	7	11
n_i	5	11	6	13	7	15	8	17
α_i	9	15	11	17	13	19	15	21
n_i	9	19	10	21	11	23	12	25
α_i	17	23	19	25	21	27	23	31
n_i	13	27	14	29	15	31		
α_i	25	33	27	35	29	37		

4.1.4. Generation of new rules: unbounded domains

We now use our numerical method to find nested rules for polynomial families whose orthogonality measure has support on infinite, domains such as the Hermite and Laguerre families.

The weight functions for these two cases are $\omega(x) \propto x^{\rho_G} e^{-x^2}$ and $\omega(x) \propto x^{\rho_L} e^{-x}$. It is easy to show that these two weight functions are transformable to each other with $x_L = x_G^2$ where x_L and x_G denote the domain for weight functions associated with Laguerre and Hermite polynomials. Using this transformation we can show $\rho_L = (\rho_G - 1)/2$. Having this transformation and having the ability to generate quadrature for any ρ_L or ρ_G one only needs to generate quadrature points for one of these families. For example generating an n-point rule where n is even for Hermite families is equivalent to generating an $n/2$-point rule for Laguerre families. However, it should be noted that the maximum integrable order for Laguerre is half of the Hermite case due to the $x_L = x_G^2$ transformation.

We find in our numerical experiments that the relationship between the number of points and the polynomial accuracy of the rule does not attain the accuracy of a Kronrod rule, i.e., given n_1 we do not achieve the parameter n_2, α_1, and α_2 specified in (29). Table 1 lists different values of the number of nodes and polynomial orders for successful cases i.e. cases that achieve tolerance less than 10^{-14}. The optimization for highest case in this table $n_1 = 15$ required 1755 iterations and 5.88 *sec* to achieve the desired tolerance. The results in the table demonstrate that in many cases the accuracy α_2 of the main rule is smaller than $3n_1 + 1$ or $3n_2 + 2$. To better demonstrate the shrinkage procedure as discussed in Remark 3.1 we show the initial guess that we used for optimization of $n_1 = 15,\ n_2 = 31$ rule in Fig. 11. Nodes on the left figure correspond to Gauss quadrature with $n = 31$ points. The maximum value for these nodes is $\max\{x\} = 6.9975$. On the right figure the same nodes are shrinked with the ratio $5.2203/6.9975 = 0.7462$ and used as an initial guess for the nested quadrature optimization. The numerator is indeed the maximum value of the Gauss quadrature nodes associated with order $\alpha_2 = 37$ which is attainable by our numerical procedure.

For cases with large number of nodes however we find that we can achieve higher order while maintaining a small tolerance, e.g., $\epsilon = 10^{-14}$. This might be attributed to the large number of degrees of freedom in this optimization as well as extremely small weights on the tails. Fig. 12 shows the optimal nodes and weights for two cases of a small number of nodes $n_1 = 15, n_2 = 31, \alpha_1 = 29, \alpha_2 = 37$ and high number of nodes $n_1 = 100, n_2 = 201, \alpha_1 = 199, \alpha_2 = 301$ for Hermite polynomial. The result for $n_1 = 100$ takes 1977 iterations and 177.32 sec. With our algorithm we can similarly repeat this experiment for $\rho_G = 1$ where we have shown the results in Fig. 13.

As discussed previously, we use the half of the optimized Hermite rule as the Laguerre rule. Fig. 14 shows the optimized Hermite rule $n_1 = 100, n_2 = 202, \alpha_1 = 199, \alpha_2 = 301$ and the Laguerre rule $n_1 = 50, n_2 = 101, \alpha_1 = 99, \alpha_2 = 150$ which takes 56.21 *sec* and 3253 iterations to achieve a residual norm $||\tilde{\boldsymbol{R}}||_2 \simeq 10^{-14}$. Note that n_2 is even in the case of Hermite rule and the accuracy of Laguerre rules are halved. It is also noted that the abscissa for these rules are according to $x_L = x_G^2$.

Finally we generate the Gauss-Kronrod-Patterson rule for Hermite polynomials with $\rho_G = 1$. The achieved orders for $n_2 = 3, 7, 15, 31$ are $\alpha_2 = 5, 9, 15, 35$ respectively cf. Fig. 15. We use these nodes in Section 4.2.2 to generate a Sparse Grid for integration in multiple dimensions.

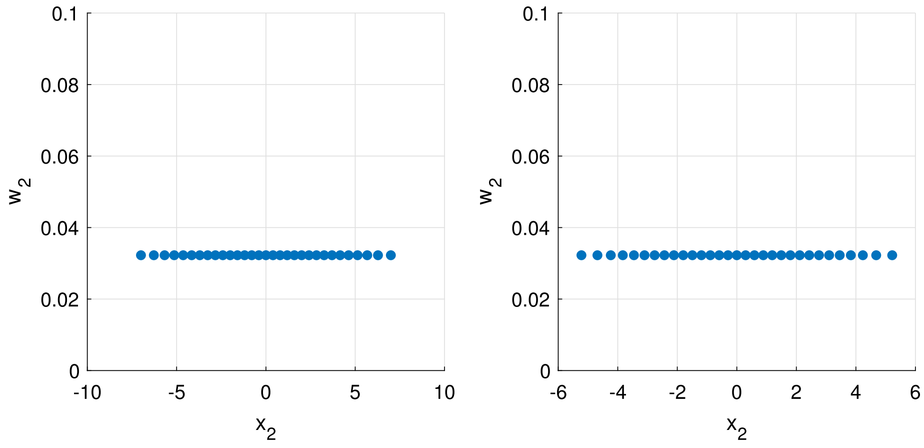

Fig. 11. Gauss quadrature nodes for $n = 31$ rule (left), the same nodes shrinked with the ratio 0.7462 and used in the optimization procedure (right).

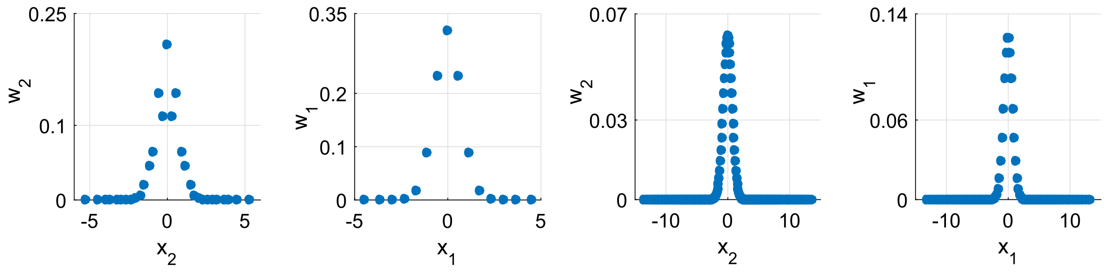

Fig. 12. Optimal main and nested rules for Hermite for $n_1 = 15, n_2 = 31, \alpha_1 = 29, \alpha_2 = 37$ (two left figures) and $n_1 = 100, n_2 = 201, \alpha_1 = 199, \alpha_2 = 301$ (two right figures) with $\rho_G = 0$.

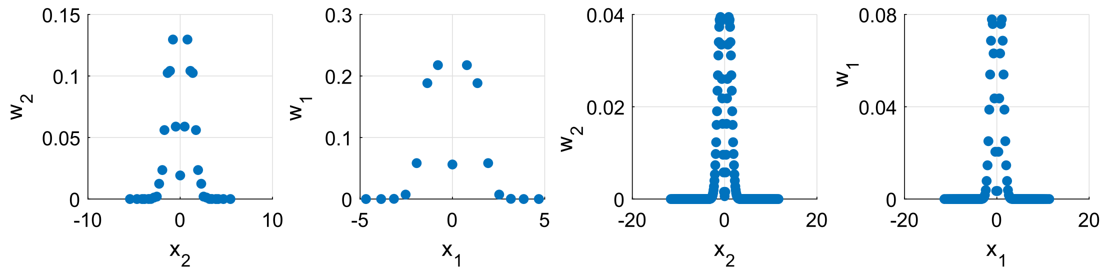

Fig. 13. Optimal main and nested rules for Hermite for $n_1 = 15, n_2 = 31, \alpha_1 = 29, \alpha_2 = 37$ (two left figures) and $n_1 = 100, n_2 = 201, \alpha_1 = 199, \alpha_2 = 301$ (two right figures) with $\rho_G = 1$.

4.1.5. Univariate integration: linear elastic problem

We investigate the accuracy of our nested quadrature rule via estimation of statistical moments for the displacement in a linear elastic structure with uncertain material properties. The L-bracket domain shown in Fig. 16 is partitioned into 978 standard triangular elements that are used to discretize a linear elastic PDE that predicts displacement. The modulus of elasticity is parameterized as one lognormal variable $E = 10^{-6} + \exp(\xi)$ for all elements. We are interested in the displacement, u, in the direction of point load, see Fig. 16. This displacement is a function of the elasticity, so that $u = u(\xi)$, where ξ is taken as a standard normal random variable. We estimate the mean and variance of $u(\xi)$ by generating nested quadrature rules in the ξ variable and evaluating u at the abscissae of these rules. Each evaluation of u requires the solution of a PDE, so this is an example of a case where parsimony of quadrature rules sizes is useful. The mean and variance of u are estimated via nested quadrature rules and are compared against "true" values, which are computed using a 100-point Gaussian quadrature rule.

For a given quadrature rule $(\xi_i, w_i)_{i=1}^{n}$, the mean and standard deviation are estimated as

$$\mu = \sum_{i=1}^{n} u(\xi_i) w_i, \quad \sigma = \sqrt{\sum_{i=1}^{n} u^2(\xi_i) w_i - \mu^2} \tag{30}$$

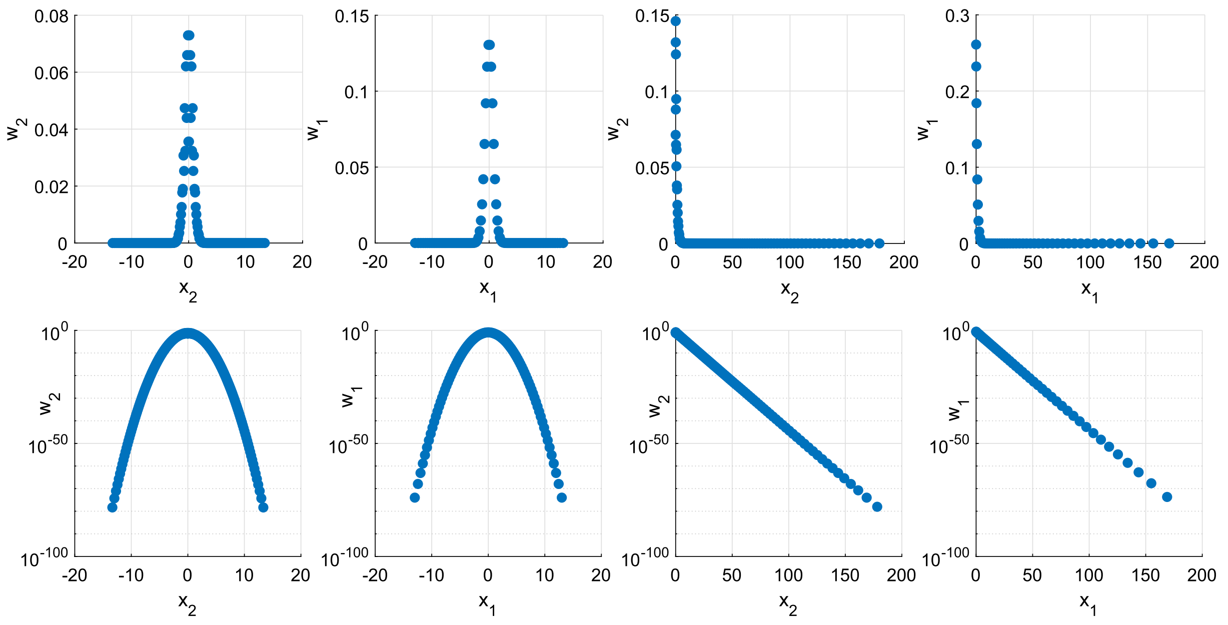

Fig. 14. Optimized Hermite rule for $n_1 = 100, n_2 = 202, \alpha_1 = 199, \alpha_2 = 301$ (two left figures) and Laguerre nested rules for $n_1 = 50, n_2 = 101, \alpha_1 = 99, \alpha_2 = 150$ (two right figures). The top and bottom figures show a linear and logarithmic scale, respectively, for the vertical axis.

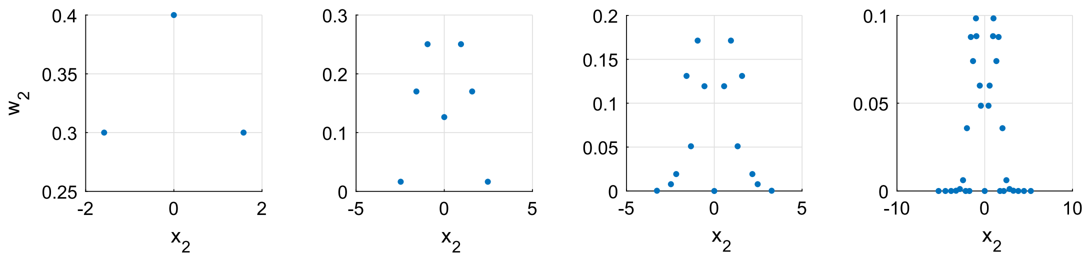

Fig. 15. Optimized nested sequence of quadrature rules $n_2 = 3, 7, 15, 31$ and $\alpha_2 = 5, 9, 15, 35$ for Hermite polynomials with $\rho_G = 1$.

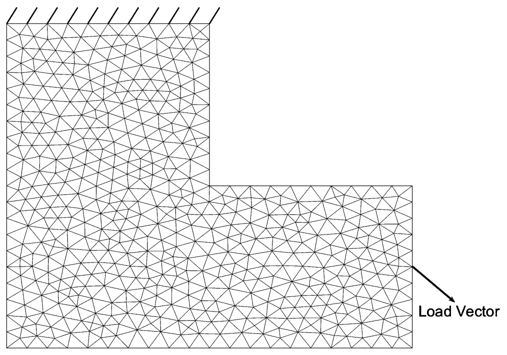

Fig. 16. Linear elastic L-bracket with random Young's modulus.

Subsequently the errors in mean and standard deviation are obtained as

$$e_\mu = |(\mu - \mu_{true})/\mu_{true}|, \quad e_\sigma = |(\sigma - \sigma_{true})/\sigma_{true}| \tag{31}$$

where μ_{true} and σ_{true} are the "true" mean and standard deviation. We also compute the error between the main and nested rule evaluations as

$$e_I = |(\mu_1 - \mu_2)/\mu_2| \tag{32}$$

where μ_1 and μ_2 are mean values evaluated with nested and main rule respectively.

Table 2
Error in mean and standard deviation for the linear elastic structure.

Quadrature rule	e_μ	e_σ	Number of nodes	e_I
Nested nodes	4.375e–06	8.757e–03	7	4.3752e–06
Main nodes	5.745e–10	8.240e–05	$7 \oplus 8 = 15$	
Gauss quadrature	4.9141e–12	1.2708e–08	15	
Nested nodes	2.918e–07	2.333e–03	8	2.9185e–07
Main nodes	2.550e–11	1.485e–05	$8 \oplus 9 = 17$	
Gauss quadrature	3.6890e–12	2.1421e–10	17	

The Poisson's ratio is $\nu = 0.3$, and the plane stress condition is assumed. We consider two sets for our analysis: i) a nested rule with $n_1 = 7, n_2 = 15$ set and compare with a 15-point Gauss quadrature which is the total number of points and ii) a nested rule with $n_1 = 8, n_2 = 17$ and compare with 17-point Gauss quadrature. We choose the Gauss quadrature sets such that their number is equal to n_2.

It is evident from Table 2 that the main (n_2-point) quadrature rules have less accuracy compared to n_2-point Gaussian quadrature. This is expected since Gaussian rules integrate higher-degree polynomials exactly. However, we see that the nested quadrature rules attain comparable accuracy for e_μ and e_I, which thus supports usage of these rules in cases when re-use of function evaluations is paramount.

4.2. Multivariate examples via sparse grids

We compute multivariate integration formulas via sparse grids, which manipulate univariate quadrature rules to form a multivariate quadrature rule. The ability to generate nested univariate quadrature rules, which is the main topic of this paper, yields sparse grid constructions that have a relatively small number of function evaluations. This idea is not new, but our procedure affords flexibility: we can generate nested rules for quite general univariate weight functions. We demonstrate the savings using this strategy on some test cases.

4.2.1. Sparse grids for multivariate quadrature

Sparse grids are multivariate quadrature rules formed from unions of tensorized univariate rules. Consider a tensorial Γ as in Section 2.1, and for simplicity assume that the univariate domains $\Gamma_j = \Gamma_1$ and weights $\omega_j = \omega_1$ are the same. Let $\mathbb{X}_i$ denote a univariate quadrature rule (nodes and weights) of "level" $i \geq 1$, and define $\mathbb{X}_0 = \emptyset$. The number of points n_i in the quadrature rule $\mathbb{X}_i$ is increasing with i, but can be freely chosen. For multi-index $\boldsymbol{i} \in \mathbb{N}^d$, a d-variate tensorial rule and its corresponding weights are

$$\mathbb{A}_{d,\boldsymbol{i}} = \mathbb{X}_{i_1} \otimes \ldots \otimes \mathbb{X}_{i_d}, \quad w^{(\boldsymbol{q})} = \prod_{r=1}^{d} w_{i_r}^{(q_r)}. \tag{33}$$

The difference between sequential univariate levels is expressed as

$$\Delta_i = \mathbb{X}_i - \mathbb{X}_{i-1}, \qquad i \geq 1, \tag{34}$$

This approximation difference is used to construct a d-variate, level-k-accurate sparse grid operator [19,20] for any $k \in \mathbb{N}$ as,

$$\mathbb{A}_{d,k} = \sum_{r=0}^{k-1} \sum_{\substack{\boldsymbol{i} \in \mathbb{N}^d \\ |\boldsymbol{i}| = d+r}} \Delta_{i_1} \otimes \ldots \otimes \Delta_{i_d} = \sum_{r=k-d}^{k-1} (-1)^{k-1-r} \binom{d-1}{k-1-r} \sum_{\substack{\boldsymbol{i} \in \mathbb{N}^d \\ |\boldsymbol{i}| = d+r}} \mathbb{X}_{i_1} \otimes \ldots \otimes \mathbb{X}_{i_d}, \tag{35}$$

where the latter equality is shown in [21].

If the univariate quadrature rule $\mathbb{X}_i$ exactly integrate univariate polynomials of order $2i - 1$ or less, then the Smolyak rule $\mathbb{A}_{d,k}$ is exact for d-variate polynomials of total order $2k - 1$ [22]. It is reasonable to use Gauss quadrature rules for the $\mathbb{X}_i$ to obtain optimal efficiency, but since the differences Δ_i appear in the Smolyak construction, then utilizing nested rules satisfying $\mathbb{X}_i \subset \mathbb{X}_{i+1}$ can generate sparse grids with many fewer nodes than non-nested constructions. One can use, for example, nested Clenshaw-Curtis rules [23], the nested Gauss-Patterson or Gauss-Kronrod rules [15,24,25], or Leja sequences [26].

Sparse grids is a popular rule for integration in many computational applications. The main reason is the easy construction of multidimensional rule from a univariate rule while yielding small number of points. As mentioned sparse grid construction results in fewer nodes by using nested univariate rules. Another alternative to sparse grid for integration in multi-dimensions is the *designed quadrature* which directly satisfies moment-matching conditions for multidimensional polynomial spaces, guarantees all positive weights and has been shown to use far fewer nodes for integration for the same level of accuracy [2].

Our multidimensional sparse grid rules are constructed via (35), but with tensorized quadrature rules formed via $\mathbb{X}_i$ that only approximately integrate polynomials. I.e., the univariate rules $\mathbb{X}_i$ only integrate polynomials up to the accurate certified by $\|\boldsymbol{R}\|_2 \le \epsilon$ from the optimization (14). This univariate error translates into an error committed for multivariate quadrature rules. For simplicity, we state this result for a tensorial probability density function with identical univariate marginals.

Proposition 4.1. *Assume $\omega(x)$ is a univariate probability density function. Let $\mathbb{X}_i$, $i = 1, \dots, L$ be a sequence of univariate quadrature rules, and for each i assume that the residual vector $\boldsymbol{R}$ defined in (13) and (15) satisfies $\|\boldsymbol{R}\|_2 < \epsilon$, where the residual vector for $\mathbb{X}_i$ is associated with a univariate polynomial space Π_{α_i}. Then, given some multi-index $\boldsymbol{i} \in \mathbb{N}_0^d$, we have for any $p \in \otimes_{q=1}^d \Pi_{\alpha_{i_q}}$,*

$$\left|\mathbb{A}_{d,\boldsymbol{i}}(p) - I(p)\right| \le \epsilon \|p\| d (1+\epsilon)^{d-1} \prod_{q=1}^{d} \sqrt{\alpha_{i_q} + 1},$$

where

$$I(p) := \int_{\Gamma} p(\boldsymbol{x}) \left(\prod_{q=1}^{d} \omega(x_q) \right) \mathrm{d}x_1 \cdots \mathrm{d}x_d,$$

and $\|p\|$ is the $\prod_{q=1}^d \omega(x_q)$-weighted $L^2(\Gamma)$ norm.

Proof. Given a multi-index $\boldsymbol{i} \in \mathbb{N}_0^d$, we define

$$\boldsymbol{\alpha} = \boldsymbol{\alpha}_{\boldsymbol{i}} := \left(\alpha_{i_1}, \dots, \alpha_{i_d}\right)^T \in \mathbb{N}_0^d,$$

Let $p \in \otimes_{q=1}^d \Pi_{\alpha_{i_q}}$. Then there are coefficients $\widehat{p}_{\boldsymbol{j}}$ such that

$$p(\cdot) = \sum_{\boldsymbol{j} \le \boldsymbol{\alpha}_{\boldsymbol{i}}} \widehat{p}_{\boldsymbol{j}} p_{\boldsymbol{j}}(\cdot),$$

where

$$p_{\boldsymbol{j}}(\boldsymbol{x}) = \prod_{k=1}^{d} p_{j_k}(x_k), \qquad \boldsymbol{x} = (x_1, \dots, x_d)^T \in \mathbb{R}^d.$$

Then

$$\left|\mathbb{A}_{d,\boldsymbol{i}}(p) - I(p)\right|^2 \le \left(\sum_{\boldsymbol{j} \le \boldsymbol{\alpha}_{\boldsymbol{i}}} \left|\widehat{p}_{\boldsymbol{j}}\right| \left|\mathbb{A}_{d,\boldsymbol{i}}(p_{\boldsymbol{j}}) - I(p_{\boldsymbol{j}})\right| \right)^2 \le \left(\sum_{\boldsymbol{j} \le \boldsymbol{\alpha}_{\boldsymbol{i}}} \widehat{p}_{\boldsymbol{j}}^2 \right) \left(\sum_{\boldsymbol{j} \le \boldsymbol{\alpha}_{\boldsymbol{i}}} \left|\mathbb{A}_{d,\boldsymbol{i}}(p_{\boldsymbol{j}}) - I(p_{\boldsymbol{j}})\right|^2 \right). \tag{36}$$

The first term, by Parseval's equality, is $\|p\|^2$. To bound the second term, we first show that, given $\boldsymbol{r}, \boldsymbol{s} \in \mathbb{R}^d$ satisfying

$$\sup_{q=1,\dots,d} \left|s_q - r_q\right| \le \epsilon, \qquad \sup_{q=1,\dots,d} |s_q| \le 1, \tag{37}$$

then $D_k(\boldsymbol{s}, \boldsymbol{r}) := \left| \prod_{q=1}^k s_q - \prod_{q=1}^k r_q \right|$ satisfies

$$D_k(\boldsymbol{s}, \boldsymbol{r}) \le k\epsilon(1+\epsilon)^{k-1}, \qquad k = 1, \dots, d. \tag{38}$$

This result can be established by induction, by first noting that $D_1(\boldsymbol{s}, \boldsymbol{r}) \le \epsilon$. For some $k \ge 2$ assume $D_{k-1} \le (k-1)\epsilon(1+\epsilon)^{k-2}$, then

$$\begin{aligned} D_k &= |\textstyle\prod_{q=1}^k s_q - \prod_{q=1}^k r_q| \\ &= |\textstyle\prod_{q=1}^k s_q - s_k \prod_{q=1}^{k-1} r_q + s_k \prod_{q=1}^{k-1} r_q - \prod_{q=1}^k r_q| \\ &= |s_k \left(\textstyle\prod_{q=1}^{k-1} s_q - \prod_{q=1}^{k-1} r_q \right) + (s_k - r_k) \textstyle\prod_{q=1}^{k-1} r_q| \\ &\le |s_k| D_{k-1} + |s_k - r_k| \textstyle\prod_{q=1}^{k-1} |r_q|. \end{aligned}$$

Since $|s_q| \le 1$ and $|s_q - r_q| \le \epsilon$, this implies that $|r_q| \le 1 + \epsilon$. Using the inductive hypothesis

$$
\begin{aligned}
D_k \quad &\le D_{k-1} + \epsilon \textstyle\prod_{q=1}^{k-1}(1+\epsilon) \\
&\le (k-1)\epsilon(1+\epsilon)^{k-2} + \epsilon(1+\epsilon)^{k-1} \\
&\le (k-1)\epsilon(1+\epsilon)^{k-1} + \epsilon(1+\epsilon)^{k-1} = k\epsilon(1+\epsilon)^{k-1}
\end{aligned}
$$

yields (38). Note then that

$$\left|\mathbb{A}_{d,\boldsymbol{i}}(p_{\boldsymbol{j}}) - I(p_{\boldsymbol{j}})\right| = \left|\prod_{q=1}^{d}\mathbb{X}_{i_q}(p_{j_q}) - \prod_{q=1}^{d} I(p_{j_q})\right|.$$

Since ω is a probability density, then $|I(p_{j_q})| \le 1$ for all j_q. Furthermore, if the univariate rules $\mathbb{X}_i$ comprising $\mathbb{A}_{d,\boldsymbol{i}}$ satisfy the residual condition $\|\boldsymbol{R}\|_2 \le \epsilon$ as in Algorithm 1, then

$$\left|\mathbb{X}_{i_q}(p_{j_q}) - I(p_{j_q})\right| = \left|R_{j_q}\right| \le \|\boldsymbol{R}\|_2 \le \epsilon.$$

Thus, defining $s_q = \mathbb{X}_{i_q}(p_{j_q})$ and $r_q = I(p_{j_q})$ satisfies (37), so that

$$\left|\prod_{q=1}^{d}\mathbb{X}_{i_q}(p_{j_q}) - \prod_{q=1}^{d} I(p_{j_q})\right| = D_d(\boldsymbol{s},\boldsymbol{r}) \le d\epsilon(1+\epsilon)^{d-1}.$$

Using this in (36) (and noting the summation has $\prod_{q=1}^{d}(\alpha_{i_q}+1)$ terms) yields the conclusion. $\square$

The above characterization expresses the error committed by a tensorized quadrature rule when the composite univariate rules commit ϵ error on a particular subspace. Our error bound does not directly translate into an error committed by a sparse grid construction, but it does suggest that sparse grid multivariate quadrature errors can also scale like ϵ. In addition, we observe in the following numerical experiments that our sparse grids constructed from ϵ-approximate univariate grids perform well in practice.

4.2.2. *Multivariate integration on sparse grids: nonlinear ODE*

As mentioned previously, sparse grids are a common tool for integration in multiple dimensions. Application of nested quadrature rules in construction of sparse grids are useful since they reduce the total number of nodes in a sparse grid, and come with inexpensive error estimates. In this example we use our nested quadrature rule in construction of sparse grids to estimate the statistical moments for a parameterized nonlinear ordinary differential equation.

We consider the Lotka-Volterra equations, classical predator-prey equations, which are primarily used to describe the dynamics of biological systems. In particular, the evolution of population for species x and y is modeled as

$$
\begin{aligned}
&\frac{\partial x}{\partial t} = ax - bxy, \quad x(0) = x_0 \\
&\frac{\partial y}{\partial t} = cxy - dy, \quad y(0) = y_0
\end{aligned}
$$

where x and y are the population of preys and predators and a, b, c, d are modeled as random variables

$$
\begin{aligned}
&a = \exp(0.1\xi_1) + 1, \quad b = 2\exp(0.1\xi_2) + 0.5 \\
&c = \exp(0.1\xi_3) + 2, \quad d = 3\exp(0.1\xi_4) + 1
\end{aligned}
$$

where the ξ_i are mutually independent and identically distributed random variables, each having distribution with weight $\omega(x) \propto xe^{-x^2}$ identical to the weight we used in Gauss-Kronrod-Patterson section 4.1.4. The initial population is $x_0 = 3$, $y_0 = 3$. We use a fourth order Runge-Kutta time integration method to simulate the time trajectory of the population for the range $t \in [0, 10]$ with the time-step $dt = 0.05$. Some solution realizations for the prey population are shown in Fig. 17.

We now estimate the mean and variance of the prey population at time 2, $x(2)$, which are computed as in (30) via two quadrature rules: i) a sparse grid constructed from univariate nested quadrature rules and ii) the sparse grid constructed with univariate Gauss quadrature rules. We compute the relative error in mean and standard deviation similarly to Equation (31) and use a 2881-point in $d = 4$ dimensions[17] to find the true mean and standard deviation.

We follow the sparse grid construction in [22] and use $|\mathbb{X}_1| = 1$, $|\mathbb{X}_2| = 3$, $|\mathbb{X}_3| = 3$, $|\mathbb{X}_4| = 7$, $|\mathbb{X}_5| = 7$, $|\mathbb{X}_6| = 7$-point univariate rules (where $|.|$ denotes the size of set) for accuracy levels $i = 1, \ldots, 6$. The construction in [22] yields a rule for integration of order $2i - 1$ corresponding to each level i. It should be noted that the three univariate rules i.e. $[1, 3, 7]$-point rules used in this example are nested consecutively i.e. 1-point rule is nested to the 3-point rule and 3-point rule is nested to 7-point rule as we generated them in Section 4.1.4.

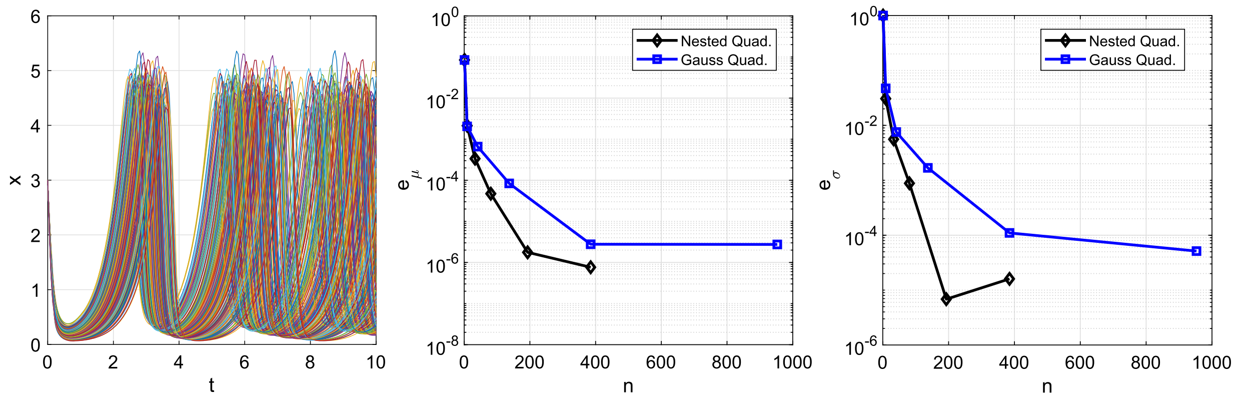

Fig. 17. Left: Realizations of prey's time history. Center and right: Relative error in mean and standard deviation for the nonlinear ODE.

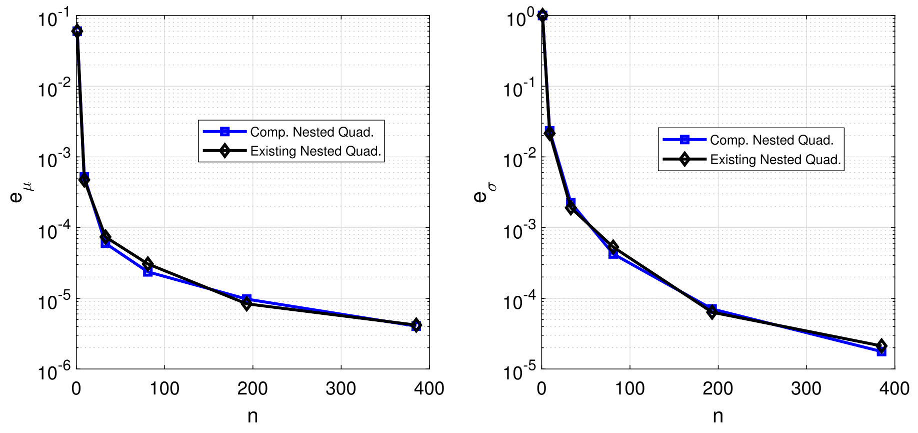

Fig. 18. Relative error in mean and standard deviation for nested quadrature rules with standard normal weight function: existing rule from [17] and the computed nested rule.

The sparse grid construction in $d = 4$ dimensions yields $n = [1, 9, 33, 81, 193, 385]$ and $n = [1, 9, 41, 137, 385, 953]$ for six accuracy levels corresponding to nested quadrature and Gauss quadrature respectively.

Fig. 17 shows the relative errors in mean and standard deviation with respect to both nested quadrature and Gauss quadrature. It is apparent that using the nested quadrature rule requires smaller number of function evaluations in addition to yielding relatively smaller errors. We note that the error in both cases almost saturates as shown in this figure. We attribute this issue to the insufficiency in the accuracy of the exact solution. In other words as we consider a very large number of nodes for the exact solution and use several levels for the integration, we expect to see more rapid decrease in the error. However the sole purpose of this example is to show the superior performance of the nested quadrature compared to Gaussian quadrature when used in the sparse grids which is evident from this figure.

It is also beneficial to compare the accuracy of computed nested quadrature rules with existing nested quadrature rules within a sparse grid framework. To this end we generate nested quadrature rules for standard normal weight function similarly to the [1,3,7]-point rules denoted by *KPN* (which is associated with standard normal weight function) in [17]. We use the same sparse grid construction in $d = 4$ as before which results in $n = [1, 9, 33, 81, 193, 385]$. Fig. 18 shows the normalized error for mean and standard deviation using the computed and existing nested quadrature rules. It is apparent that the computed rules almost yield the same accuracy as the existing rules in [17].

4.2.3. *Multivariate integration on sparse grids: elliptic PDE*

In this example we use sparse grid with nested quadrature to estimate the statistical moment for the steady state heat distribution. Such distribution is modeled via an elliptic PDE with the form

$$\begin{aligned} &-\nabla.(a(\boldsymbol{x}, \boldsymbol{\xi})\nabla u(\boldsymbol{x}, \boldsymbol{\xi})) = 1 && \boldsymbol{x} \in \Omega \\ &u(\boldsymbol{x}, \boldsymbol{\xi}) = u_0 && \boldsymbol{x} \in \partial\Omega \end{aligned}$$

where c is the heat conductivity which we consider as a random field in our example. We assume a Karhunen-Loeve expansion in the form of

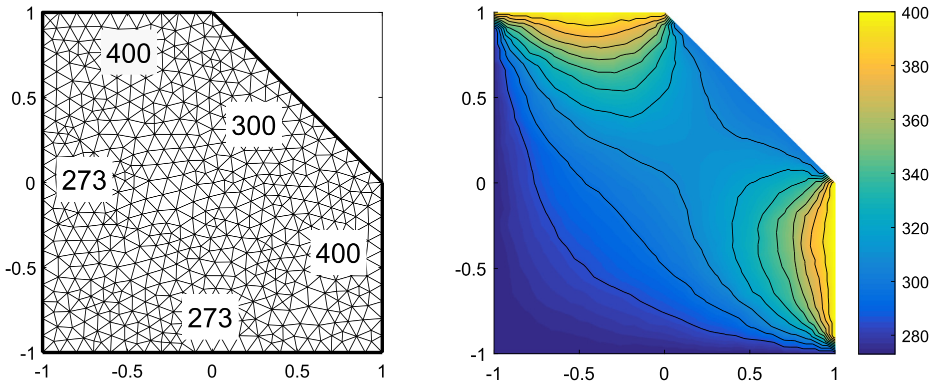

Fig. 19. Finite element mesh with Dirichlet boundary condition (left) and a solution realization for the heat equation (right).

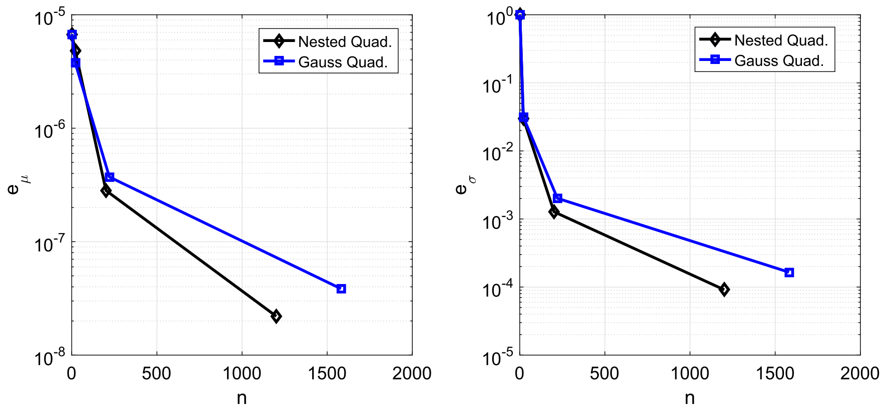

Fig. 20. Relative error in mean and standard deviation for the elliptic PDE.

$$a(\boldsymbol{x},\boldsymbol{\xi}) = \phi_0 + \sum_{i=1}^{d} \sqrt{\lambda_i}\phi_i(\boldsymbol{x})\xi_i$$

with $\xi_i \sim U[-1,1]$ and ϕ_0 is a positive constant. The eigenvalues and eigenmodes are obtained from decomposition of a Gaussian covariance kernel

$$C(\boldsymbol{x},\boldsymbol{x}') = \exp\Big(-\frac{||\boldsymbol{x}-\boldsymbol{x}'||_2^2}{2l_c^2}\Big), \tag{39}$$

with $l_c = \sqrt{2}/2$.

The spatial domain Ω and Dirichlet boundary condition are shown in Fig. 19. We truncate the expansion at $d = 10$, capturing almost 90% of the energy in the random field, $\sum_{i=1}^{10}\sqrt{\lambda_i}/\sum_{i=1}^{500}\sqrt{\lambda_i} = 0.8825$. The value ϕ_0 is fixed at $\phi_0 = 3$.

Similarly to previous example we use three univariate rules and consider [1,3,3,7]-point rules for accuracy levels $i = 1,\dots,4$. The sparse grid construction for $d = 10$ results in $n = [1,21,201,1201]$ and $n = [1,21,221,1581]$ nodes for four accuracy levels corresponding to nested quadrature and Gauss quadrature respectively.

Finally, we use a 5281-point rule for estimating the true mean and standard deviation and focus on a particular node with coordinate $[0.0037,-0.0024]$ in the spatial domain to study the convergence. Fig. 20 shows the relative errors in mean and standard deviation. It is again evident that relatively better accuracy is gained with smaller number of nodes when using a nested quadrature rule.

5. Concluding remarks

A numerical method for systematic generation of nested quadrature rules is presented. Our method uses a flexible bi-level optimization that solves the moment-matching conditions for the main and nested rule. The constraints, namely the node bounds and weight positivity are enforced throughout the optimization via a penalty method. We generalize the Gauss-Kronrod rule for various weight functions including those with finite/infinite and symmetric/asymmetric supports. The extension of algorithm to generate Gauss-Kronod-Patterson rules i.e. nested sequence of quadrature is also discussed. In particular results for the nested sequence of Chebyshev quadrature are tabulated which have not been reported elsewhere.

We used our nested univariate rules to construct sparse grids for integration in multiple dimensions. We showed the improved efficiency and accuracy of the resulting multidimensional quadrature on parameterized initial and boundary value problems when compared with Gauss quadrature-based sparse grids.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This research was sponsored by ARL under Cooperative Agreement Number W911NF-12-2-0023. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of ARL or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein. The first and third authors are partially supported by AFOSR FA9550-15-1-0467. The third author is partially supported by DARPA EQUiPS N660011524053 and NSF DMS 1720416.

References

[1] A. Kronrod, Nodes and Weights for Quadrature Formulae. Sixteen Place Tables, Nauka, Moscow, 1964, Translation by Consultants Bureau, New York.
[2] V. Keshavarzzadeh, R.M. Kirby, A. Narayan, Numerical integration in multiple dimensions with designed quadrature, SIAM J. Sci. Comput. 40 (4) (2018) A2033–A2061, https://doi.org/10.1137/17M1137875.
[3] G. Szegö, Orthogonal Polynomials, 4th edition, American Mathematical Soc., 1975.
[4] J. Stoer, R. Bulirsch, Introduction to Numerical Analysis, vol. 12, Springer-Verlag, New York, 2002.
[5] P. Davis, P. Rabinowitz, Methods of Numerical Integration, 2nd edition, Courier Corporation, 2007.
[6] G. Golub, J. Welsch, Calculation of Gauss quadrature rules, Math. Comput. 23 (1969) 221–230.
[7] W. Gautschi, Construction of Gauss-Christoffel quadrature formulas, Math. Comput. 22 (1968) 251–270.
[8] D. Bertsekas, Nonlinear Programming, second edition, Athena Scientific, 2008.
[9] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, 2004.
[10] E. Van den berg, M. Friedlander, Probing the Pareto frontier for basis pursuit solutions, SIAM J. Sci. Comput. 31 (3) (2008) 890–912.
[11] E. Van den berg, M. Friedlander, Sparse optimization with least-squares constraints, SIAM J. Optim. 21 (4) (2011) 1201–1229.
[12] G.H. Golub, C.F.V. Loan, Matrix Computations Johns Hopkins Studies in Mathematical Sciences, 3rd edition, The Johns Hopkins University Press, 1996.
[13] P. Hansen, Rank-Deficient and Discrete Ill-Posed Problems, SIAM, Philadelphia, 1998.
[14] P. Hansen, D. O'Leary, The use of the L-curve in the regularization of discrete ill-posed problems, SIAM J. Sci. Comput. 14 (6) (1993) 1487–1503.
[15] T. Patterson, The optimum addition of points to quadrature formulae, Math. Comput. 22 (1968) 847–856.
[16] A. Genz, B. Keister, Fully symmetric interpolatory rules for multiple integrals over infinite regions with Gaussian weight, J. Comput. Appl. Math. 71 (2) (1996) 299–309.
[17] F. Heiss, V. Winschel, Quadrature on sparse grids, http://www.sparse-grids.de/.
[18] W. Gautschi, The circle theorem and related theorems for Gauss-type quadrature rules, Electron. Trans. Numer. Anal. [electronic only] 25 (2006) 129–137, https://eudml.org/doc/127679?lang=it&limit=15.
[19] H. Bungartz, M. Griebel, Sparse grids, Acta Numer. 13 (2004) 147–269.
[20] S. Smolyak, Quadrature and interpolation formulas for tensor products of certain classes of functions, Sov. Math. Dokl. 4 (1963) 240–243.
[21] G. Wasilkowski, H. Wozniakowski, Explicit cost bounds of algorithms for multivariate tensor product problems, J. Complex. 11 (1) (1995) 1–56.
[22] F. Heiss, V. Winschel, Likelihood approximation by numerical integration on sparse grids, J. Econom. 144 (1) (2008) 62–80.
[23] D. Xiu, J.S. Hesthaven, High-order collocation methods for differential equations with random inputs, SIAM J. Sci. Comput. 27 (3) (2005) 1118–1139, https://doi.org/10.1137/040615201.
[24] M. Liu, Z. Gao, J.S. Hesthaven, Adaptive sparse grid algorithms with applications to electromagnetic scattering under uncertainty, Appl. Numer. Math. 61 (1) (2011) 24–37, https://doi.org/10.1016/j.apnum.2010.08.002.
[25] T. Gerstner, M. Griebel, Numerical integration using sparse grids, Numer. Algorithms 18 (3) (1998) 209–232, https://doi.org/10.1023/A:1019129717644.
[26] A. Narayan, J. Jakeman, Adaptive Leja sparse grid constructions for stochastic collocation and high-dimensional approximation, SIAM J. Sci. Comput. 36 (6) (2014) A2952–A2983, https://doi.org/10.1137/140966368, arXiv:1404.5663 [math.NA].