Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

RESEARCH ARTICLE

High-order cubature rules for tetrahedra

Jan Jaśkowiec[1] | N. Sukumar*[,2]

[1]Faculty of Civil Engineering, Cracow University of Technology, Warszawska 24, 31-155 Cracow, Poland

[2]Department of Civil and Environmental Engineering, University of California, Davis, CA 95616, USA

Correspondence
*N. Sukumar, Department of Civil and Environmental Engineering, University of California, One Shields Avenue, Davis, CA 95616, USA
Email: nsukumar@ucdavis.edu

In this paper, we construct new high-order numerical integration schemes for tetrahedra, with positive weights and integration points that are in the interior of the domain. The construction of cubature rules is a challenging problem, which requires the solution of strongly nonlinear algebraic (moment) equations with side conditions given by affine inequality constraints. We present a robust algorithm based on a sequence of three modified Newton procedures to solve the constrained minimization problem. In the literature, numerical integration rules for the tetrahedron are available up to order $p = 15$. We obtain integration rules for the tetrahedron from $p = 2$ to $p = 20$, which are computed using multi-precision arithmetic. For $p \leq 15$, our approach provides integration rules that have the same or fewer number of integration points than existing rules; for $p = 16$ to $p = 20$, our rules are new. Numerical tests are presented that verify the polynomial-precision of the cubature rules. Convergence studies are performed for the integration of exponential, rational, weakly singular and trigonometric test functions over tetrahedra with flat and curved faces. In all tests, improvements in accuracy is realized as p is increased, though in some cases nonmonotonic convergence is observed.

KEYWORDS:
numerical integration; polynomial basis; tetrahedral domain; least squares; strictly interior integration points; positive weights

1 | INTRODUCTION

Numerical integration using univariate Gaussian quadrature rules and their extensions to cubature rules in higher dimensions are widely adopted in computational methods to solve problems in engineering and the applied sciences. One-dimensional quadrature ($d = 1$) and multi-dimensional cubature ($d > 1$) rules in a domain $\Omega \subset \mathbb{R}^d$ consist of a set of integration points and scalar weights. An n-point cubature rule is represented as $\{\boldsymbol{x}_i, w_i\}_{i=1}^n$, where $\boldsymbol{x}_i \in \mathbb{R}^d$ is the i-th integration point and w_i is the corresponding weight. In one dimension, an n-point Gaussian quadrature rule can exactly integrate a polynomial of degree $2n-1$ and is optimal. No such optimality exists for integration schemes on domains in higher dimensions. Finite element

methods that are based on quadrilateral and hexahedral elements use tensor-product univariate Gaussian quadrature rules on the reference square and cube, respectively. For simplices and hypercubes, generalized Gaussian quadrature schemes have been developed, which solve the moment equations using least squares and a node elimination technique.[1,2] In such generalized Gaussian integration schemes, it is desirable that all integration points (nodes) are located within the domain ($\boldsymbol{x}_i \in \Omega$) and have positive weights ($w_i > 0$). These attributes for a cubature rule are known by the acronym PI, which stems from 'Positive weights and Inside nodes.'[3,4]

Simplices are preferred in finite element and boundary element analysis due to the ease of Delaunay mesh generation for complex geometries. Triangles are also used to represent surfaces in three dimension for applications in geometric design and computer graphics. Numerical integration schemes on triangles have been the subject of significant research, with many early contributions in the latter part of the last century.[5,6,7,8,9,10,11] This effort has sustained over the past two decades with greater emphasis on constructing high-order integration rules on triangles.[1,2,12,13] With an eye on polygonal finite element methods, generalized Gaussian cubature rules have also been constructed for convex and nonconvex polygons.[14] A different approach to constructing cubature schemes is pursued by Bucero et al,[15] who interpret the moment matrix in terms of the computation of truncated Hankel operators with flat extensions, and obtain cubature rules by solving a hierarchy of convex optimization problems. They obtain minimal number of integration points for low-order schemes on the square and construct an integration rule for Wachspress basis functions[16] on the pentagon. In comparison to the triangle, less effort has been placed on developing integration rules for tetrahedra. Early contributions in this area have been on low-order integration rules.[17,18,19,20] Besides tetrahedra in 3D, cubature rules have also been proposed for prisms, pyramids, as well as hexahedra.[21,22,23,24] For a comprehensive listing of some of the early studies on cubature rules for domains in 2D and 3D, the interested reader can refer to Cools and Rabinowitz[25] and Cools.[26]

High-order cubature rules on tetrahedral elements are used in the p-version of the finite element method.[27,28,29,30] Such integration schemes are also pertinent in high-order discontinuous Galerkin methods on curvilinear meshes.[31] Grundmann and Möller[32] proposed a general formula for integration rules over n-simplex domains. However, the cubatures generated by this rule have negative weights and the number of integration points for higher orders are not optimal. Over the past decade, only a few contributions have appeared that have produced cubature rules on a tetrahedron,[2,3,33,34,35] with integration rules of order $p = 15$ being the maximum that is currently available.[2] The presence of only a handful of studies in this important topic in computational mathematics is due to the fact that with increasing p in 3D, constructing cubature rules is a notoriously difficult problem. When p is greater than 10, convergence of most algorithms slows down and for $p > 15$ often nonconvergence occurs—this is due to the fact that one must solve strongly nonlinear equations with inequality constraints that result in an ill-conditioned Jacobian matrix when using high-order basis functions, and truncation errors occur due to computations in finite-precision arithmetic. In Zhang et al,[33] two symmetric integration rules for the tetrahedron of orders 9 and 14 are generated with 48 and 236

points, respectively. Barycentric coordinates are applied so that the tetrahedron is divided into four symmetric orbits so that the cubature rule is constructed on a smaller domain. A set of new and efficient tetrahedral cubature rules are proposed in Xiao and Gimbutas.[2] They realize rules up to $p = 15$ with fewer number of integration points in comparison to earlier schemes. The least squares Newton methods with orthogonal polynomial basis functions is utilized with symmetric partitioning of the domain. The relatively fewer number of integration points is achieved due to use of the node elimination procedure. Shunn and Ham[34] describe a family of symmetric integration rules for tetrahedra. The distribution of points in each cubature rule is based on an underlying so-called cubic close-packed grid, and the precise point locations and weights are optimized to reduce the truncation error in the cubature approximation. This approach results in a family of symmetric rules up to $p = 7$ for a tetrahedron. In William et al,[35] a sphere close-packed lattice arrangement of points is utilized for formulating symmetric quadrature rules on simplices, including the tetrahedron. Witherden and Vincent[3] use the refinement approach of Zhang et al[33] to generate new symmetric schemes for tetrahedron and other 3D elements. Yacobi[29] compares the efficiency and performance of some of the well-known tetrahedral cubature rules.[3,12,18,33]

In this paper, a new approach is presented to solve the nonlinear constrained problem, which delivers an integration rule with minimal number of integration points and meets the PI criteria. Unlike existing methods in the literature, we do not directly construct orthogonal basis functions nor use symmetric partitioning of the domain. In addition, we begin the search for a cubature rule using a lower bound estimate for the number of integration points, which is in contrast to most cubature algorithms in the literature. We also choose monomials as basis function, which are easy to integrate and furthermore their derivatives are simple to evaluate to form the Jacobian matrix. Since with a power basis, the Jacobian matrix become ill-conditioned for high-order problems, we resolve this issue via use of a preconditioning procedure at each iteration step of the algorithm. This ensures that the modified Jacobian matrix is always well-conditioned throughout the computations. A sequence of three modified Newton procedures is adopted to construct high-order rules. Even though the overall algorithm may require user-intervention, it is robust and efficient for the generation of high-order cubature rules. We point out that the use of monomial basis functions is particularly attractive to devise cubature rule on arbitrarily-shaped convex domains in 2D and 3D since the integrals of monomials can be readily computed.[36] For $p \leq 15$, we present integration rules that have the same or fewer number of integration points than those available in the literature. Furthermore, for $p = 16$ to $p = 20$, we obtain new integration rules for the tetrahedron.

The structure of the remainder of this paper follows. The algorithmic details for constructing integration scheme for tetrahedra are presented in Section 2. The description of the cubatures obtained in this paper are presented in Section 3, with specifics on the use of multi-precision arithmetic in the computations. The main results of this paper are cubature schemes from $p = 2$ up to $p = 20$, which are provided in the supplementary materials. Two verification tests are performed in Section 4, one on a single tetrahedron and the second one on an unstructured tetrahedral mesh. In addition, the sound accuracy of the cubature rules is

demonstrated for integrating exponential, rational, weakly singular and trigonometric test functions over tetrahedra with flat and curved faces. Finally, the main findings from this work are provided in Section 5.

2 | ALGORITHM FOR INTEGRATION SCHEMES

In this section, the algorithm is presented to generate cubature schemes over a tetrahedron, though the same approach is also applicable to any other 2D or 3D convex domain. We consider the standard tetrahedral domain that is given by

$$T = \{(x, y, z) : 0 \leq x \leq 1,\ 0 \leq x + y \leq 1,\ 0 \leq x + y + z \leq 1\}. \tag{1}$$

The integration of an arbitrary function $f(\boldsymbol{x})$ over the domain T is given by a cubature formula of the form

$$\int_T f(\boldsymbol{x})\, d\boldsymbol{x} \approx |T| \sum_{i=1}^{n} f(\boldsymbol{x}_i) w_i, \tag{2}$$

where the pair $\{\boldsymbol{x}_i, w_i\}_{i=1}^{n}$ represents the cubature rule with $\boldsymbol{x}_i$ the i-th integration point, w_i the i-th weight, n is the number of integration points, and $|T| = 1/6$ is the volume of the reference tetrahedron. Note that for any p-order cubature scheme, the weights must sum to unity.

Since any smooth function $f(\boldsymbol{x})$ can be well-approximated by a polynomial (and so is its integral), we require the above integration scheme to be exact for all polynomials up to order p. Such a scheme is called a p-order scheme. Let $\mathbb{P} = \{\Phi_i(\boldsymbol{x})\}_{i=1}^{\ell}$ be a set of basis (polynomial) functions that span any p-th order polynomial. In 1D, $\ell = p+1$; in 2D, $\ell = (p+1)(p+2)/2$; and in 3D, $\ell = (p+1)(p+2)(p+3)/6$. The objective is to determine the integration points and weights that can exactly integrate all functions in $\mathbb{P}$. Note that a p-order integration scheme is nonunique, and hence for the same n various different schemes are possible. In this paper, we assume the PI criteria is met—all integration points are located inside the tetrahedron and the weights are positive. For efficiency in computational methods such as finite elements, discontinuous Galerkin, and others, it is desirable that the number of integration point is as small as possible.

For a cubature scheme of order p to exactly integrate all functions in $\mathbb{P}$, we let $f(\boldsymbol{x}) := \Phi_i(\boldsymbol{x})$ $(i = 1, 2, \ldots, \ell)$ in (2), which leads us to the moment equations:

$$\boldsymbol{M}\boldsymbol{w} = \boldsymbol{p}, \tag{3a}$$

where

$$\boldsymbol{M} = |T| \begin{bmatrix} \Phi_1(\boldsymbol{x}_1) & \Phi_1(\boldsymbol{x}_2) & \dots & \Phi_1(\boldsymbol{x}_n) \\ \Phi_2(\boldsymbol{x}_1) & \Phi_2(\boldsymbol{x}_2) & \dots & \Phi_2(\boldsymbol{x}_n) \\ \dots & \dots & \dots & \dots \\ \Phi_\ell(\boldsymbol{x}_1) & \Phi_\ell(\boldsymbol{x}_2) & \dots & \Phi_\ell(\boldsymbol{x}_n) \end{bmatrix}, \quad \boldsymbol{w} = \begin{Bmatrix} w_1 \\ w_2 \\ \dots \\ w_n \end{Bmatrix}, \quad \boldsymbol{p} = \begin{Bmatrix} \int_T \Phi_1(\boldsymbol{x})\, d\boldsymbol{x} \\ \int_T \Phi_2(\boldsymbol{x})\, d\boldsymbol{x} \\ \dots \\ \int_T \Phi_\ell(\boldsymbol{x})\, d\boldsymbol{x} \end{Bmatrix}. \tag{3b}$$

The moment equations are nonlinear algebraic equations where the location of the cubature points, weights, and n are unknown. A lower bound estimate for n in $\mathbb{R}^d$ is $n_{\text{lb}} = \lceil \ell/(d+1) \rceil$, and therefore $n_{\text{lb}} = \lceil \ell/4 \rceil$ in 3D. Define the error vector

$$\boldsymbol{f} := \boldsymbol{M}\boldsymbol{w} - \boldsymbol{p}. \tag{4}$$

To determine the unknowns in the cubature scheme, the following nonlinear system of equations are solved:

$$\boldsymbol{f}(\boldsymbol{z}) = \boldsymbol{0}, \tag{5a}$$

with additional inequality constraints to meet the PI rules:

$$\boldsymbol{A}\boldsymbol{x}_i < \boldsymbol{b}, \quad \boldsymbol{A} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 1 & 1 \end{bmatrix}, \quad \boldsymbol{b} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{Bmatrix}, \tag{5b}$$

$$w_i > 0 \qquad (i = 1, 2, \dots, n), \tag{5c}$$

where $\boldsymbol{z} = [w_1\ x_1\ y_1\ z_1\ w_2\ x_2\ y_2\ z_2\ \dots w_n\ x_n\ y_n\ z_n]^T$ is the extended vector that consists of the unknowns to be found. The inequality constraint in (5b) ensures that $\boldsymbol{x}_i \in T$ and (5c) is the positivity constraint on the weights.

To compute $\boldsymbol{p}$, the integrals of all polynomial basis functions over T have to be computed. The integral of a monomial over T is given by[33]

$$\int_T x^r y^s z^t\, d\boldsymbol{x} = \frac{r!\, s!\, t!}{(r+s+t+3)!}. \tag{6}$$

The problem in (5) is strongly nonlinear, and an iterative procedure based on Newton's method is applied. Equation (6) is used to precompute the integrals of monomials in multi-precision (128 digits) to form $\boldsymbol{p}$ in (3b), and the results are stored in a data file. To enhance robustness and accelerate convergence, a sequence of three stages of modified Newton procedures are adopted. In the first stage, instead of a standard Newton procedure we solve a minimization problem with inequality constraints using a fixed step size for the update; in the second stage, a quadratic minimization problem with equality and inequality constraints is considered using step size control to accelerate convergence; and finally in the third stage a unit step size is used with the search direction given as the solution of a quadratic minimization problem with equality constraints. In the second stage, second-order Newton update is used for $p > 13$. The computations in the first and second stages are always carried out using double precision. The initial iterations of the third stage are performed in double precision to obtain a solution for $\boldsymbol{z}$, and the latter iterations are carried out in multi-precision arithmetic (128 digits) to obtain the solution $\boldsymbol{z}$ with high-precision. For moderate p, the second stage can be omitted, whereas for $p > 8$ all three stages are required to construct a cubature rule. Specific details on all three stages of the algorithm follows in Sections 2.1–2.3.

2.1 | Stage 1

In the first stage of the algorithm, we start with $n_{\text{lb}} = \lceil (p+1)(p+2)(p+3)/24 \rceil$ randomly generated points in the interior of the tetrahedron. The Newton equations to solve $\boldsymbol{f}(\boldsymbol{z}) = \boldsymbol{0}$ in (5a) are:

$$\boldsymbol{z}_{k+1} = \boldsymbol{z}_k + \alpha \Delta \boldsymbol{z}, \tag{7a}$$

$$\bar{\boldsymbol{J}}_k \Delta \boldsymbol{z} + \bar{\boldsymbol{f}}_k = \boldsymbol{0}, \tag{7b}$$

where

$$\bar{\boldsymbol{J}}_k = \boldsymbol{B}_k \boldsymbol{J}_k, \quad \bar{\boldsymbol{f}}_k = \boldsymbol{B}_k \boldsymbol{f}_k, \tag{7c}$$

$$\boldsymbol{f}_k = \boldsymbol{f}\big(\boldsymbol{z}_k\big), \quad \boldsymbol{J}_k = \frac{\partial \boldsymbol{f}_k}{\partial \boldsymbol{z}}. \tag{7d}$$

In (7), $\boldsymbol{z}_k$ and $\boldsymbol{J}_k$ are the solution vector and Jacobian matrix, respectively, at the k-th iteration step, $\Delta \boldsymbol{z}$ is the increment in $\boldsymbol{z}$, α is the step size, and $\boldsymbol{B}_k$ is a preconditioning matrix that is used to construct the modified Jacobian matrix $\bar{\boldsymbol{J}}_k$. The details on forming the preconditioning matrix is given in Section 2.4. At each iteration step, the vector $\boldsymbol{z}_k$ has to meet the PI rules in (5b) and (5c). However, for randomly generated points, it is highly unlikely that both (7b) and the PI rules are satisfied. Hence, instead of solving the Newton update equations in (7), we formulate a constrained minimization problem with inequality constraints to ensure that the integration points are in interior of the tetrahedron and the integration weights are positive at each iteration step.

To this end, we seek to find $\Delta \boldsymbol{z}$ that solves the following constrained least squares minimization problem:

$$\min_{\Delta \boldsymbol{z}} \frac{1}{2} \| \bar{\boldsymbol{J}}_k \Delta \boldsymbol{z} + \bar{\boldsymbol{f}}_k \|^2, \tag{8a}$$

subject to the inequality constraints

$$\boldsymbol{C} \Delta \boldsymbol{z} < \boldsymbol{d}, \tag{8b}$$

where (8b) uses the inequality constraints in (5b) and (5c). In order to solve (8), the `Matlab` optimization package `lsqlin` is used, which solves box-constrained linear least-squares problems.[37] At this stage, a fixed step size α is used. For $p < 15$, α is set to 0.1, whereas for higher orders it is set to 0.01 (to avoid divergence) for the first hundred iterations and then to 0.1 for the subsequent iterations. The search direction vector $\Delta \boldsymbol{z}$ that is obtained in each iteration step is used to update $\boldsymbol{z}$ via (7a). The stopping criterion is: $\| \boldsymbol{f}(\boldsymbol{z}_{k+1}) \| < \epsilon_1$, where ϵ_1 is a user-specified tolerance. As a guideline, we use $\epsilon_1 = 10^{-8}$ for $p < 12$ and $\epsilon_1 = 10^{-3}$ for $p = 20$.

2.2 | Stage 2

We use the output $\boldsymbol{z}$ from the previous stage as the input in this stage. Note that the solution from stage 1 does not satisfy (5a) to sufficient accuracy, even though the right-hand side of (7b) is close to the zero vector. To find a better approximation to $\boldsymbol{f}(\boldsymbol{z}) = \boldsymbol{0}$, we use a second-order Taylor expansion of $\boldsymbol{f}$:

$$\boldsymbol{f}(\boldsymbol{z} + \Delta \boldsymbol{z}) \approx \boldsymbol{f}(\boldsymbol{z}) + \boldsymbol{J}(\boldsymbol{z}) \Delta \boldsymbol{z} + \frac{1}{2} \Delta \boldsymbol{z}^T \boldsymbol{H}(\boldsymbol{z}) \Delta \boldsymbol{z} = \boldsymbol{0}, \tag{9}$$

where $\boldsymbol{H}$ is the third order Hessian tensor, $H_{ijk} = \frac{\partial^2 f_i}{\partial z_j \partial z_k}$. Since a quadratic term in $\Delta \boldsymbol{z}$ appears in (9), we determine $\Delta \boldsymbol{z}$ through an inner iteration procedure:

$$\Delta \boldsymbol{z}_{j+1} = \Delta \boldsymbol{z}_j + \Delta \Delta \boldsymbol{z}. \tag{10}$$

The quadratic term in (9) is linearized to first order to yield:

$$\begin{aligned} \frac{1}{2} \Delta \boldsymbol{z}_{j+1}{}^T \boldsymbol{H} \Delta \boldsymbol{z}_{j+1} &\approx \frac{1}{2} \Delta \boldsymbol{z}_j{}^T \boldsymbol{H} \Delta \boldsymbol{z}_j + \Delta \boldsymbol{z}_j{}^T \boldsymbol{H} \Delta \Delta \boldsymbol{z} \\ &= \frac{1}{2} \Delta \boldsymbol{z}_j{}^T \boldsymbol{H} \Delta \boldsymbol{z}_j + \Delta \boldsymbol{z}_j{}^T \boldsymbol{H} \left(\Delta \boldsymbol{z}_{j+1} - \Delta \boldsymbol{z}_j \right) \\ &= -\frac{1}{2} \Delta \boldsymbol{z}_j{}^T \boldsymbol{H} \Delta \boldsymbol{z}_j + \Delta \boldsymbol{z}_j{}^T \boldsymbol{H} \Delta \boldsymbol{z}_{j+1}, \end{aligned} \tag{11}$$

where (10) is used in the second line of the above equation. On using (9), the inner iterative procedure to obtain $\Delta \boldsymbol{z}_{j+1}$ can be developed using the preconditioning matrix to give

$$\boldsymbol{B} \left(\boldsymbol{J} + \Delta \boldsymbol{z}_j{}^T \boldsymbol{H} \right) \Delta \boldsymbol{z}_{j+1} + \boldsymbol{B} \left(\boldsymbol{f} - \frac{1}{2} \Delta \boldsymbol{z}_j{}^T \boldsymbol{H} \Delta \boldsymbol{z}_j \right) = \boldsymbol{0}. \tag{12}$$

For the second-order Newton update, the problem is formulated as: find $\Delta \boldsymbol{z}_{j+1}$ that solves the following constrained quadratic minimization problem:

$$\min_{\Delta \boldsymbol{z}_{j+1}} \frac{1}{2} \|\Delta \boldsymbol{z}_{j+1}\|^2, \tag{13a}$$

subject to the equality and inequality constraints that are given by

$$\boldsymbol{B}_k \left(\boldsymbol{J}_k + \Delta \boldsymbol{z}_j{}^T \boldsymbol{H}_k \right) \Delta \boldsymbol{z}_{j+1} + \boldsymbol{B}_k \left(\boldsymbol{f}_k - \frac{1}{2} \Delta \boldsymbol{z}_j{}^T \boldsymbol{H}_k \Delta \boldsymbol{z}_j \right) = \boldsymbol{0}, \tag{13b}$$

$$\boldsymbol{C} \Delta \boldsymbol{z}_{j+1} < \boldsymbol{d}. \tag{13c}$$

The problem in (13) is solved using the `Matlab` package `lsqlin`. At each outer step k, fewer than 5 inner iterations are needed for convergence. The step size α is determined by a one-dimensional Newton scheme, which is presented in Section 2.2.1. Once α and $\Delta \boldsymbol{z}$ are known, they are used to form $\boldsymbol{z}_{k+1}$ via (7a), and the process is repeated until the stopping criterion $\|\boldsymbol{f}(\boldsymbol{z}_{k+1})\| < \epsilon_2$ is met, where ϵ_2 is a user-specified tolerance. As a guideline, we use $\epsilon_2 = 10^{-12}$ for $p < 12$ and $\epsilon_2 = 10^{-5}$ for $p = 20$.

2.2.1 | Computing the step size

In (7a), α is the step size in the direction given by the vector $\Delta \boldsymbol{z}$. In a full Newton method $\alpha = 1$, whereas in a guarded or damped Newton method with step size control, $0 < \alpha < 1$. Given $\boldsymbol{z}_k$ and $\Delta \boldsymbol{z}$, consider the scalar error function

$$r(\alpha) = \frac{1}{2} \boldsymbol{f}(\boldsymbol{z}_k + \alpha \Delta \boldsymbol{z}) \cdot \boldsymbol{f}(\boldsymbol{z}_k + \alpha \Delta \boldsymbol{z}). \tag{14}$$

To find the optimal step size, we consider the following box-constrained one-dimensional minimization problem:

$$\min_{\alpha} r(\alpha), \quad \epsilon < \alpha < 1, \tag{15}$$

where $\epsilon = 10^{-3}$ is used in the computations. Since a stationary point of $r(\alpha)$ satisfies $r'(\alpha) = 0$, we perform a Taylor series expansion of $r(\alpha)$ to first order to obtain:

$$r'(\alpha + \Delta\alpha) \approx r'(\alpha) + r''(\alpha) \Delta\alpha = 0. \tag{16}$$

From (14), we can write the first and second derivatives of $r(\alpha)$ as:

$$r'(\alpha) = \Delta \boldsymbol{z} \cdot \boldsymbol{J}(\boldsymbol{z}_k + \alpha \Delta \boldsymbol{z}) \cdot \boldsymbol{f}(\boldsymbol{z}_k + \alpha \Delta \boldsymbol{z}), \tag{17a}$$

$$r''(\alpha) = \Delta \boldsymbol{z} \cdot \boldsymbol{J}(\boldsymbol{z}_k + \alpha \Delta \boldsymbol{z}) \cdot \boldsymbol{J}(\boldsymbol{z}_k + \alpha \Delta \boldsymbol{z}) \cdot \Delta \boldsymbol{z} + \Delta \boldsymbol{z} \cdot \boldsymbol{H}(\boldsymbol{z}_k + \alpha \Delta \boldsymbol{z}) \cdot \boldsymbol{f}(\boldsymbol{z}_k + \alpha \Delta \boldsymbol{z}) \cdot \Delta \boldsymbol{z}. \tag{17b}$$

Now, we seek to find $\Delta\alpha$ that solves the following box-constrained least squares minimization problem:

$$\min_{\Delta\alpha} \frac{1}{2}\left[r'(\alpha_j) + r''(\alpha_j)\Delta\alpha\right]^2, \tag{18a}$$

$$\epsilon - \alpha_j < \Delta\alpha < 1 - \alpha_j. \tag{18b}$$

Once $\Delta\alpha$ is found, then $\alpha_{j+1} = \alpha_j + \Delta\alpha$. The problem in (18) is solved using the `Matlab` package `lsqlin`. Usually two to four iterations are required to converge $\Delta\alpha$ to machine precision.

2.3 | Stage 3

We use the output $\boldsymbol{z}$ from the previous stage as the input in this stage. In this final stage, we use double precision for the first few iterations and then multi-precision (128 digits) to obtain $\boldsymbol{z}$ with high-precision. On referring to (7), we seek the $\Delta\boldsymbol{z}$ that solves the following constrained quadratic minimization problem:

$$\min_{\Delta z} \frac{1}{2}\|\Delta\boldsymbol{z}\|^2, \tag{19a}$$

subject to the equality constraint

$$\bar{\boldsymbol{J}}_k \Delta\boldsymbol{z} + \bar{\boldsymbol{f}}_k = \boldsymbol{0}, \tag{19b}$$

where $\alpha = 1$ is used in (7a).

Since in general $\bar{\boldsymbol{J}}_k$ is a rectangular matrix, we solve the problem posed in (19) using a constrained least square approach. The Lagrangian associated with the problem in (19) is:

$$\min_{\Delta z}\max_{\lambda} L(\Delta\boldsymbol{z}, \lambda) = \frac{1}{2}\,\Delta\boldsymbol{z}\cdot\Delta\boldsymbol{z} - \lambda\cdot\left(\bar{\boldsymbol{J}}_k\cdot\Delta\boldsymbol{z} + \bar{\boldsymbol{f}}_k\right), \tag{20}$$

where λ is the vector of Lagrange multipliers. On setting the first variation $\delta L = 0$ and using the arbitrariness of the first variations of $\Delta\boldsymbol{z}$ and λ, we obtain the following linear system of equations:

$$\bar{\boldsymbol{J}}_k\bar{\boldsymbol{J}}_k^T\lambda = -\boldsymbol{f}_k, \quad \Delta\boldsymbol{z} = \bar{\boldsymbol{J}}_k^T\lambda, \tag{21}$$

which on solving provides λ and then $\Delta\boldsymbol{z}$.

Having found $\Delta\boldsymbol{z}$, $\boldsymbol{z}_{k+1}$ is obtained using (7a), and then the process is repeated. Computations are done in double precision for the first 5 to 10 iterations, and the last 5 iterations are done in high-precision without use of the preconditioning matrix $\boldsymbol{B}_k$. The VPA (Variable-Precision Arithmetic) package in `Matlab` with 128 digits is used. Since the incremental PI rules, $\boldsymbol{C}\Delta\boldsymbol{z} < \boldsymbol{d}$, are not considered, we explicitly check if they are satisfied after each iteration. If they are not met, then we restart from the beginning by adding one more random integration point. If $\boldsymbol{C}\Delta\boldsymbol{z} < \boldsymbol{d}$ is satisfied and in addition $\|\Delta\boldsymbol{z}\| < \epsilon$ (ϵ is machine precision),

then the cubature rule is obtained. The final cubature schemes are stored with 64 decimal digits of precision. A flowchart that summarizes the main components of the cubature algorithm is presented in Fig. 1.

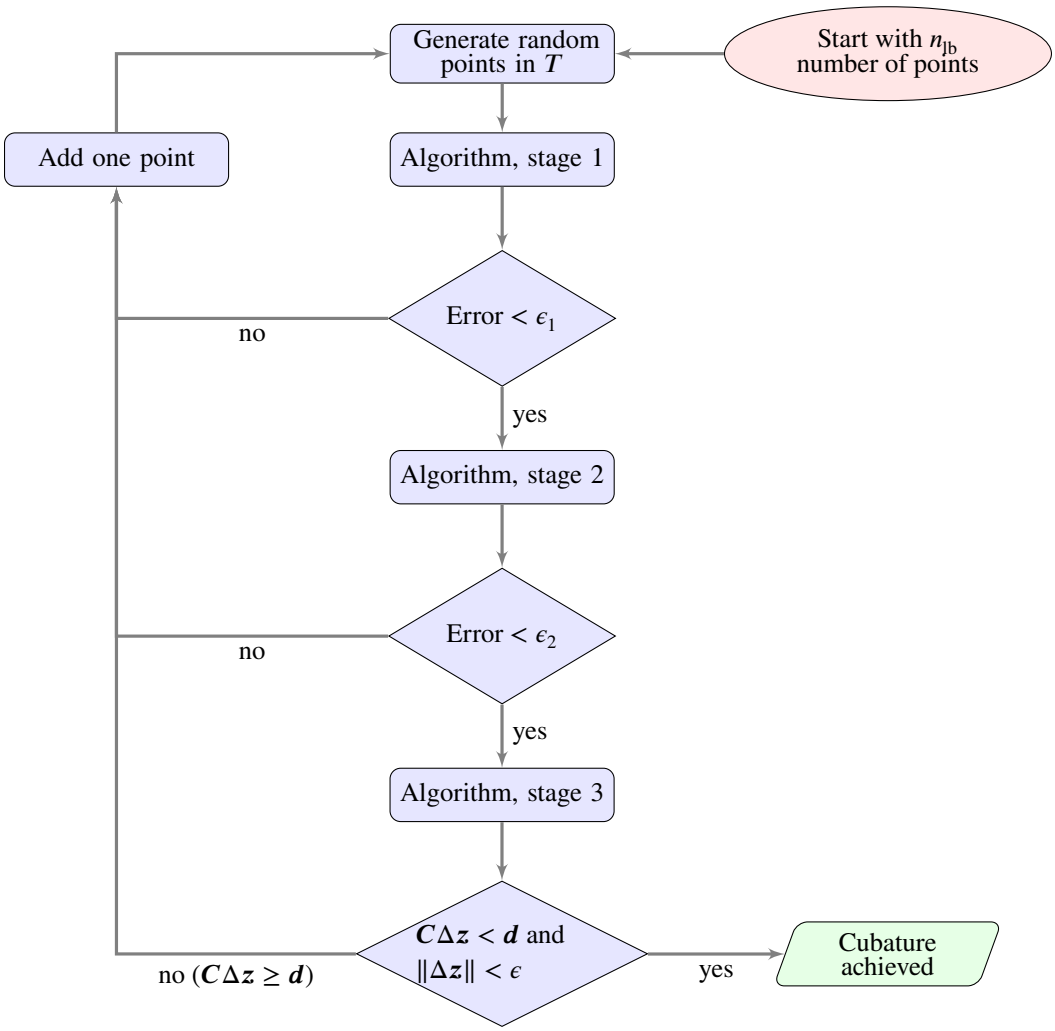

FIGURE 1 Flowchart of the cubature algorithm.

2.4 | Preconditioning

The basic Newton set of equations is constructed using the monomials as basis functions. The monomial basis functions are simple to generate and also easy to integrate over the tetrahedron using (6). On the other hand, for $p \geq 10$, the Jacobian matrix becomes ill-conditioned and then truncation errors can significantly affect the results during the iteration process, and may even cause divergence of the solution. Hence, a preconditioner is applied to the Newton equations at every iteration step.

The preconditioning matrix operator $\boldsymbol{B}_k$ at the k-th iteration step is used to modify the system of equations into the numerically stable form, see (7). The matrix $\boldsymbol{B}_k$ is updated after each iteration, since the Jacobian matrix depends on $\boldsymbol{z}_k$. The use of $\boldsymbol{B}_k$ ensures

that the nonlinear Newton equations are well-conditioned during all stages when calculations are done in double precision. The algorithm for constructing the $\boldsymbol{B}_k$ matrix is based on Gauss-Jordan elimination procedure that is performed on the selected columns of the Jacobian matrix. The algorithm starts with locating the maximum absolute value in the Jacobian matrix. Let the maximum value be located at the (i, j) position in the matrix. Then the standard Gauss elimination procedure on the entire j-th column is performed using the leading i-th row. After this, all entries in the j-th column are zero except at the (i, j) position, which is one. This procedure is repeated until ℓ columns (recall that ℓ is the number of basis functions) in the Jacobian matrix have the zero-one construction. With this transformation, the modified Jacobian matrix becomes well-conditioned.

The effectiveness of the preconditioning can be measured by checking the condition number of the Jacobian matrix before and after applying the transformation. The condition number κ for a matrix $\boldsymbol{A}$ is defined as:

$$\kappa(\boldsymbol{A}) = \|\boldsymbol{A}\|\|\boldsymbol{A}^{\dagger}\|, \tag{22}$$

where $\boldsymbol{A}^{\dagger}$ is the pseudo-inverse of $\boldsymbol{A}$. The condition number measures how sensitive the solution is to changes in the input data and to round-off errors in the computations. For a well-conditioned matrix, κ is close to one, whereas for an ill-conditioned matrix κ is large (infinity for a singular matrix).

We compute the Jacobian matrix for different p and by varying the choice of the polynomial basis. The condition number of the Jacobian matrices are listed in Table 1 for Chebyshev, Legendre, orthonormal, and monomial basis before and after applying the transformation. The orthonormal basis are generated using the classical Gram-Schmidt algorithm for up to $p = 10$. With increasing p, the condition number increases, though to a lesser extent for the orthonormal basis. It can be noticed that the condition number of the Jacobian matrix for $p = 2$ are small for all bases. The condition number of the preconditioned Jacobian using the monomial basis is the smallest and is $\mathcal{O}(1)$, while κ is significantly larger for the other basis functions. The preconditioning procedure is applied at each iteration step to ensure that the algorithm remains stable during the construction of higher order cubature rules.

3 | CUBATURE SCHEMES

The proposed cubature algorithm is implemented to generate p-order cubature schemes ($p = 2$ to $p = 20$) on a tetrahedron. The computations are done using double precision and also high-precision (128 digits). Double precision is used in the first and second stages of the algorithm as well as in the initial iterations of stage 3 as described in Section 2. High-precision calculations are applied only in the last few iterations of stage 3 of the algorithm, but without use of the preconditioning procedure.

The algorithm starts with n_{lb} randomly scattered points in the interior of the tetrahedron. For moderate p, convergence is fast, but for $p > 15$, the algorithm becomes sensitive to the location of the integration points. Thus, for higher orders the initial points

TABLE 1 Condition number κ of the Jacobian matrix for varying order of Chebyshev polynomials $\boldsymbol{J}_C$, Legendre polynomials $\boldsymbol{J}_L$, orthonormal polynomials $\boldsymbol{J}_R$, monomials $\boldsymbol{J}_m$ and monomials after transformation $\bar{\boldsymbol{J}}_m$.

p	$\boldsymbol{J}_C$	$\boldsymbol{J}_L$	$\boldsymbol{J}_R$	$\boldsymbol{J}_m$	$\bar{\boldsymbol{J}}_m$
2	713	486	23.1	422	2
5	7×10^{7}	4×10^{7}	9×10^{3}	3×10^{7}	3
8	7×10^{11}	2×10^{11}	3×10^{5}	2×10^{11}	6
10	2×10^{15}	3×10^{14}	8×10^{7}	3×10^{14}	7
15	8×10^{18}	2×10^{19}	–	2×10^{21}	7
20	4×10^{20}	1×10^{21}	–	8×10^{25}	9

TABLE 2 Number of integration points for various order of tetrahedral cubature rules. Results are compared to the lower bound estimate, $n_{\text{lb}} = \lceil (p+1)(p+2)(p+3)/24 \rceil$, and to the rules presented in Xiao and Gimbutas[2] and Witherden and Vincent.[3]

p	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
n	4	6	11	14	23	31	44	57	74	94	117	144	175	207	247	288	338	390	448
n_{lb}	3	5	9	14	21	30	41	55	72	91	114	140	170	204	242	285	333	385	443
n^2	4	6	11	14	23	31	44	57	74	95	122	146	177	214	–	–	–	–	–
n^3	4	8	–	14	24	35	46	59	81	–	–	–	–	–	–	–	–	–	–

are selected using the integration points from two previously generated lower order cubature schemes. It takes about 30 minutes to obtain the cubature rule for $p = 10$ using double-precision calculations, whereas it takes about 2 days to generate the cubature rule for $p = 20$. For the high-precision calculations using VPA (128 decimal digits) in `Matlab`, it is sufficient to perform about 5 iterations. However, these calculations are time-consuming. For example, an iteration for $p = 10$ takes 30 minutes, whereas for $p = 20$, one iteration takes about 9 hours on a multiprocessor server.

In Table 2, the number of cubature points for different p are listed using our scheme and those presented in Xiao and Gimbutas[2] and Witherden and Vincent.[3] Up to $p = 10$, the number of integration points are the same for our scheme and that of Xiao and Gimbutas.[2] However, for $p > 10$, our scheme provides fewer number of integration points than the other scheme. Furthermore, we have generated rules for $p = 16$ to $p = 20$, which are new. The reader can find the listing of the cubature points and weights in the supplementary materials, where the data are given in the column-format $x\ y\ z\ w$ with 64 digits of precision.

4 | NUMERICAL TESTS

We conduct the verification tests and assessment of the accuracy of the cubature rules in double-precision arithmetic (16 digits of precision) since use of double precision is common in scientific computing. The tests are divided into two parts. In the first part the verification of the integration schemes is assessed for a polynomial function f that is a random linear combination of polynomial basis functions:

$$f(\boldsymbol{x}) = \sum_{i=1}^{m} \Psi_i(\boldsymbol{x}) f_i, \quad I = \int_{\Omega} f(\boldsymbol{x})\, d\boldsymbol{x}, \tag{23}$$

where f_i are randomly generated coefficients and $\{\Psi_i\}_{i=1}^{m}$ are the polynomial basis functions, which are monomials, Chebyshev or Legendre functions. Any cubature rule of order p should integrate any polynomial up to order p over the domain Ω (meshed with tetrahedral elements) with machine-precision accuracy. In the second part, we first consider the integration of exponential and rational functions over the reference tetrahedron ($\Omega = T$) and the biunit cube ($\Omega = [-1, 1]^3$) that is meshed using tetrahedral elements. Then, we consider the integration of weakly singular functions over T and the integration of a trigonometric function over a semi-cylindrical domain that is meshed using tetrahedral elements with curved faces. In all these tests, the convergence and accuracy of the cubature rules are assessed.

The relative error is defined as

$$R = \frac{|I - I_q|}{|I|}, \tag{24}$$

where I is the exact integration of the function in (23), and I_q is the numerically computed value of the integral using the cubature rule. We skip cases for which the random coefficients lead to $|I|$ being very small. For comparison purposes, the same tests are carried out for the cubatures proposed in Xiao and Gimbutas[2] (up to $p = 15$) and Witherden and Vincent[3] (up to $p = 10$). The cubature rules from these papers have been taken from the 'quadpy' project.[38]

4.1 | Verification tests

Verification tests of the cubature rules are conducted on the references tetrahedron T given in (1) and the biunit cube that is meshed with tetrahedral elements.

4.1.1 | Reference tetrahedron

The reference tetrahedron is shown in Fig. 2a, and two other possible permutations of the numbering of the vertices is depicted in Figures 2b and 2c. Altogether there are 24 possible permutations of the numbering. Each permutation is connected with

an affine transformation of the integration points from the reference tetrahedron to the one with the current permutation. Any cubature rule should be invariant to such transformations, which is checked by this test.

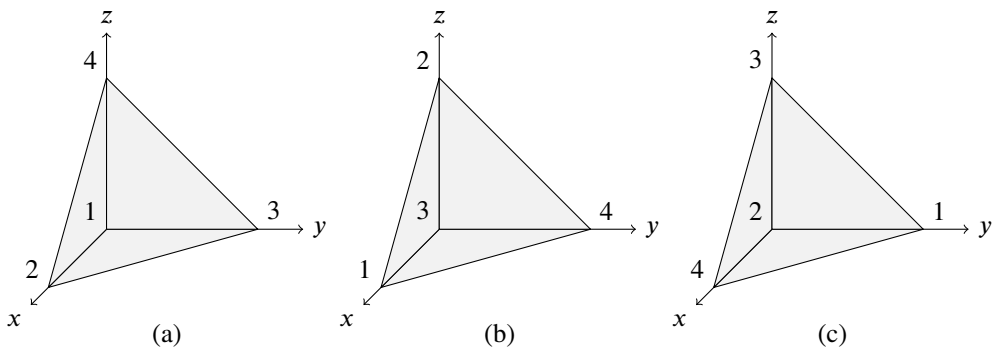

FIGURE 2 Three permutations in the numbering of the vertices for a tetrahedron used in the verification test. Reference tetrahedron is shown in (a), and two possible permutations appear in (b) and (c).

Equation (23) with p-th order monomial basis functions is used to test the p-th order cubature rule. The numerical integration of the function is performed for all 24 possible permutations of the numbering of the vertices of the tetrahedron. We performed 1000 such tests, and hence a total of 24,000 integrals are computed on the tetrahedron for each $p \in [2, 20]$. From all these results, we report the maximum and mean values of the relative errors in Table 3. The aim of this test is to check the sensitivity of the cubatures on the permutation in the numbering of the vertices of the tetrahedron. On applying the cubature rules generated in this paper as well as those obtained in Xiao and Gimbutas[2] and Witherden and Vincent[3] result in relative errors of the same order. These tests verify the accuracy of the cubature rules for integration over tetrahedra.

We repeat the same tests for Chebyshev basis functions in (23). The 3D Chebyshev basis functions are constructed using the tensor product of 1D Chebyshev functions, which are generated by the standard recurrence scheme. On the other hand, the exact integrals of the Chebyshev basis functions are obtained from their representation in terms of monomial basis functions:

$$\boldsymbol{p}_T = \boldsymbol{D}\boldsymbol{p}, \tag{25}$$

where $\boldsymbol{p}_T$ is the vector of integrated Chebyshev basis functions and $\boldsymbol{D}$ is the transformation matrix from the monomial basis functions to the Chebyshev basis functions. It is known that the transformation matrix $\boldsymbol{D}$ consists of integers, and hence the computations are performed without significant truncation errors.

The results of the test with the Chebyshev basis functions are listed in Table 4. Also in this case one thousand tests for each p are performed where in each test all 24 possible numbering of vertices are considered. It turns out that the use of Chebyshev functions leads to increase in the maximum integration error by about one order, while the mean error stays at the same level

TABLE 3 Integration errors for test on a single tetrahedron using monomials as basis functions. Comparisons are made to results obtained using the schemes presented in Xiao and Gimbutas[2] and Witherden and Vincent.[3] The maximum and mean values of the error from thousand permutation tests for each p are listed.

p	This work			Xiao and Gimbutas[2]			Witherden and Vincent[3]		
		$R \times 10^{16}$			$R \times 10^{16}$			$R \times 10^{16}$	
	n	max	mean	n	max	mean	n	max	mean
2	4	9	2	4	13	2	4	9	2
3	6	18	3	6	20	3	8	15	3
4	11	47	4	11	44	4	–	–	–
5	14	40	5	14	41	5	14	40	6
6	23	62	6	23	67	8	24	45	6
7	31	117	8	31	119	10	35	106	8
8	44	127	10	44	133	12	46	122	10
9	57	143	11	57	145	11	59	117	11
10	74	164	14	74	166	14	81	150	14
11	94	178	17	95	191	17	–	–	–
12	117	331	19	122	329	19	–	–	–
13	144	365	23	146	361	24	–	–	–
14	175	344	23	177	335	23	–	–	–
15	207	365	27	214	365	28	–	–	–
16	247	598	35	–	–	–	–	–	–
17	288	307	18	–	–	–	–	–	–
18	338	239	25	–	–	–	–	–	–
19	392	497	21	–	–	–	–	–	–
20	448	494	25	–	–	–	–	–	–

as for the monomial basis. We can infer that using Chebyshev basis functions does not significantly degrade the accuracy of the numerical integration.

TABLE 4 Integration errors for test on a single tetrahedron using Chebyshev basis functions. Comparisons are made to results obtained using the schemes presented in Xiao and Gimbutas[2] and Witherden and Vincent.[3] The maximum and mean values of the error from thousand permutation tests for each p are listed.

p	This work			Xiao and Gimbutas[2]			Witherden and Vincent[3]		
		$R \times 10^{16}$			$R \times 10^{16}$			$R \times 10^{16}$	
	n	max	mean	n	max	mean	n	max	mean
2	4	18	2	4	19	2	4	15	2
3	6	42	3	6	38	4	6	36	3
4	11	50	4	11	54	4	–	–	–
5	14	102	6	14	111	7	14	87	7
6	23	144	8	23	175	10	23	156	9
7	31	267	10	31	336	12	31	249	10
8	44	333	13	44	360	16	44	258	13
9	57	499	16	57	583	18	57	402	17
10	74	450	17	74	470	18	74	463	17
11	94	699	24	95	738	25	–	–	–
12	117	765	21	122	814	22	–	–	–
13	144	1076	29	146	1345	32	–	–	–
14	175	1158	30	177	1259	33	–	–	–
15	207	1430	32	214	1396	36	–	–	–
16	247	2481	44	–	–	–	–	–	–
17	288	593	11	–	–	–	–	–	–
18	338	792	13	–	–	–	–	–	–
19	392	939	18	–	–	–	–	–	–
20	448	975	15	–	–	–	–	–	–

4.1.2 | Biunit cube

In this test, for each p, 100 random polynomials are generated and then integrated over the biunit cube, $[-1, 1]^3$. The cube is meshed by 162 nonuniform tetrahedra (see Fig. 3). The basis functions are randomly chosen to be complete monomials, Chebyshev or Legendre polynomials.

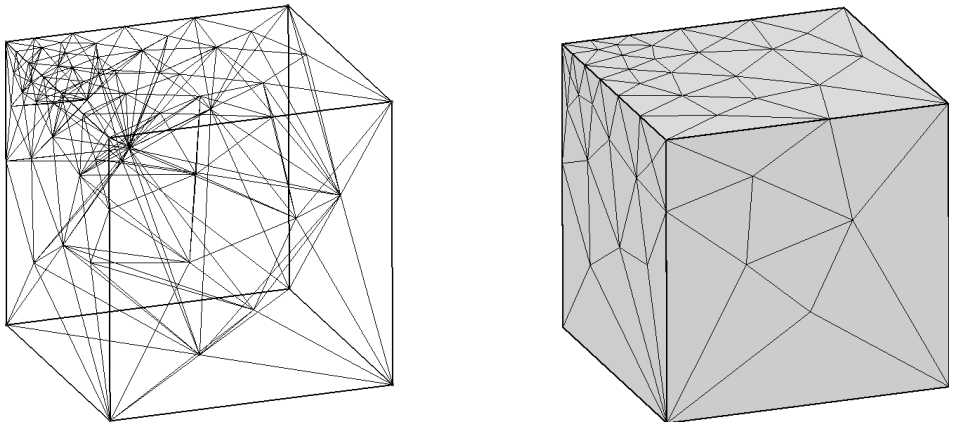

FIGURE 3 Tetrahedral mesh of the biunit cube, $\Omega = [-1, 1]^3$, with 162 nonuniform tetrahedra of various sizes. The mesh vertices (left) and element faces on the boundary surfaces (right) are shown.

In the tests, we choose the order of $f(\boldsymbol{x})$ to be the same as the order of the cubature to be checked. The exact integral of f over the cube is readily computed. For the numerical value of the integral, we use the cubature rule on each tetrahedron of the mesh and sum the results. The results of the numerical tests are presented in Table 5. In the table, the maximum as well as the algebraic mean values of the relative integration errors from a set of 100 tests for each p are presented. It can be noticed in Table 5 that the errors for the integration schemes of order $p \leq 15$ generated from this study and those from Xiao and Gimbutas[2] and Witherden and Vincent[3] are of the same magnitude and oscillate about machine precision. For all the integration schemes, the maximum error is $\mathcal{O}(10^{-14})$, but their mean values are $\mathcal{O}(10^{-15})$. For $p \geq 16$, which are the new rules from this work, the relative errors are also close to machine precision.

4.2 | Convergence tests

We assess the accuracy of the p-order cubature rules for the integration of an exponential function, a rational function, weakly singular functions and a trigonometric function over tetrahedral elements. In the last example, the tetrahedra have curved faces.

4.2.1 | Exponential test function

In this test, we consider the exponential function

$$f(\boldsymbol{x}) = \exp(\alpha x + \beta y + \gamma z), \tag{26}$$

where α, β and γ are real numbers. For this test, we set $\alpha = 9$, $\beta = 12$, and $\gamma = 4$. When $\alpha \neq \beta \neq \gamma$, the integral of the test function over the reference tetrahedron is given by

$$I = \int_T f(\boldsymbol{x})\, d\boldsymbol{x} = \frac{\exp(\alpha) - 1}{\alpha\beta\gamma} - \frac{\exp(\alpha) - \exp(\gamma)}{\gamma(\alpha - \gamma)(\beta - \gamma)} - \frac{\exp(\alpha) - \exp(\beta)}{\beta\gamma(\alpha - \beta)} - \frac{\exp(\alpha) - \exp(\beta)}{\gamma(\alpha - \beta)(\beta - \gamma)}, \tag{27}$$

TABLE 5 Integration errors for test on biunit cube with tetrahedral mesh, Fig. 3. Comparisons are made to results obtained using the schemes presented in Xiao and Gimbutas[2] and Witherden and Vincent.[3] The listing is the maximum and mean values of the error from hundred tests for each p.

p	This work			Xiao and Gimbutas[2]			Witherden and Vincent[3]		
		$R \times 10^{16}$			$R \times 10^{16}$			$R \times 10^{16}$	
	n	max	mean	n	max	mean	n	max	mean
2	4	68	5.4	4	43	6	4	74	5
3	6	109	5	6	146	7	8	183	7
4	11	36	6	11	54	7	–	–	–
5	14	237	10	14	171	10	14	65	7
6	23	120	10	23	119	10	24	150	9
7	31	314	17	31	259	18	35	38	6
8	44	202	10	44	184	11	46	200	10
9	57	167	12	57	260	14	59	570	17
10	74	553	22	74	685	23	81	104	11
11	94	195	14	95	99	13	–	–	–
12	117	76	11	122	96	12	–	–	–
13	144	281	19	146	343	21	–	–	–
14	175	328	20	177	332	22	–	–	–
15	207	399	28	214	251	25	–	–	–
16	247	467	30	–	–	–	–	–	–
17	288	190	17	–	–	–	–	–	–
18	338	668	38	–	–	–	–	–	–
19	390	852	42	–	–	–	–	–	–
20	448	662	30	–	–	–	–	–	–

and for the chosen values of α, β, and γ, we have $I = 5.054368832531350 \times 10^2$.

The numerical tests for the integration of the exponential function on the reference tetrahedron are the same as those performed in Section 4.1.1. Our cubature schemes and those of Xiao and Gimbutas[2] are used in the study. From all the permutations of the numbering of the vertices, the maximum error for each p is selected and used in the convergence plots shown in Fig. 4. Plots

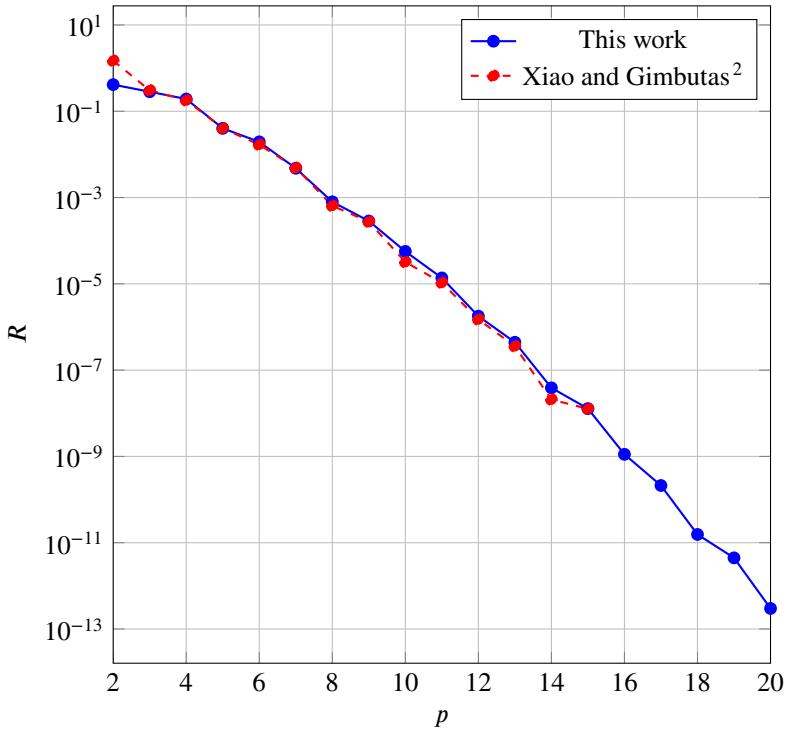

FIGURE 4 Relative error in the integration of the exponential function in (26) ($\alpha = 9$, $\beta = 12$, $\gamma = 4$) on the reference tetrahedron.

are shown for the relative integration error R versus p. For $p \leq 15$ the convergence rates and accuracy of both cubature schemes are similar. For $p > 15$, our cubature schemes continue to show monotonic convergence and attain an accuracy of $\mathcal{O}(10^{-13})$ for $p = 20$.

The exponential function in (26) is also used to integrate over the biunit cube. The exact integral of the exponential function in $C = [-1,\ 1]^3$ is:

$$I = \int_C f(\boldsymbol{x})\, d\boldsymbol{x} = \frac{1}{\alpha\beta\gamma}\big(\exp(\alpha) - \exp(-\alpha)\big)\big(\exp(\beta) - \exp(-\beta)\big)\big(\exp(\gamma) - \exp(-\gamma)\big). \tag{28}$$

For this test, we choose $\alpha = 15$, $\beta = 12$, and $\gamma = 14$, and the exact value of the integral is $I = 2.539061482164276 \times 10^{14}$.

For the purpose of numerical integration, the cube is meshed by 1466 tetrahedra of similar sizes. We compare the accuracy of the cubature rules generated in this paper to those by Xiao and Gimbutas.[2] In Fig. 5, convergence curves of the relative integration error of both schemes are shown as a function of p. We observe that the integration schemes of Xiao and Gimbutas[2] converge monotonically, while the convergence of our scheme is nonmonotonic; however, for most p, our cubature schemes have better accuracy. For $p = 20$, our scheme attains machine-precision accuracy.

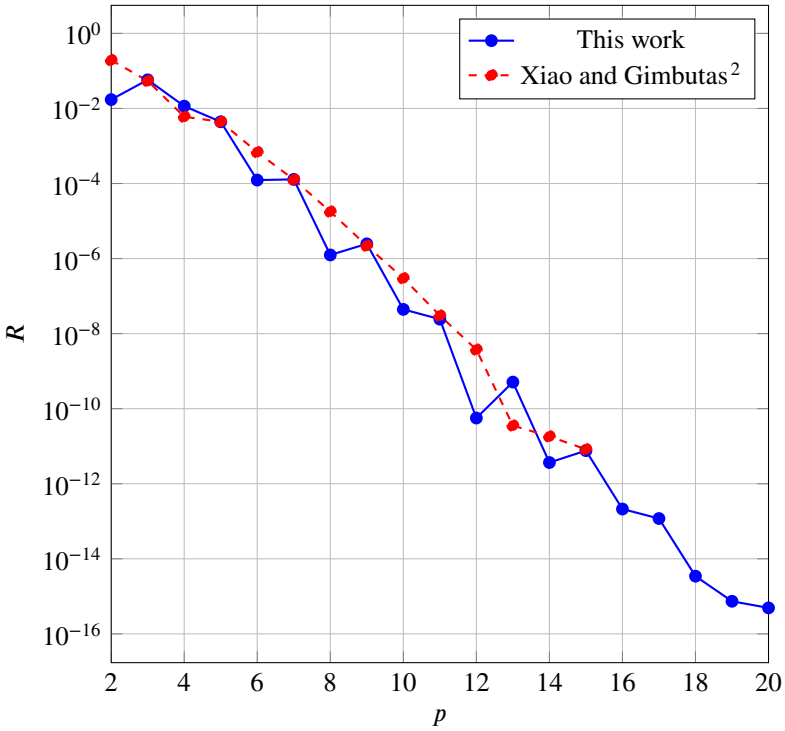

FIGURE 5 Relative error in the integration of the exponential function in (26) ($\alpha = 15$, $\beta = 12$, $\gamma = 14$) over the biunit cube, which is meshed with 1466 tetrahedra.

4.2.2 | Rational test function

In this test, we consider the rational function

$$f(\boldsymbol{x}) = \frac{g_5(\boldsymbol{x})}{1 + \left[g_3(\boldsymbol{x})\right]^2}, \tag{29}$$

where $g_k(\boldsymbol{x})$ is a randomly generated polynomial of order k. The cubature rules are used to compute the integral of $f(\boldsymbol{x})$ over the reference tetrahedron. A reference value of the integral is obtained by meshing the reference tetrahedron with 1156 tetrahedra and performing numerical integration on each tetrahedron and summing the results. On each tetrahedron of the mesh, our cubature schemes from $p = 13$ to $p = 20$ and those of Xiao and Gimbutas[2] ($p = 13,\ 14,\ 15$) yield the same reference result.

The cubature schemes from this work are tested on the reference tetrahedron with permutations on the numbering of the vertices. The tetrahedral cubatures for $p = 2$ up to $p = 20$ are used for the numerical integration of the rational test function in (29). For each permutation, the integration errors fluctuate and the maximum error for each p is chosen. In Fig. 6, the relative integration error R versus p is presented for our scheme and those obtained using the cubature scheme of Xiao and Gimbutas.[2] It can be seen that for $p \leq 15$, the accuracy of both schemes is proximal and convergence is monotonic. For $p > 15$, the convergence of our cubature schemes retain monotonicity, and an accuracy of $\mathcal{O}(10^{-7})$ is attained for $p = 20$.

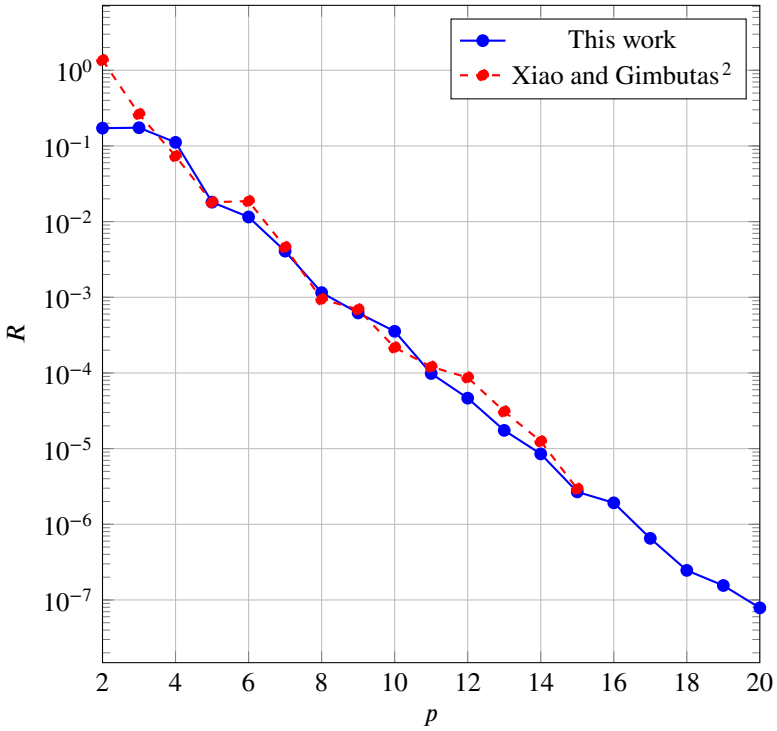

FIGURE 6 Relative error in the integration of the rational function in (29) over the reference tetrahedron.

4.2.3 | Weakly singular test functions

In this test, we consider the weakly singular functions

$$f_1(\boldsymbol{x}) = \frac{1}{\sqrt{r}}, \quad f_2(\boldsymbol{x}) = \frac{1}{r}, \quad r = \sqrt{x^2 + y^2 + z^2}. \tag{30}$$

These functions are unbounded at the origin, but have integrable singularities over the reference tetrahedron T. The exact value of the integral of $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$ over T are $I = 163/679$ and $I = 1531/4236$, respectively.

For this example, we conduct studies that are similar to those performed in Sections 4.2.1 and 4.2.2. The results are shown in Fig. 7. We observe that the accuracy improves for increasing p, but the convergence curve is nonmonotonic using our cubature rules as well as for the rules from Xiao and Gimbutas.[2] In most instances, our cubature rules give slightly better results that those obtained using the cubature rules of Xiao and Gimbutas.[2] However, even for $p = 20$ the accuracy is limited and tends to level-off. These accuracy trends are similar to what is obtained in the integration of $1/\sqrt{r}$ and $1/r$ on a square using tensor-product Gauss quadrature.[36] Most of the error resides in the vicinity of the vertex located at the origin, and hence more cubature points are needed in this region. Singularity-canceling schemes such as the Duffy transformation or the generalized Duffy transformation and also the homogeneous numerical integration method[36] are more accurate and efficient approaches for evaluating such integrals.

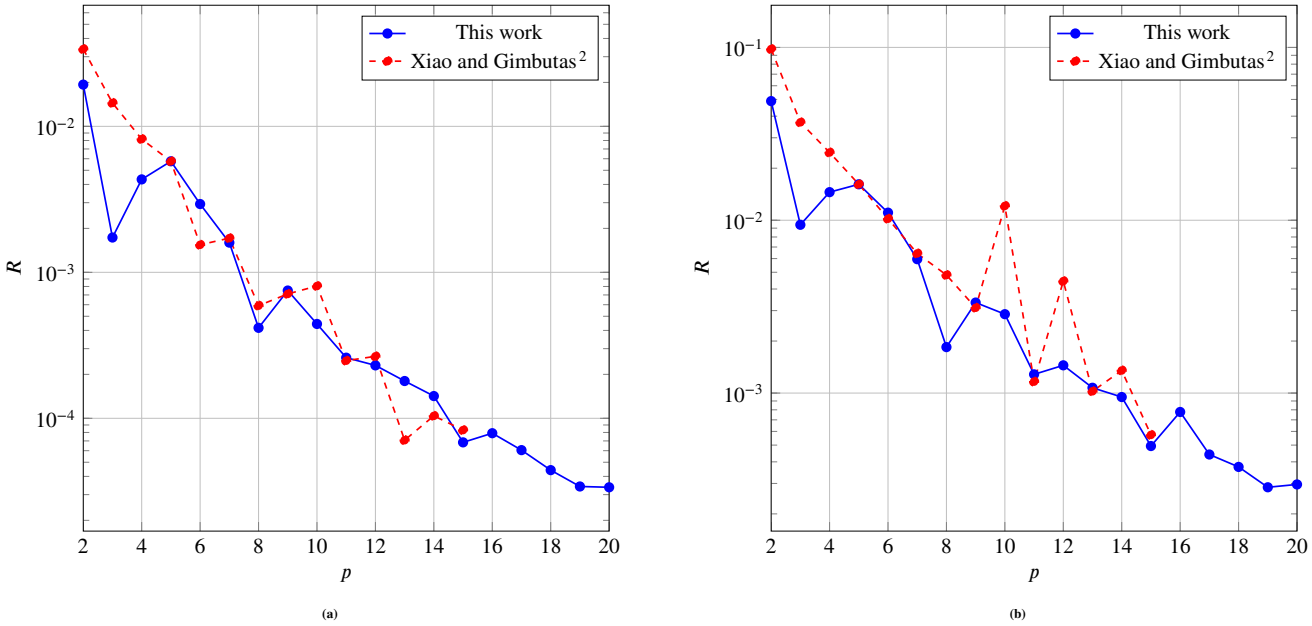

FIGURE 7 Relative error in the integration of weakly singular functions over the reference tetrahedron. (a) $f_1(\boldsymbol{x})$ and (b) $f_2(\boldsymbol{x})$ are given in (30).

4.2.4 | Integration over a semi-cylindrical domain

In this example, we consider a semi-cylindrical domain Ω of height 2, inner radius $R_1 = \sqrt{\pi/2}$ and outer radius $R_2 = \sqrt{3\pi}$ that is shown in Fig. 8. We integrate the test function $f(x, y) = \sin(x^2 + y^2)$ over Ω. The exact value of this integral is:

$$I = \int_{\Omega} \sin(x^2 + y^2)\, d\boldsymbol{x} = \int_{-1}^{1} \int_{0}^{\pi} \int_{R_1}^{R_2} r \sin(r^2)\, dr d\theta dz = -\pi \cos(r^2)\Big|_{R_1}^{R_2} = \pi. \tag{31}$$

In order to perform the numerical integration over Ω, the biunit cube is meshed using 12 tetrahedra. Then the following geometric map of the biunit cube to the semi-cylindrical domain (see Fig. 8) is used:

$$\begin{aligned} x &= R(\eta) \sin\left[\frac{\pi}{2}(1+\xi)\right], \\ y &= R(\eta) \cos\left[\frac{\pi}{2}(1+\xi)\right], \\ z &= \zeta, \\ R(\eta) &= R_1 \frac{1-\eta}{2} + R_2 \frac{1+\eta}{2}. \end{aligned} \tag{32}$$

Due to the geometric map, the domain Ω is covered by curved tetrahedra. Numerical integration is performed using the cubature schemes proposed in this work and those from Xiao and Gimbutas [2]. The relative error in the numerical integration as a function of the order p is presented in Fig. 9. Up to $p = 15$, both schemes yield similar results but with nonmonotonic convergence. For $p > 15$, the cubature rules proposed in this work continue to show improved accuracy and convergence towards the exact value.

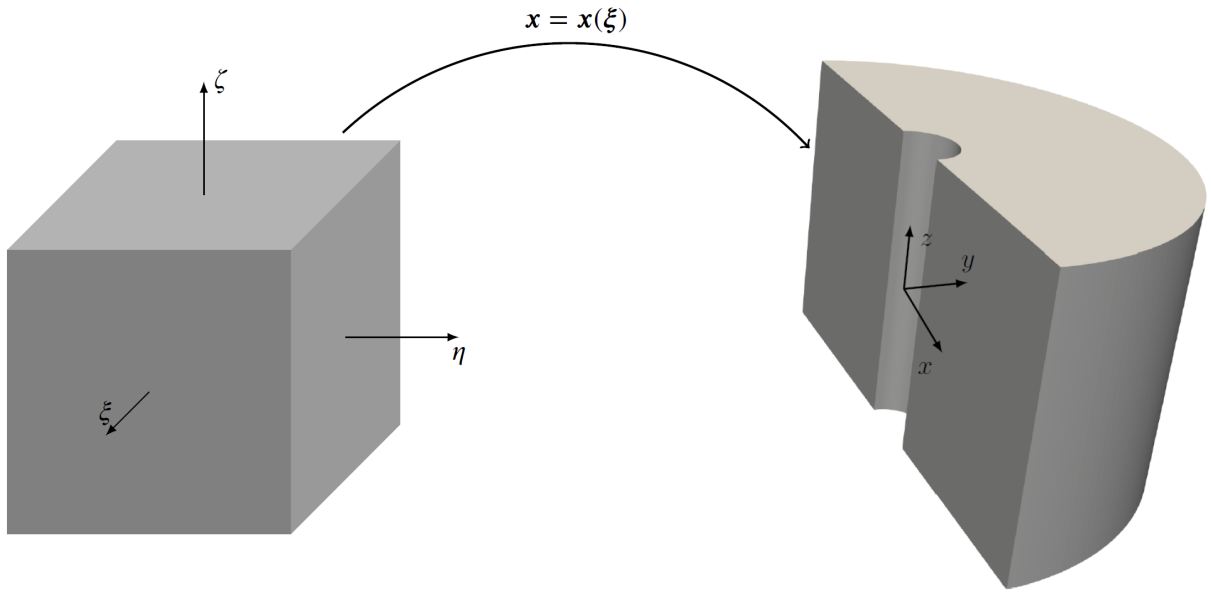

FIGURE 8 Geometric map from a binunit cube to a semi-cylindrical domain of height $h = 2$, inner radius $R_1 = \sqrt{\pi/2}$ and outer radius $R_2 = \sqrt{3\pi}$.

5 | CONCLUSIONS

In this paper, we have presented a new algorithm to generate high-order cubature rules for tetrahedra. To construct a cubature scheme, the nonlinear moment equations were solved with positive weights and with integration points (nodes) in the interior of the tetrahedron (so-called 'PI rules'). The cubature algorithm consisted of three stages, with each a modified Newton procedure. In the first stage, we began the search for a cubature rule by selecting random integration points inside the tetrahedron (used the lower bound estimate to set the number of inserted points). The search direction in the Newton update was found by minimizing the square of the norm of the Newton equations subject to inequality constraints that result from the PI rules. A fixed step size was used in the update of the solution vector. In the second stage, a quadratic minimization problem with equality and inequality constraints was considered using step size control to accelerate convergence. Finally, in the third stage a full Newton method (unit step size) was used in conjunction with an equality constrained quadratic least squares problem that provided the search direction as the solution of a linear system of equations. In all stages of the algorithm that involved calculations in double precision, a preconditioned matrix (via Gauss-Jordan elimination) was used at every iteration step, which ensured that the Jacobian matrix was well-conditioned. The initial iterations in the third stage were carried out in double precision, and the last few iterations were conducted using Variable Precision Arithmetic (128 digits) in `Matlab`, which provided cubature rules with high-precision. Cubature schemes of order $p = 2$ to $p = 20$ were constructed with the same or fewer number of integration points than existing rules. In the literature only rules up to $p = 15$ are available; the rules for $p = 16$ to $p = 20$ that were generated were new.

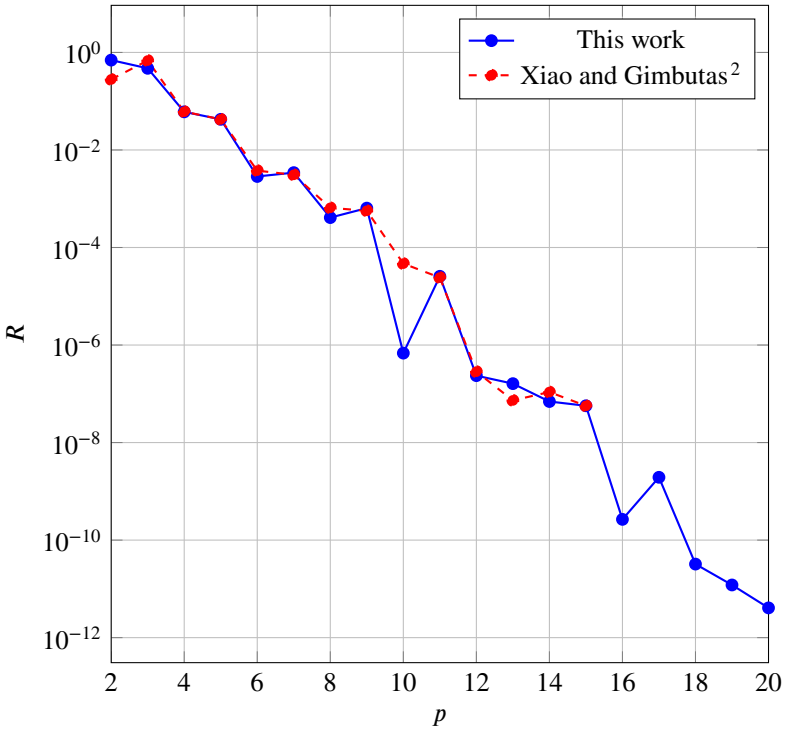

FIGURE 9 Relative error in the integration of the function $\sin(x^2 + y^2)$ over the semi-cylindrical domain.

Verification tests and accuracy assessment of the cubature rules (truncated to 64 decimal digits) were conducted using double-precision calculations. The domains of integration were the reference tetrahedron $T = \{(x, y, z) : 0 \leq x \leq 1, 0 \leq x + y \leq 1, 0 \leq x + y + z \leq 1\}$, a tetrahedral mesh-partition of the biunit cube $C = [-1, 1]^3$, and a tetrahedral (curved elements) mesh-partition of a semi-cylindrical domain. The p-order rules successfully passed the verification tests with monotonic convergence and near machine-precision accuracy for the integration of p-order monomial and Chebyshev (polynomial) basis functions over T and C. The sound accuracy of the cubature schemes was also demonstrated for the integration of an exponential test function over T and C, rational and weakly singular test functions over T, and a trigonometric function over a semi-cylindrical domain. For some of the test examples, we observed nonmonotonic convergence. To better understand the performance of the cubature scheme on different integrands, developing error estimators for the numerical integration scheme would be valuable. As part of future work, we plan to construct polynomial-precise cubature rules over other 3D domains such as pyramids, prisms and cubes.

ACKNOWLEDGEMENTS

The first author gratefully acknowledges the support provided by the Polish National Agency for Academic Exchange via scholarship in Bekker programme (PPN/BEK/2018/1/00125) at the University of California at Davis. The second author is grateful for the research support of Sandia National Laboratories to the University of California at Davis.

References

1. Wandzura S, Xiao H. Symmetric quadrature rules on a triangle. *Computers & Mathematics with Applications* 2003; 45(12): 1829–1840.

2. Xiao H, Gimbutas Z. A numerical algorithm for the construction of efficient quadrature rules in two and higher dimensions. *Computers & Mathematics with Applications* 2010; 59(2): 663–676.

3. Witherden F, Vincent P. On the identification of symmetric quadrature rules for finite element methods. *Computers & Mathematics with Applications* 2015; 69(10): 1232–1241.

4. Papanicolopulos SA. Computation of moderate-degree fully-symmetric cubature rules on the triangle using symmetric polynomials and algebraic solving. *Computers & Mathematics with Applications* 2015; 69(7): 650–666.

5. Silvester P. Symmetric quadrature formulae for simplexes. *Mathematics of Computation* 1970; 24(109): 95–100.

6. Cowper GR. Gaussian quadrature formulas for triangles. *International Journal for Numerical Methods in Engineering* 1973; 7(3): 405–408.

7. Lyness JN, Jespersen D. Moderate degree symmetric quadrature rules for the triangle. *IMA Journal of Applied Mathematics* 1975; 15(1): 19–32.

8. Hillion P. Numerical integration on a triangle. *International Journal for Numerical Methods in Engineering* 1977; 11(5): 797–815.

9. Laursen ME, Gellert M. Some criteria for numerically integrated matrices and quadrature formulas for triangles. *International Journal for Numerical Methods in Engineering* 1978; 12(1): 67–76.

10. Dunavant DA. High degree efficient symmetrical Gaussian quadrature rules for the triangle. *International Journal for Numerical Methods in Engineering* 1985; 21(6): 1129–1148.

11. Berntsen J, Espelid TO. Degree 13 symmetric quadrature rules for the triangle. Technical Report, Department of Informatics, University of Bergen; Norway: 1990.

12. Rathod HT, Nagaraja KV, Venkatesudu B. Symmetric Gauss Legendre quadrature formulas for composite numerical integration over a triangular surface. *Applied Mathematics and Computation* 2007; 188(1): 865–876.

13. Papanicolopulos SA. New fully symmetric and rotationally symmetric cubature rules on the triangle using minimal orthonormal bases. *Journal of Computational and Applied Mathematics* 2016; 294: 39–48.

14. Mousavi SE, Xiao H, Sukumar N. Generalized Gaussian quadrature rules on arbitrary polygons. *International Journal for Numerical Methods in Engineering* 2010; 82(1): 99-113.

15. Bucero MA, Bajaj C, Mourrain B. On the construction of general cubature formula by flat extensions. *Linear Algebra and its Applications* 2016; 502: 104–125.

16. Wachspress E. *Rational Bases and Generalized Barycentrics*. Cham: Springer . 2016.

17. Hillion P. Numerical integration on a tetrahedron. *Calcolo* 1981; 18(2): 117–130.

18. Yu J. Symmetric Gaussian quadrature formulae for tetrahedronal regions. *Computer Methods in Applied Mechanics and Engineering* 1984; 43(3): 349–353.

19. Keast P. Moderate-degree tetrahedral quadrature formulas. *Computer Methods in Applied Mechanics and Engineering* 1986; 55(3): 339–348.

20. Walkington N. Quadrature on simplices of arbitrary dimension. Research Report, Department of Mathematical Sciences, Center for Nonlinear Analysis, Carnegie Mellon University; 5000 Forbes Avenue, Pittsburgh, PA 15213: 2000.

21. Felippa CA. A compendium of FEM integration formulas for symbolic work. *Engineering Computations* 2004; 21(8): 867–890.

22. Cools R, Kim K. Rotation invariant cubature formulas over the *n*-dimensional unit cube. *Journal of Computational and Applied Mathematics* 2001; 132(1): 15–32.

23. Kubatko EJ, Yeager BA, Maggi AL. New computationally efficient quadrature formulas for triangular prism elements. *Computers & Fluids* 2013; 73: 187–201.

24. Kubatko EJ, Yeager BA, Maggi AL. New computationally efficient quadrature formulas for pyramid elements. *Finite Elements in Analysis and Design* 2013; 65: 63–75.

25. Cools R, Rabinowitz P. Monomial cubature rules since "Stroud": a compilation. *Journal of Computational and Applied Mathematics* 1993; 48(3): 309–326.

26. Cools R. Monomial cubature rules since "Stroud": a compilation — part 2. *Journal of Computational and Applied Mathematics* 1999; 112(3): 21–27.

27. Šolín P, Segeth K, Doležel I. *Higher-Order Finite Element Methods*. Studies in Advanced MathematicsBoca Raton, FL: Chapman & Hall, CRC Press . 2004.

28. Yosibash Z. *p*-FEMs in biomechanics: Bones and arteries. *Computer Methods in Applied Mechanics and Engineering* 2012; 249–252: 169–184.

29. Yacobi I. High order hierarchical tetrahedral finite elements. Master's thesis. Ben-Gurion University of the Negev. Beer-Sheva, Israel: 2018.

30. Schneider T, Hu Y, Dumas J, Gao X, Panozzo D, Zorin D. Decoupling simulation accuracy from mesh quality. *ACM Transactions on Graphics* 2018; 37(6): 280.

31. Chan J, Hewett RJ, Warburton T. Weight-adjusted discontinuous Galerkin methods: Curvilinear meshes. *SIAM Journal on Scientific Computing* 2017; 39(6): A2395–A2421.

32. Grundmann A, Möller H. Invariant integration formulas for the *n*-simplex by combinatorial methods. *SIAM Journal on Numerical Analysis* 1978; 15(2): 282–290.

33. Zhang L, Cui T, Liu H. A set of symmetric quadrature rules on triangles and tetrahedra. *Journal of Computational Mathematics* 2009; 27(1): 89–96.

34. Shunn L, Ham F. Symmetric quadrature rules for tetrahedra based on a cubic close-packed lattice arrangement. *Journal of Computational and Applied Mathematics* 2012; 236(17): 4348–4364.

35. Williams DM, Shunn L, Jameson A. Symmetric quadrature rules for simplexes based on sphere close packed lattice arrangements. *Journal of Computational and Applied Mathematics* 2014; 266: 18–38.

36. Chin EB, Lasserre JB, Sukumar N. Numerical integration of homogeneous functions on convex and nonconvex polygons and polyhedra. *Computational Mechanics* 2015; 56(6): 967–981.

37. Coleman T, Li Y. A reflective Newton method for minimizing a quadratic function subject to bounds on some of the variables. *SIAM Journal on Optimization* 1996; 6(4): 1040–1058.

38. Schlömer N. Quadpy. Available at https://pypi.org/project/quadpy/; 2019 (Accessed on July 14, 2019).