# An efficient method to integrate polynomials over polytopes and curved solids

Eric B. Chin[a], N. Sukumar[b,*]

[a]Lawrence Livermore National Laboratory, 7000 East Avenue, Livermore, CA 94550, USA
[b]Department of Civil and Environmental Engineering, University of California, Davis, CA 95616, USA

## Abstract

In this paper, we present an efficient approach to compute the integral of monomials and polynomials over polyhedra and regions defined by parametric curved boundary surfaces. We use Euler's theorem for homogeneous functions in combination with Stokes's theorem to reduce the integration of a monomial over a three-dimensional solid to its boundary. If the solid is a polytope, through a recursive application of these theorems, the integral is further reduced to just the evaluation of the monomial and its derivatives at the vertices of the polytope. The present approach is simpler than existing techniques that rely on repeated use of the divergence theorem, which require the antiderivative of the monomials and the projection of these functions onto hyperplanes. For convex and nonconvex polytopes, our approach does not introduce any approximation for the integration of monomials. For curved solid regions bounded by surfaces that admit a parameterization, the same approach yields simplified formulas to compute the integral of any homogeneous function, including monomials. For surfaces parameterized by polynomial surfaces (such as Bézier surface triangles and B-spline patches), the method yields machine-precision accuracy for the volumetric integration of monomials with an appropriate quadrature rule. Numerical examples over regions bounded by polynomial surfaces and rational surfaces are presented to establish the accuracy and efficiency of the method.

*Keywords:* Euler's homogeneous function theorem, polyhedral mass properties, inertia tensor, nonconvex polyhedra, Bézier triangles, NURBS surfaces

## 1. Introduction

Accurate and efficient integration of polynomials over arbitrarily-shaped polygons and polyhedra is required for the solution of partial differential equations (PDEs) using emerging computational methods such as extended finite elements (Fries and Belytschko, 2010; Moës et al., 1999; Sukumar et al., 2015), embedded interface and fictitious domain methods (de Almeida and Wall, 2014; Schillinger and Reuss, 2015; Sudhakar and Wall, 2013), mimetic finite-differences (Beirão da

*Corresponding author
*Email address:* nsukumar@ucdavis.edu (N. Sukumar)

Veiga et al., 2014), virtual element method (Beirão da Veiga et al., 2013), and discontinuous Galerkin method (Bassi et al., 2012; Cangiani et al., 2014). Integration of polynomials on polyhedra is also a key component of rigid-body dynamics simulations in computer graphics (Guendelman et al., 2003), where the inertia tensor (polyhedral mass properties) requires integration of quadratic monomials. Furthermore, for higher-order approximations using the extended finite element method (Chin and Sukumar, 2019) (or embedded interface-type methods, more generally) and for applications in computer-aided manufacturing, physics-based animations, and computer-aided design (CAD) (Krishnamurthy and McMains, 2011), accurate integration of polynomials over curved regions (such as those bounded by parametric surfaces in the Lagrange or Bernstein basis) becomes a necessity.

For integrating polynomials over arbitrary polytopes, three general approaches have been employed (see Sudhakar and Wall (2013)): (i) tessellation of the domain into simplices; (ii) application of generalized Stokes's (Gauss's divergence) theorem to reduce the volume integral to a surface integral; and (iii) use of moment-fitting methods. Tessellation requires partitioning the domain into smaller subdomains (usually simplices) and then performing numerical integration over the subdomains. This requires local remeshing algorithms, complicates programming, and increases computational costs since many integration points are needed. The divergence theorem converts integration over the domain into integration over the boundary of the domain (Mirtich, 1996), but requires construction of a local coordinate system that increases the degree and number of terms that appear on each lower-dimensional facet. Moment-fitting methods, which solve a linear system of equations to build a cubature rule over the domain to integrate polynomial basis functions, are specific to each polytope, and hence are not attractive for use in the aforementioned computational methods. Exact integration of polynomials over polyhedra using powers of linear forms is now also available (Baldoni et al., 2014; De Loera et al., 2013), but the approach is restricted to convex polytopes and is targeted towards very high dimensions.

Several methods have been proposed for integrating polynomial functions over regions bounded by curved surfaces. Some of these methods include: (i) planar approximation of curved boundaries (Messner and Taylor, 1980; Min and Gibou, 2007), (ii) interface approximation using octree-based methods (Düster et al., 2008; Lee and Requicha, 1982), (iii) using generalized Stokes's (Gauss's divergence) theorem to reduce the volume integral to lower dimensional integrals (Al-Daccak and Angeles, 1993; Krishnamurthy and McMains, 2011; Timmer and Stern, 1980), and (iv) subdividing the region into topologically simpler shapes and then approximating curves on the boundary (Legay et al., 2005). In general, these methods, except for those based on Stokes's theorem, require approximation of the curved boundaries, which introduce error in the integrals. The Stokes's theorem-based approaches are exact, but require selecting an antiderivative of the integrand, which is nonunique. Furthermore, the antiderivative must be carefully selected, such that the face normal and the projection direction are not nearly orthogonal.

In this paper, we present an improved solution for integration over polyhedra and regions bounded by curved surfaces by appealing to Euler's theorem for homogeneous functions in combination with Stokes's theorem. For polytopes, this idea was first introduced by Lasserre (1998). However, it was not until recently (Chin et al., 2015) that the applicability of the method to nonconvex polytopes was made explicit. Since then, this integration method has been successfully applied to solve problems in the discontinuous Galerkin method on polytopal meshes (Antonietti et al., 2018; Lipnikov and Morgan, 2019), extended finite element method (Chin et al., 2017; Chin and Sukumar, 2019), and the virtual element method (Benvenuti et al., 2019). For integration of homogeneous functions over curved regions, Lasserre (1999) considered regions bounded by curves that are defined by homogeneous functions, and Chin and Sukumar (2019) extended the method to two-dimensional regions bounded by parametric curves. This paper extends the integration method further, by introducing fast algorithms to integrate monomials over arbitrary polyhedra and by developing formulas to exactly integrate homogeneous functions over three-dimensional regions bounded by parametric surfaces, such as Bézier triangles and NURBS patches.

The remainder of this paper is organized as follows. In Section 2, the integration method is introduced. Euler's homogeneous function theorem and generalized Stokes's theorem are combined to produce an integration formula that reduces the dimension of integration, and further reduces integration to the vertices for monomial integrands over polytopes. Section 3 applies the integration method to arbitrary convex and nonconvex polyhedra for the purposes of integrating monomials. Simplified formulas are introduced for computing integrals required for the inertia tensor and numerical examples are presented. Section 4 establishes that the integration method is also applicable to regions bounded by parametrically-defined curved surfaces, such as Bézier surface triangles and non-uniform rational B-spline (NURBS) patches. Several numerical examples are presented to demonstrate the capabilities of the integration method to accurately and efficiently integrate monomials. Finally, in Section 5 we summarize our main findings and provide a few directions for future research related to the integration method.

## 2. Homogeneous numerical integration method

The integration method used in Chin et al. (2017) is coined the homogeneous numerical integration (HNI) method and, for brevity, we will refer to it as such in this paper. In this section, the HNI method will be summarized for arbitrary regions, and several simplifications for polytopes will be given. These developments will be utilized in the sections that follow.

### *2.1. Method for arbitrary regions*

Consider a closed, bounded domain $M \subset \mathbb{R}^d$ whose boundary is denoted by $\partial M$. The boundary $\partial M$ is defined by a set of $m$ $(d-1)$-dimensional orientable, smooth boundary faces $S_i \subset \mathcal{S}_i$ for $i = 1, \ldots, m$, where

$$\mathcal{S}_i := \left\{ \boldsymbol{x} \subset \mathbb{R}^d \ : \ \varphi(\boldsymbol{x}) = 0 \right\} \tag{1}$$

defines a hypersurface with $\varphi(\boldsymbol{x})$ being a scalar-valued level set function. In general, the boundary faces will satisfy the properties $S_1 \cup \ldots \cup S_m = \partial M$ and $S_1 \cap \ldots \cap S_m = \emptyset$, such that one and only one boundary face defines $\partial M$ at any point.

Let $f(\boldsymbol{x})$ be a positively homogeneous function of degree $q$ that is continuously differentiable:

$$f(\lambda \boldsymbol{x}) = \lambda^q f(\boldsymbol{x}) \quad (\lambda > 0), \tag{2}$$

which satisfies Euler's homogeneous function theorem:

$$qf(\boldsymbol{x}) = \nabla f(\boldsymbol{x}) \cdot \boldsymbol{x} \quad \forall \boldsymbol{x} \in E \subset \mathbb{R}^d, \tag{3}$$

where $E$ is the domain of definition of $f(\boldsymbol{x})$. We point out that the class of functions that are homogeneous extend beyond monomials. For example, functions in polar coordinates are also homogeneous: $f(r) = \sqrt{x^2 + y^2} := r$ with $E = \mathbb{R}^2$ and $g(r) = 1/r$ with $E = \mathbb{R}^2 \backslash \{\boldsymbol{0}\}$ are homogeneous functions with degrees-of-homogeneity $q = 1$ and $q = -1$, respectively.

Our objective is to integrate a homogeneous function, $f(\boldsymbol{x})$, over the domain $M \subset \mathbb{R}^d$, i.e.,

$$I := \int_M f(\boldsymbol{x})\, d\boldsymbol{x}. \tag{4}$$

Consider the generalized Stokes's theorem, which can be stated as ((see Taylor (1996)):

$$\int_M (\nabla \cdot \boldsymbol{X}(\boldsymbol{x})) f(\boldsymbol{x})\, d\boldsymbol{x} + \int_M \boldsymbol{X}(\boldsymbol{x}) \cdot \nabla f(\boldsymbol{x})\, d\boldsymbol{x} = \int_{\partial M} (\boldsymbol{X}(\boldsymbol{x}) \cdot \boldsymbol{n}) f(\boldsymbol{x})\, d\sigma, \tag{5}$$

where $d\sigma$ is the Lebesgue measure on $\partial P$.

For a homogeneous function $f$ and choosing $\boldsymbol{X}(\boldsymbol{x}) := \boldsymbol{x}$, (5) yields

$$d \int_M f(\boldsymbol{x})\, d\boldsymbol{x} + \int_M (\nabla f(\boldsymbol{x}) \cdot \boldsymbol{x}) d\boldsymbol{x} = \sum_{i=1}^{m} \int_{S_i} (\boldsymbol{x} \cdot \boldsymbol{n}) f(\boldsymbol{x})\, d\sigma. \tag{6}$$

Invoking Euler's theorem given in (3), (6) simplifies to (Chin et al., 2015; Lasserre, 1999)

$$\int_M f(\boldsymbol{x})\, d\boldsymbol{x} = \frac{1}{d+q} \sum_{i=1}^{m} \int_{S_i} (\boldsymbol{x} \cdot \boldsymbol{n}) f(\boldsymbol{x})\, d\sigma. \tag{7}$$

Equation (7) relates integration of a positively homogeneous, continuously differentiable function $f(\boldsymbol{x})$ over a domain in $\mathbb{R}^d$ to integration of the same function over the domain's $(d-1)$-dimensional boundary, $\partial M$. As a point of contrast and comparison between using the divergence theorem and the formula in (7), consider integrating $f(\boldsymbol{x}) = x + y$ over a domain $M$. Using the divergence theorem on the vector field $\boldsymbol{F} = x^2/2\, \boldsymbol{i} + y^2/2\, \boldsymbol{j}$, requires one to integrate $x^2/2$ and $y^2/2$ over the boundary, whereas using (7), we obtain a formula in which the integral of $f$ itself is to be computed over the boundary.

## *2.2. Simplifications for polytopes*

The equations developed in the previous section are applicable to polytopes, though further simplifications can be made to the integrals if the domain is known to be polytopal. On carrying out the simplification process repeatedly the evaluation of these integrals is reduced to vertex evaluation of the integrand and its partial derivatives. This idea is developed in this section. Consider a closed, bounded polytopal region $P \subset \mathbb{R}^d$ that is bounded by a set of $m$ $(d-1)$-dimensional affine, closed, orientable boundary facets, $F_i \subset \mathcal{H}_i$ for $i = 1, \ldots, m$, where

$$\mathcal{H}_i := \left\{ \boldsymbol{x} \in \mathbb{R}^d \ : \ \varphi(\boldsymbol{x}) = \boldsymbol{a}_i \cdot \boldsymbol{x} - b_i = 0 \right\} \tag{8}$$

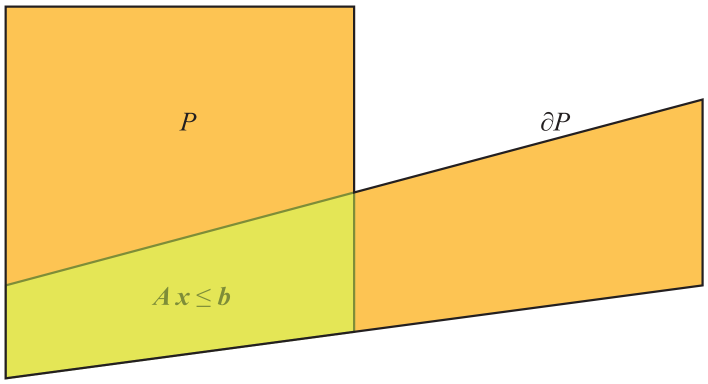


Figure 1: The region defined by $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$ is smaller than the nonconvex polygon $P$.

uniquely defines a hyperplane for some $\boldsymbol{a}_i \in \mathbb{R}^d$ and for some $b_i \in \mathbb{R}$. This definition, which was introduced in Chin et al. (2015), is broader than the one used in Lasserre (1998), since it now includes nonconvex polytopes. In Lasserre (1998), a convex polytope is defined by $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$ for a matrix $\boldsymbol{A}$ of dimensions $m \times d$, and a vector $\boldsymbol{b}$ of length $m$. As Fig. 1 illustrates, this definition does not apply to nonconvex polytopes.

Since the polytope $P$ with boundary facets $F_i$ also fit the definition of the region $M$ with boundary facets $S_i$, (7) is also applicable to $P$. Further, the normal of the hyperplane is given by

$$\boldsymbol{n} = \frac{\nabla\varphi(\boldsymbol{x})}{\|\nabla\varphi(\boldsymbol{x})\|} = \frac{\boldsymbol{a}_i}{\|\boldsymbol{a}_i\|}, \tag{9}$$

which is a constant. Substituting (9) into (7) yields

$$\int_P f(\boldsymbol{x})\,d\boldsymbol{x} = \frac{1}{d+q}\sum_{i=1}^{m}\frac{b_i}{\|\boldsymbol{a}_i\|}\int_{F_i} f(\boldsymbol{x})\,d\sigma, \tag{10}$$

where $\boldsymbol{a}_i \cdot \boldsymbol{x} = b_i$ is used. Note that the quantity $b_i/\|\boldsymbol{a}_i\|$ gives the signed distance from the origin $\boldsymbol{0}$ to the hyperplane $\mathcal{H}_i$. Therefore, we can also write (10) as

$$\int_P f(\boldsymbol{x})\,d\boldsymbol{x} = \frac{1}{d+q}\sum_{i=1}^{m} d(\boldsymbol{0},\mathcal{H}_i)\int_{F_i} f(\boldsymbol{x})\,d\sigma, \tag{11}$$

where we define the signed distance function $d(\alpha,\mathcal{H})$ as the distance from the point $\alpha$ to the hyperplane $\mathcal{H}$, where the sign is provided by the direction of the normal to $\mathcal{H}$.

We can further reduce the integration of $\int_{F_i} f(\boldsymbol{x})\,d\sigma$ through application of Stokes's theorem. Following Lasserre (1998), define $F_{ij} := F_i \cap F_j$ for $j \neq i$. $\mathcal{H}_{ij}$ is the $(d-2)$-dimensional hyperplane that is the intersection of $\mathcal{H}_i$ and $\mathcal{H}_j$, and $\boldsymbol{n}_{ij}$ is the $d$-dimensional unit vector that lies in $\mathcal{H}_i$ and is normal to $F_{ij}$. Now,

$$\boldsymbol{x} := \boldsymbol{x}_0 + \sum_{i=1}^{d-1} x'_i \boldsymbol{e}'_i \tag{12}$$

is a point in $\mathbb{R}^d$ that lies in $\mathcal{H}_i$. In (12), $\boldsymbol{x}_0 \in \mathcal{H}_i$ is an arbitrary point that satisfies $\boldsymbol{a}_i \cdot \boldsymbol{x}_0 = b_i$, and $\{\boldsymbol{e}'_i\}_{i=1}^{d-1}$ forms an orthonormal basis on the $(d-1)$-dimensional subspace $\mathcal{H}_i$. Note that the divergence of $\boldsymbol{x} - \boldsymbol{x}_0$ is $d-1$. After choosing the vector field $\boldsymbol{X}(\boldsymbol{x}) := \boldsymbol{x} - \boldsymbol{x}_0$ and applying Euler's homogeneous function theorem, (5) becomes

$$(d-1)\int_{F_i} f(\boldsymbol{x})\,d\sigma + q\int_{F_i} f(\boldsymbol{x})\,d\sigma = \sum_{j\neq i}\int_{F_{ij}} (\boldsymbol{x}-\boldsymbol{x}_0)\cdot\boldsymbol{n}_{ij} f(\boldsymbol{x})\,d\nu + \int_{F_i} (\nabla f(\boldsymbol{x})\cdot\boldsymbol{x}_0)\,d\sigma. \tag{13}$$

Note that $(\boldsymbol{x}-\boldsymbol{x}_0)\cdot\boldsymbol{n}_{ij}$ is the algebraic (signed) distance from $\boldsymbol{x}_0$ to $\mathcal{H}_{ij}$. Then, (13) simplifies to (Chin et al., 2015)

$$\int_{F_i} f(\boldsymbol{x})\,d\sigma = \frac{1}{d+q-1}\left[\sum_{j\neq i} d(\boldsymbol{x}_0,\mathcal{H}_{ij})\int_{F_{ij}} f(\boldsymbol{x})\,d\nu + \int_{F_i}(\nabla f(\boldsymbol{x})\cdot\boldsymbol{x}_0)d\sigma\right]. \tag{14}$$

When $f(\boldsymbol{x})$ is a homogeneous polynomial, the partial derivatives of $f$ eventually vanish, and hence (14) can be applied recursively to realize an exact cubature rule—integration over the polytope is reduced to evaluations of $f(\boldsymbol{x})$ and its partial derivatives at the vertices of $P$ (Chin et al., 2015). On combining (14) with (11), one obtains an explicit formula for the volume of a polygon and polyhedra in terms of the locations of its vertices, and we can also write down a closed-form expression for the integral of a homogeneous polynomial over a polygon (Chin et al., 2015).

## 3. Integration over polyhedra

### *3.1. Monomials*

Integrating monomials over polyhedra using (14) requires integrating partial derivatives over the boundary faces of the polyhedron. Further, as dimension reduces, repeated application of (14) requires integration of the monomial and its partial derivatives on lower dimensional entities, continuing to evaluation of the monomial and its partial derivatives at the vertices of the polyhedron.

The set of all monomials of degree less than or equal to $p$ is defined as

$$\mathcal{M}_p = \Big\{ f:\mathbb{R}^d \to \mathbb{R},\ f(\boldsymbol{x}) = x_1^{\alpha_1}x_2^{\alpha_2}\ldots x_d^{\alpha_d} \quad \forall \alpha_i \in \mathbb{Z}^{\geq} \text{ and } i = 1,\ldots,d \\ \text{such that } \alpha_1+\alpha_2+\ldots+\alpha_d \leq p \Big\}, \tag{15}$$

where $\mathbb{Z}^{\geq}$ is the set of integers greater than or equal to zero. If integration of all monomials in $\mathcal{M}_p$ is desired over a polytope, the integrals of monomials of degree $p-1$ are equivalent to the integrals of partial derivatives of a monomial of degree $p$, scaled by a constant. This observation can be used to significantly reduce calculations required to integrate all monomials in $\mathcal{M}_p$, by storing the values of the integrals of monomials of lower degree. The resulting integration rule is $O(n)$, where $n$ is the number of monomials to integrate.

Furthermore, if the polyhedron is stored in a graph-based structure that contains all higher and lower dimensional connectivities (with correct orientation), redundant integration can be eliminated by first evaluating vertex values, then integrating over edges, followed by integrating over faces, and finally integrating over the polyhedron. Also, if multiple, connected polyhedra are stored in a single graph, redundant integration can be eliminated over shared boundary facets. For example, eliminating redundant integration over the same geometric entities reduces the number of evaluations to compute the integral of monomials over a mesh of polyhedral finite elements.

A C++11 code that implements these ideas to integrate monomials over a polyhedron is provided in the supplementary material. In the provided code, $\boldsymbol{x}_0$ for each plane $\mathcal{H}_i$ and for each line $\mathcal{H}_{ij}$ is selected as the closest point on the plane/line to the origin. Note that this choice is arbitrary, provided $\boldsymbol{x}_0$ lies on the plane or the line. The required inputs to the function (named `ComputeIntegral()`) are *max_order*, the highest order monomials that will be integrated, and *polyhedron*, a `struct` that contains the following fields:

1. *polyhedron.coords*, a vector containing the vertex coordinates;
2. *polyhedron.edges*, a vector containing the vertex connectivities of each edge;
3. *polyhedron.faces*, a vector containing the edge connectivities of each face; and
4. *polyhedron.faces_dir*, a vector containing the direction of the edges on the face.

The function `ComputeIntegral()` first computes edge and face quantities, including the point $\boldsymbol{x}_0$ and the distance $d(\boldsymbol{x}_0, \mathcal{H}_{ij})$ for the edge $F_i \cap F_j$ and $d(\boldsymbol{x}_0, \mathcal{H}_i)$ for face $F_i$. These quantities are constants for the given polyhedron, so they can be computed once and stored. Within a loop over each monomial (where the monomial order stays the same or increases with each loop traversal), loops over the vertices, edges, and faces of the polyhedron are conducted sequentially. In the loop over vertices, the vertex values of the monomial are computed and stored. These vertex values and stored edge integrals of partial derivatives of the monomial are then used in the edge loop to integrate the monomial over each edge using (14). The edge integrals are stored for use in integrating subsequent monomials. Next, the edge integrals and the face integrals of the partial derivatives of the monomial are used in the face loop to compute and store the integral of the monomial over each face using (14), analogously to the loop over the edges. Finally, a second loop over each face is used to compute the integral over the polyhedron, using (11).

Integral values are returned in a two-dimensional array, *integrals*, where the first dimension is the monomial order and the second dimension identifies the monomial. Monomials are stored sequentially in terms of $x$, $y$, and $z$. For example, the monomials for $p = 2$ are $x^2$, $xy$, $xz$, $y^2$, $yz$, $z^2$. In addition to returning the integrals over the polyhedron, monomial integrals over the edges and faces of the polyhedron are also computed and stored and can be returned if desired. The function `ComputeIntegral()` is used in the following example to compute integrals of monomials over several convex and nonconvex polyhedra.

### *3.1.1. Numerical example*

The computer program `Mathematica`™ provides polyhedron geometry (vertex location and face connectivities) through the command `PolyhedronData[]`. Further, `Mathematica`™ can integrate functions over regions defined by these polyhedra, which provides a method to verify our results. The command `NIntegrate[]` is used to compute these integrals in `Mathematica`™. Monomials are integrated over the Szilassi polyhedron, the dodecahedron-small triambic icosahedron compound, and the echidnahedron from the `PolyhedronData[]` database. Also, monomials are integrated over the truncated hexagonal trapezohedron, which when combined with an

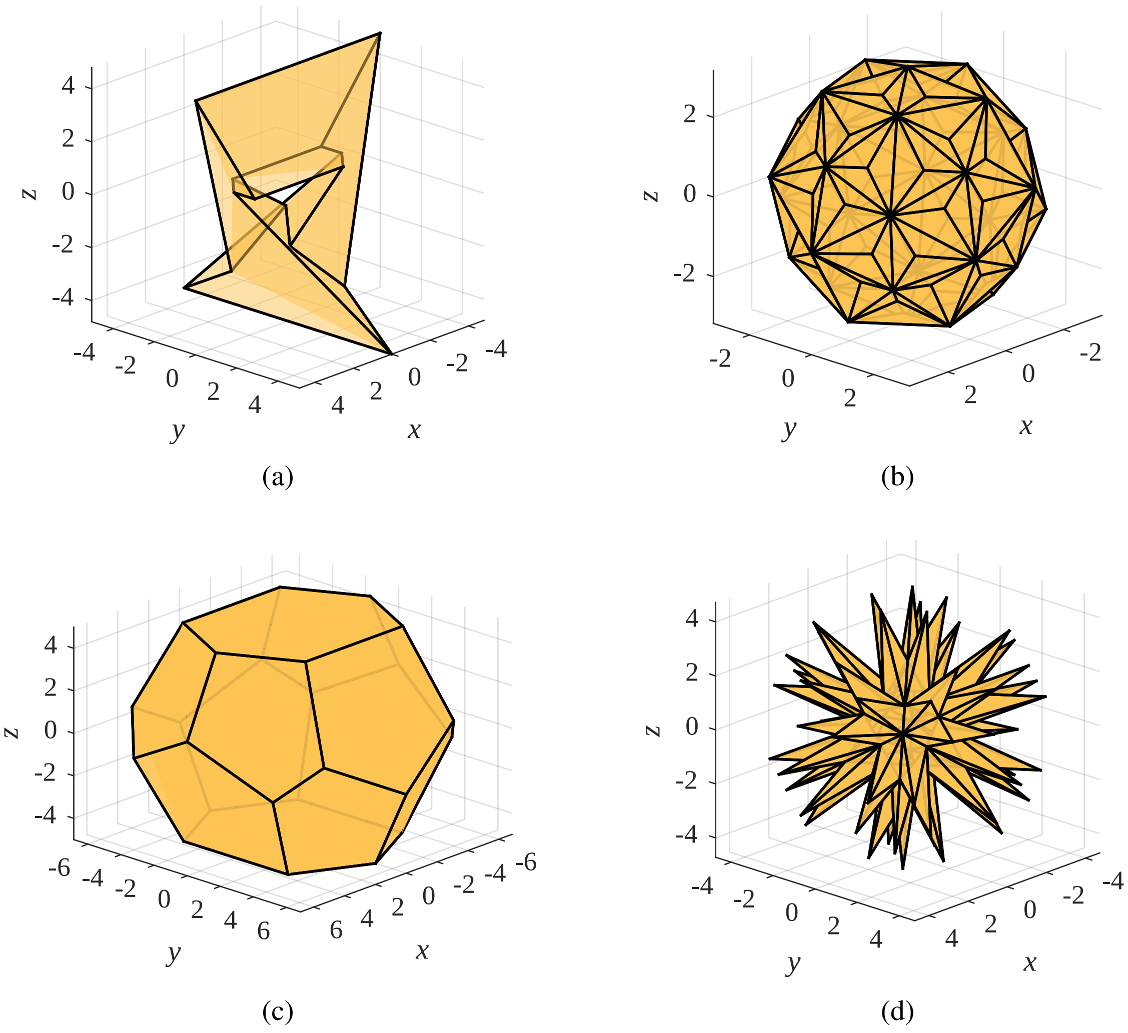


Figure 2: Integrating monomials over polyhedra. (a) Szilassi polyhedron (14 vertices, 7 faces), (b) dodecahedron-small triambic icosahedron compound (92 vertices, 180 faces), (c) truncated hexagonal trapezohedron (24 vertices, 14 faces), and (d) echidnahedron (92 vertices, 180 faces).

irregular dodecahedron is known to form the Weaire-Phelan structure, the most optimal solution to the Kelvin problem proposed thus far (Weaire and Phelan, 1994). The Kelvin problem is to find polyhedra with equal volume that fill $\mathbb{R}^3$ while minimizing the surface area of the faces in the structure, and has applications in the modeling of foam bubbles (Thomson, 1887). Illustrations with vertex and face counts for these polyhedra are provided in Fig. 2. As the order of the monomial increases, the accuracy of the solution reduces in both `Mathematica`$^{\text{TM}}$ and the C++11 function `ComputeIntegral()`; however, the accuracy of both solutions match. With the `ComputeIntegral()` function, this increased error is a result of cancellation errors caused by summing over the integrals of monomials with odd and even powers, as required by (14).

To demonstrate the benefit of storing and reusing integral values from lower order monomials,

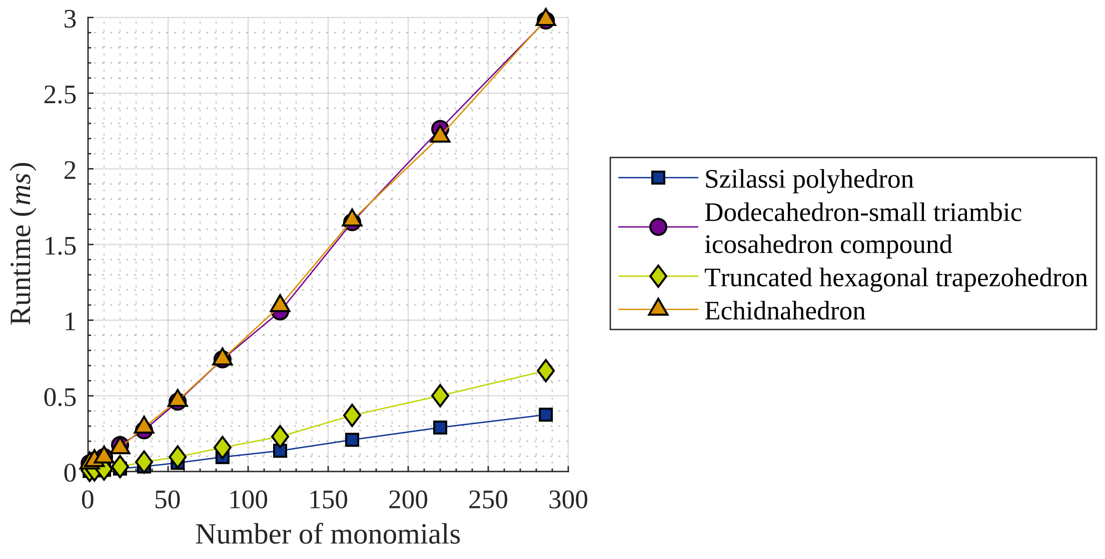


Figure 3: Runtime to integrate monomials up to order 10.

the C++ code is run with the four polyhedra in Fig. 2. The function is executed 10,000 times on each polyhedron with the maximum monomial order varied from 0 to 10. The function runtime is averaged over the 10,000 executions, and this time is plotted versus the number of monomials integrated (see Fig. 3).[1] As Fig. 3 reveals, the integration runtime increases approximately linearly with the number of monomials integrated, despite the additional partial derivatives that must be computed when integrating monomials of higher degree using (14).

To quantify integration error as monomial order increases, we compute the integrals of odd monomials of up to order 20 over the dodecahedron-small triambic icosahedron compound and the echidnahedron, which are symmetric about the Cartesian axes. Due to the symmetry of the polyhedra, the integral of an odd monomial function is zero. These results are presented in Fig. 4. The logarithm of the integration error scales approximately linearly as monomial order increases, though the error is dependent on the specific monomial and the shape of the polyhedron. For monomials that are the same order, generally integration error is the largest when the power of one of the variables approaches the order of the monomial.

## 3.2. *Inertia tensor computation*

For the purpose of computing the inertia tensor with HNI, only monomials of degree zero, one, and two need to be integrated over the polyhedron. This section develops simplified expressions for the integration of these monomials through recursive application of (11) and (14). Since only polyhedra are considered in this section, we make several explicit definitions that simplify the formulas that follow. We define a polyhedra $P \subset \mathbb{R}^3$ through a set of $\ell$ faces, $\mathcal{F} = \{F_i\}_{i=1}^{\ell}$, a set of $m$ edges, $\mathcal{E} = \{E_j\}_{j=1}^{m}$, and a set of $n$ vertices, $\mathcal{V} = \{V_k\}_{k=1}^{n}$. Each face, $F_i$, contains an index set of

[1]All timings are performed on Windows 10 computer with an Intel® Core$^{\text{TM}}$ i7-8550U CPU and 16 GB RAM.

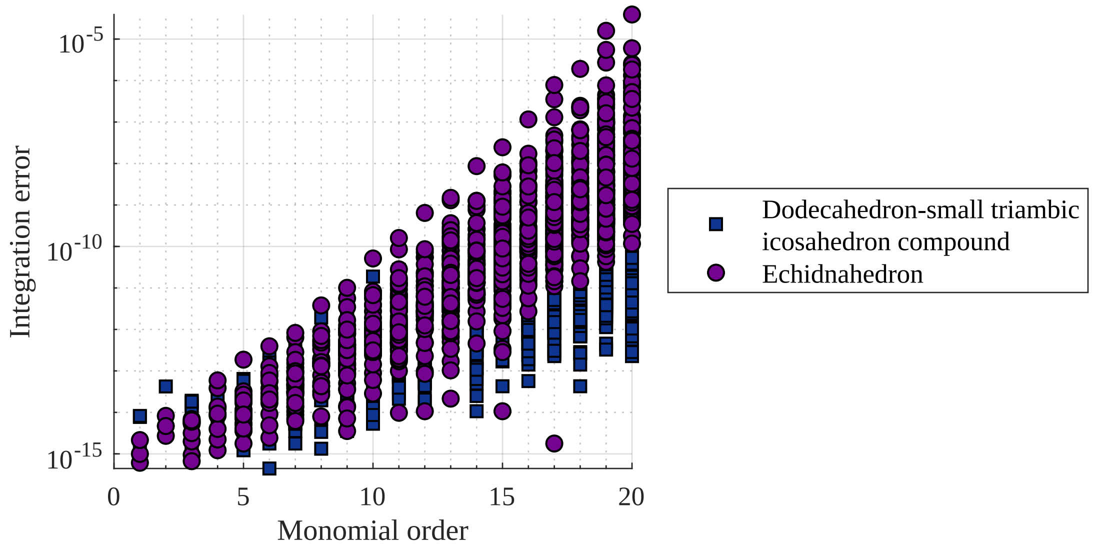


Figure 4: Error in integrating monomials that are odd functions over symmetric polyhedra.

edges,

$$\mathcal{E}_i := \left\{ j \in [1, m] \ : \ E_j \subset F_i \right\}, \tag{16}$$

and each edge, $E_j$, contains an index set of vertices,

$$\mathcal{V}_j := \left\{ k \in [1, n] \ : \ V_k \subset E_j \right\}. \tag{17}$$

Also, each face $F_i$ is the subset of a plane whose location is given by the affine variety

$$\bar{F}_i := \left\{ \boldsymbol{x} \in \mathbb{R}^3 \ : \ \varphi(\boldsymbol{x}) = \boldsymbol{a}_i \cdot \boldsymbol{x} - b_i = 0 \right\} \tag{18}$$

for some vector $\boldsymbol{a}_i \in \mathbb{R}^3$ and scalar $b_i \in \mathbb{R}$, and each edge $E_j$ is the subset of a line whose location, $\bar{E}_j$, is the intersection of two planes. Finally, each vertex $V_k$ contains a vertex coordinate, $\boldsymbol{x}_k$.

For a constant function, we write the integrals over a polyhedron $P$, its faces, and its edges explicitly as follows:

$$\int_P d\boldsymbol{x} = \frac{1}{3} \sum_{i \in \mathcal{F}} d(\boldsymbol{0}, \bar{F}_i) \int_{F_i} d\boldsymbol{x}, \tag{19a}$$

$$\int_{F_i} d\boldsymbol{x} = \frac{1}{2} \sum_{j \in \mathcal{E}_i} d(\boldsymbol{\xi}_i, \bar{E}_j) \int_{E_j} d\boldsymbol{x}, \tag{19b}$$

$$\int_{E_j} d\boldsymbol{x} = \sum_{k \in \mathcal{V}_j} d(\boldsymbol{\eta}_j, V_k) = \ell(E_j), \tag{19c}$$

where $\ell(E_j)$ is the length of edge $E_j$, $\boldsymbol{\xi}_i$ is a point on $\bar{F}_i$, and $\boldsymbol{\eta}_j$ is a point on $\bar{E}_j$. Selecting $\boldsymbol{\eta}_j$ as $\boldsymbol{x}_k$, the coordinate of $V_k$ for $k \in \mathcal{V}_j$, causes $d(\boldsymbol{\eta}_j, V_k)$ to be zero for one of the vertices. Since

$|\mathcal{V}_j| = 2$, this reduces the summation in (19c) to a single distance evaluation. Also, selecting $\boldsymbol{\xi}_i$ as the coordinate of a vertex indexed in $E_j$ for $j \in \mathcal{E}_i$ causes $d(\boldsymbol{\xi}_i, \bar{E}_j)$ to be zero for two of the edges, eliminating two terms from the summation. Finally, we can select one of the vertices to serve as the origin for the coordinate system by shifting the given coordinate system by

$$\tilde{\boldsymbol{x}} = \boldsymbol{x} - \boldsymbol{x}_\ell, \tag{20}$$

where $\boldsymbol{x}_\ell$ is the coordinate of $V_\ell$, where $\ell \in \mathcal{V}$. Integration can then be conducted in the $\tilde{\boldsymbol{x}}$ coordinate system. This is handled by first shifting vertex coordinates using (20). Observe that, for non-constant integrands, the resulting integrals must then be mapped back to $\boldsymbol{x}$ coordinates. With the coordinates shifted such that the origin is located at a vertex coordinate, $d(\boldsymbol{0}, \bar{F}_i)$ is zero for three or more faces that contain the vertex $V_\ell$. Using shifted coordinates so that the origin is at a vertex and selecting vertices as $\boldsymbol{\xi}_i$ and $\boldsymbol{\eta}_j$ also eliminates loss of precision errors that are caused by these points being far from the polyhedron. The effects of loss of precision errors are evaluated in Section 3.2.2.

We combine (19b) and (19c), shift all vertex coordinates such that a vertex is located at the origin, and select (shifted) vertex coordinates as $\boldsymbol{\xi}_i$ and $\boldsymbol{\eta}_j$ to produce a simplified expression for determining the area of planar facets:

$$\int_{F_i} d\tilde{\boldsymbol{x}} = \frac{1}{2} \sum_{\substack{j \in \mathcal{E}_i \\ \boldsymbol{\xi}_i \notin \bar{E}_j}} d(\boldsymbol{\xi}_i, \bar{E}_j)\, \ell(E_j). \tag{21}$$

Equation (21) can be interpreted as the area of triangles with one vertex at $\boldsymbol{\xi}_i$ and the other two vertices in the set $\mathcal{V}_j$. Finally, combining (21) with (19a) gives

$$\int_P d\tilde{\boldsymbol{x}} = \frac{1}{6} \sum_{\substack{i \in \mathcal{F} \\ \boldsymbol{0} \notin \bar{F}_i}} \left[ d(\boldsymbol{0}, \bar{F}_i) \sum_{\substack{j \in \mathcal{E}_i \\ \boldsymbol{\xi}_i \notin \bar{E}_j}} d(\boldsymbol{\xi}_i, \bar{E}_j)\, \ell(E_j) \right], \tag{22}$$

where each term can be interpreted as the signed volume of a tetrahedron, where two vertices of the tetrahedron are located at $\boldsymbol{0}$ and $\boldsymbol{\xi}_i$ on $F_i$ and the other two vertices are $V_k$ for all $k \in \mathcal{V}_j$. In (22), the benefit of shifting coordinates and selecting a vertex coordinate as $\boldsymbol{\xi}_i$ and $\boldsymbol{\eta}_j$ becomes easier to interpret. For tetrahedra, (22) reduces to a single term (versus 24 for the general case) whereas for a hexahedra, (22) reduces to six terms (versus 48 for the general case). In general, as the number of edges and faces in the polyhedron increases, the gains in computational efficiency diminish.

From (11) and (14), the integral expressions for linear monomials over the polyhedron, its faces, and its edges are:

$$\int_P x_a \, d\boldsymbol{x} = \frac{1}{4} \sum_{i \in \mathcal{F}} d(\boldsymbol{0}, \bar{F}_i) \int_{F_i} x_a \, d\boldsymbol{x}, \tag{23a}$$

$$\int_{F_i} x_a \, d\boldsymbol{x} = \frac{1}{3} \left[ \sum_{j \in \mathcal{E}_i} d(\boldsymbol{\xi}_i, \bar{E}_j) \int_{E_j} x_a \, d\boldsymbol{x} + (\xi_a)_i \int_{F_i} d\boldsymbol{x} \right], \tag{23b}$$

$$\int_{E_j} x_a \, d\boldsymbol{x} = \frac{1}{2} \left[ \sum_{k \in \mathcal{V}_j} d(\boldsymbol{\eta}_j, V_k)(x_a)_k + (\eta_a)_j \int_{E_j} d\boldsymbol{x} \right], \tag{23c}$$

where the subscript $a$ for $a = 1, 2, 3$ refers to $x$-, $y$-, and $z$-component of the coordinate, respectively, and $\boldsymbol{x}_k$ is the coordinate of vertex $V_k$. As with the constant function, we can select the origin, $\boldsymbol{\xi}_i$, and $\boldsymbol{\eta}_j$ as vertex coordinates in the polyhedron to reduce the number of terms required to compute the integral. Accordingly, the mapping in (20) is applied and integration is performed in $\tilde{\boldsymbol{x}}$ coordinates. The benefits of doing so remain the same, namely increased computational efficiency and potentially improved accuracy.

We combine (23c) and (19c), shift coordinates such that a vertex coincides with the origin, and select a (shifted) vertex coordinate for $\boldsymbol{\eta}_j$ to produce a simplified expression for integrating a linear monomial over an edge:

$$\int_{E_j} \tilde{x}_a \, d\tilde{\boldsymbol{x}} = \frac{1}{2} \ell(E_j) \sum_{k \in \mathcal{V}_j} (x_a)_k. \tag{24}$$

The two terms give the area of triangles, which when combined, yield the result of the integral over the edge. An example illustrating the contribution of each term to the integral is presented in Fig. 5a. Next, utilizing (24), (23b), and (19b), a simplified formula for integrating linear monomials over a facet is

$$\int_{F_i} \tilde{x}_a \, d\tilde{\boldsymbol{x}} = \frac{1}{6} \sum_{\substack{j \in \mathcal{E}_i \\ \boldsymbol{\xi}_i \notin \bar{E}_j}} d(\boldsymbol{\xi}_i, \bar{E}_j) \ell(E_j) \left( (\xi_a)_i + \sum_{k \in \mathcal{V}_j} (x_a)_k \right). \tag{25}$$

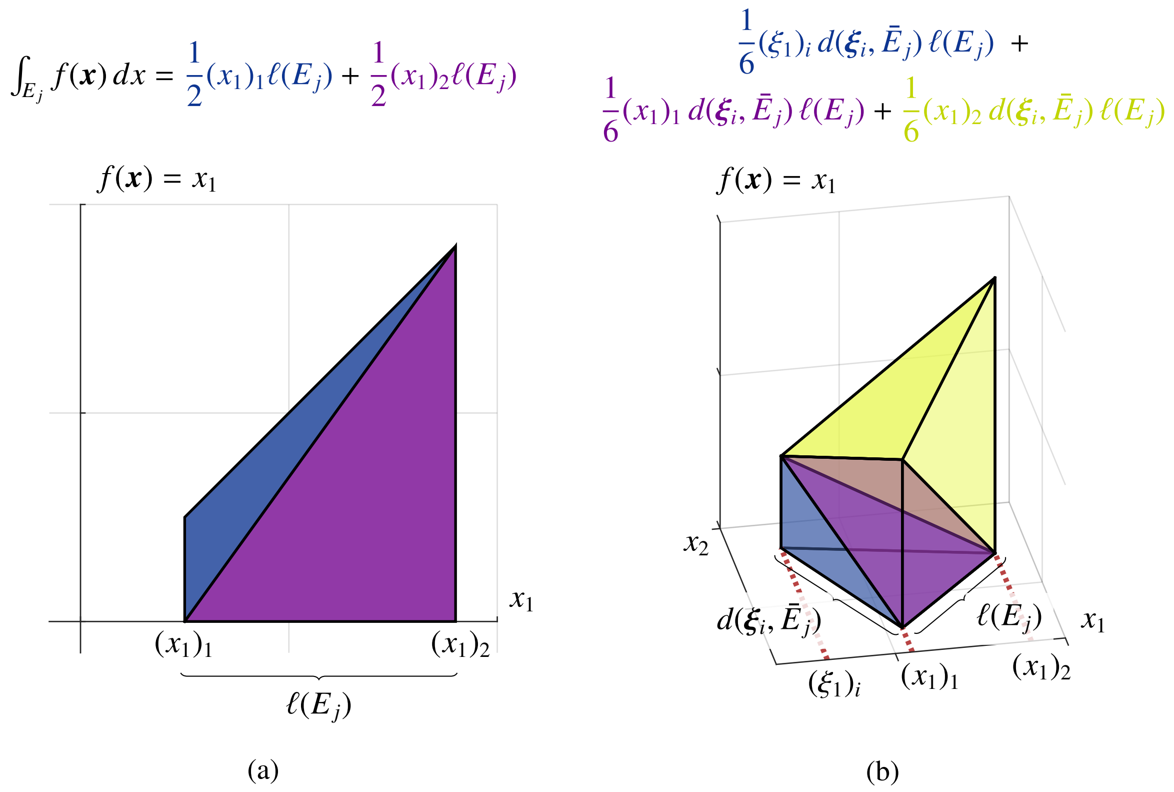
As illustrated in Fig. 5b, the three terms on each edge correspond to the volume of tetrahedra that result in the integral when combined. Furthermore, applying (25) to (23a), shifting coordinates, and selecting vertex coordinates as $\boldsymbol{\xi}_i$ and $\boldsymbol{\eta}_j$ yields

$$\int_P \tilde{x}_a \, d\tilde{\boldsymbol{x}} = \frac{1}{24} \sum_{\substack{i \in \mathcal{F} \\ \boldsymbol{0} \notin F_i}} \left[ d(\boldsymbol{0}, \bar{F}_i) \sum_{\substack{j \in \mathcal{E}_i \\ \boldsymbol{\xi}_i \notin \bar{E}_j}} d(\boldsymbol{\xi}_i, \bar{E}_j) \, \ell(E_j) \left( (\xi_a)_i + \sum_{k \in \mathcal{V}_j} (x_a)_k \right) \right], \tag{26}$$

where each term is the hypervolume of a 5-cell (a simplex in four dimensions).

The integrals can be shifted back into $\boldsymbol{x}$ coordinates by inverting the mapping in (20) and using the linearity of integration to obtain

$$\int_P x_a \, d\boldsymbol{x} = \int_P \tilde{x}_a \, d\tilde{\boldsymbol{x}} + (x_a)_\ell \int_P d\tilde{\boldsymbol{x}}, \tag{27}$$



$\int_{E_j} f(\boldsymbol{x})\,dx = \frac{1}{2}(x_1)_1\ell(E_j) + \frac{1}{2}(x_1)_2\ell(E_j)$
$f(\boldsymbol{x}) = x_1$
$x_1$
$(x_1)_1$
$(x_1)_2$
$\ell(E_j)$
(a)
$\frac{1}{6}(\xi_1)_i\, d(\boldsymbol{\xi}_i, \bar{E}_j)\,\ell(E_j)$ +
$\frac{1}{6}(x_1)_1\, d(\boldsymbol{\xi}_i, \bar{E}_j)\,\ell(E_j) + \frac{1}{6}(x_1)_2\, d(\boldsymbol{\xi}_i, \bar{E}_j)\,\ell(E_j)$
$f(\boldsymbol{x}) = x_1$
$x_2$
$d(\boldsymbol{\xi}_i, \bar{E}_j)$
$\ell(E_j)$
$x_1$
$(\xi_1)_i$
$(x_1)_1$
$(x_1)_2$
(b)

Figure 5: Visualization of integrals of a linear monomial. (a) Integral over an edge (sum of two triangular areas) and (b) integral over a triangular face (sum of three tetrahedral volumes).

where the first and second terms are obtained from (26) and (22), respectively. In addition, from (20), we have used $d\boldsymbol{x} = d\tilde{\boldsymbol{x}}$.

For a quadratic monomial, (11) and (14) also provide expressions for the integral over the polyhedron, its faces, and its edges:

$$\int_P x_a x_b\, d\boldsymbol{x} = \frac{1}{5}\sum_{i\in\mathcal{F}} d(\boldsymbol{0}, \bar{F}_i) \int_{F_i} x_a x_b\, d\boldsymbol{x}, \tag{28a}$$

$$\int_{F_i} x_a x_b\, d\boldsymbol{x} = \frac{1}{4}\left[\sum_{j\in\mathcal{E}_i} d(\boldsymbol{\xi}_i, \bar{E}_j) \int_{E_j} x_a x_b\, d\boldsymbol{x} + (\xi_a)_i \int_{F_i} x_b\, d\boldsymbol{x} + (\xi_b)_i \int_{F_i} x_a\, d\boldsymbol{x}\right], \tag{28b}$$

$$\int_{E_j} x_a x_b\, d\boldsymbol{x} = \frac{1}{3}\left[\sum_{k\in\mathcal{V}_j} d(\boldsymbol{\eta}_j, V_k)(x_a)_k(x_b)_k + (\eta_a)_j \int_{E_j} x_b\, d\boldsymbol{x} + (\eta_b)_j \int_{E_j} x_a\, d\boldsymbol{x}\right]. \tag{28c}$$

As with the constant and linear monomials, we shift the coordinates such that the origin is located at a vertex coordinate and we select $\boldsymbol{\xi}_i$ and $\boldsymbol{\eta}_j$ as vertex coordinates to improve solution efficiency and accuracy.

With (28c) and (24), we develop a simplified expression for integrating quadratic monomials over an edge:

$$\int_{E_j} \tilde{x}_a \tilde{x}_b \, d\tilde{\boldsymbol{x}} = \frac{1}{6}\ell(E_j)\left[\sum_{k\in\mathcal{V}_j}(x_a)_k(x_b)_k + \sum_{k\in\mathcal{V}_j}\left((x_a)_k \sum_{\ell\in\mathcal{V}_j}(x_b)_\ell\right)\right]. \tag{29}$$

The simplified case when $a = b = 1$ is illustrated in Fig. 6. For this case, (29) reduces to three terms, which are the areas under parabolas that result in the integral value when combined. To integrate quadratic monomials over facets, (28b) can be simplified by shifting coordinates to $\tilde{\boldsymbol{x}}$, selecting $\boldsymbol{\xi}_i$ as a vertex, and applying (29) and (25). The simplified expression is

$$\int_{F_i} \tilde{x}_a \tilde{x}_b \, d\tilde{\boldsymbol{x}} = \frac{1}{24} \sum_{\substack{j\in\mathcal{E}_i \\ \boldsymbol{\xi}_i \notin \bar{E}_j}} d(\boldsymbol{\xi}_i, \bar{E}_j)\ell(E_j)\left[\sum_{k\in\mathcal{V}_j}(x_a)_k(x_b)_k + \sum_{k\in\mathcal{V}_j}\left((x_a)_k \sum_{\ell\in\mathcal{V}_j}(x_b)_\ell\right)\right.$$
$$\left. +(\xi_a)_i\left((\xi_b)_i + \sum_{k\in\mathcal{V}_j}(x_b)_k\right) + (\xi_b)_i\left((\xi_a)_i + \sum_{k\in\mathcal{V}_j}(x_a)_k\right)\right]. \tag{30}$$

Finally, we apply (30) to (28a), shift coordinates such that a vertex is at the origin, and select vertex coordinates for $\boldsymbol{\xi}_i$ and $\boldsymbol{\eta}_j$ to obtain a simplified expression to integrate quadratic monomials over a polyhedron:

$$\int_P \tilde{x}_a \tilde{x}_b \, d\tilde{\boldsymbol{x}} = \frac{1}{120} \sum_{\substack{i\in\mathcal{F} \\ \mathbf{0}\notin F_i}} \left\{ d(\mathbf{0}, \bar{F}_i) \sum_{\substack{j\in\mathcal{E}_i \\ \boldsymbol{\xi}_i \notin \bar{E}_j}} d(\boldsymbol{\xi}_i, \bar{E}_j)\ell(E_j)\left[\sum_{k\in\mathcal{V}_j}(x_a)_k(x_b)_k + \sum_{k\in\mathcal{V}_j}\left((x_a)_k \sum_{\ell\in\mathcal{V}_j}(x_b)_\ell\right)\right.\right.$$
$$\left.\left. +(\xi_a)_i\left((\xi_b)_i + \sum_{k\in\mathcal{V}_j}(x_b)_k\right) + (\xi_b)_i\left((\xi_a)_i + \sum_{k\in\mathcal{V}_j}(x_a)_k\right)\right]\right\}. \tag{31}$$

The integral is shifted from $\tilde{\boldsymbol{x}}$ coordinates back to $\boldsymbol{x}$ coordinates by computing

$$\int_P x_a x_b \, d\boldsymbol{x} = \int_P \tilde{x}_a \tilde{x}_b \, d\tilde{\boldsymbol{x}} - (x_b)_\ell \int_P \tilde{x}_a \, d\tilde{\boldsymbol{x}} - (x_a)_\ell \int_P \tilde{x}_b \, d\tilde{\boldsymbol{x}} + (x_a)_\ell (x_b)_\ell \int_P d\tilde{\boldsymbol{x}}, \tag{32}$$

where the integrals computed using (31), (26), and (22) are utilized.

A C++11 code that implements these equations is provided in the supplementary material. This code is used to generate the results presented in Sections 3.2.1 and 3.2.2.

### *3.2.1. Numerical example: Integration timings*

Using polyhedron information from the `PolyhedronData[]` command in Mathematica, run-times to integrate monomials of up to degree two for 195 different polyhedra are recorded. Complexity of the polyhedra that are integrated ranges from a tetrahedron (4 faces and 4 vertices) to a

$$\int_{E_j} f(\boldsymbol{x})\,dx = \frac{1}{3}(x_1)_1^2\,\ell(E_j) + \frac{1}{3}(x_1)_1(x_1)_2\,\ell(E_j) + \frac{1}{3}(x_1)_2^2\,\ell(E_j)$$

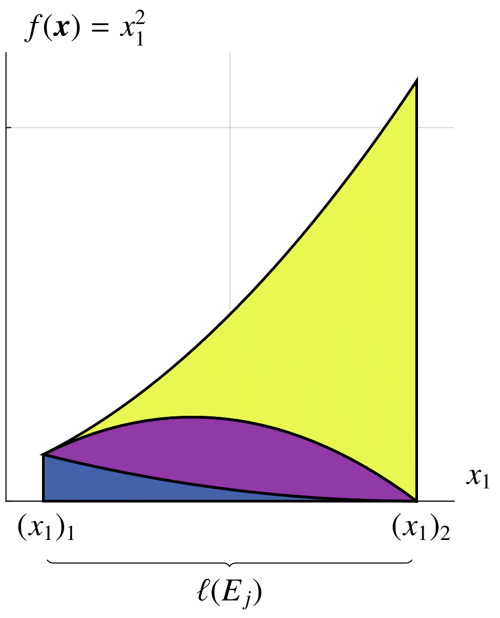


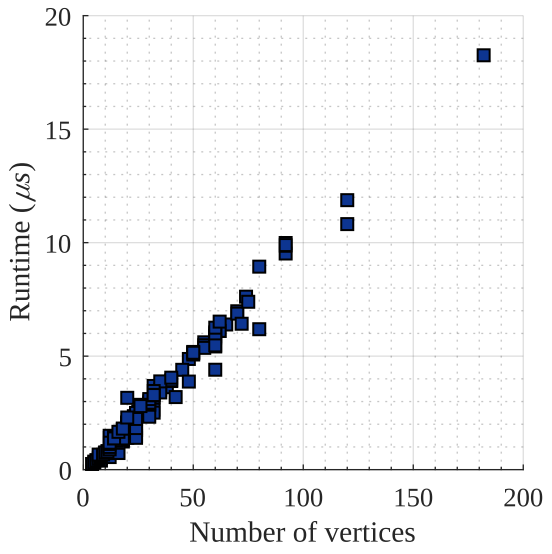
Figure 6: Visualization of integral of $x_1^2$ over an edge (sum of three parabolic areas).

great rhombic triacontahedron (180 faces and 182 vertices). Integration is conducted using formulas developed in Section 3.2 and implemented in a C++11 code that is provided in the supplementary material. Integration of all monomials is performed 100,000 times over each polyhedron and the average time is recorded. In Fig. 7, the time to integrate all monomials versus the number of vertices and versus the number of faces is plotted. In this figure, the linear relationship between the number of vertices/faces in the polyhedron and the time to integrate monomials is evident. As a measure of comparison, constant, linear, and quadratic monomials over the same polyhedra are integrated using a C code from Mirtich (1996)[2]. Time to integrate all monomials versus the number of vertices and versus the number of faces is also plotted in Fig. 7. As with the C++ code developed from the equations in this section, a linear relationship between number of vertices/faces and the time to integrate all monomials is observed. Figure 7 reveals that our method is more efficient than the scheme due to Mirtich (1996).

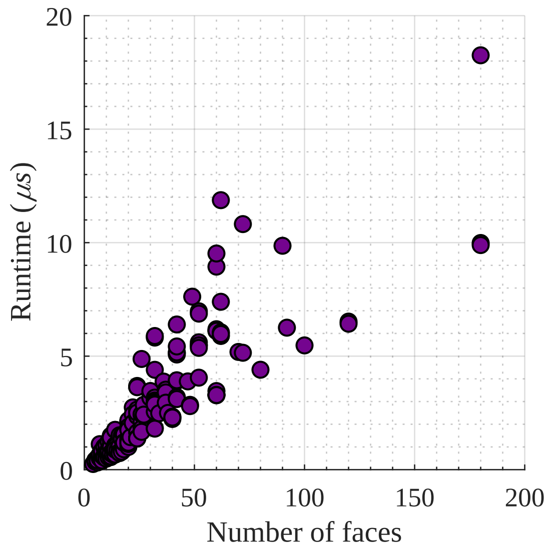
### 3.2.2. *Numerical example: Shifted coordinates*

To demonstrate the benefits of shifting the coordinate system such that the origin is located at a vertex coordinate, we compute the integral of monomials of degree 2 or less over a biunit cube. The $x$-coordinate of the centroid of the cube is varied from 0 to $1\times 10^{16}$, while the centroid of the $y$-

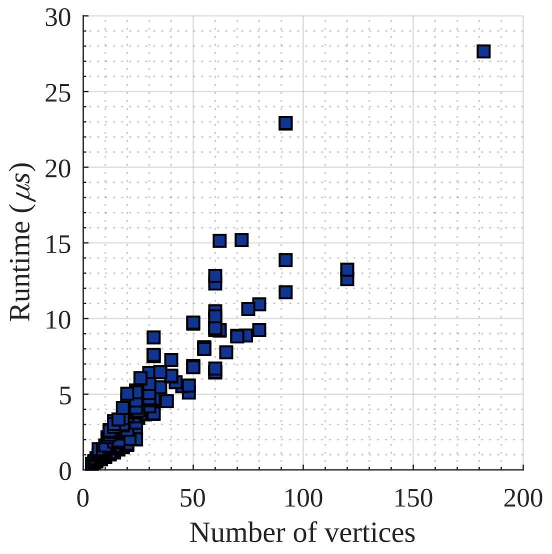
[2]Accessed from `https://people.eecs.berkeley.edu/~jfc/mirtich/code/volumeIntegration.tar`



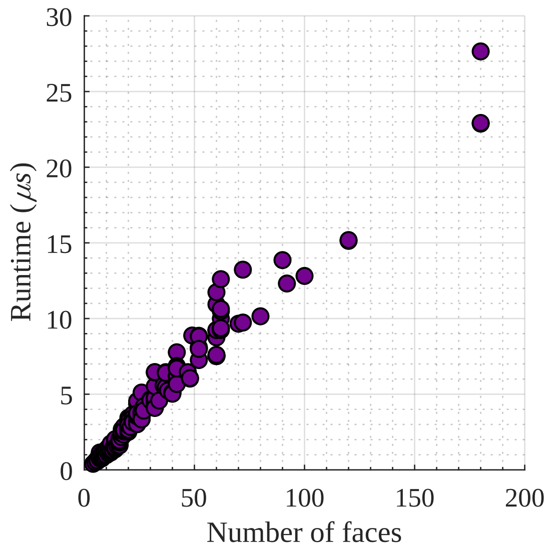
(a)



(b)



(c)



(d)

Figure 7: Timing to compute integrals of constant, linear, and quadratic monomials over 195 different polyhedra using (a–b) C++11 code provided in the supplementary material, and using (c–d) code from Mirtich (1996). (a), (c) Timing versus number of vertices and (b), (d) timing versus number of faces.

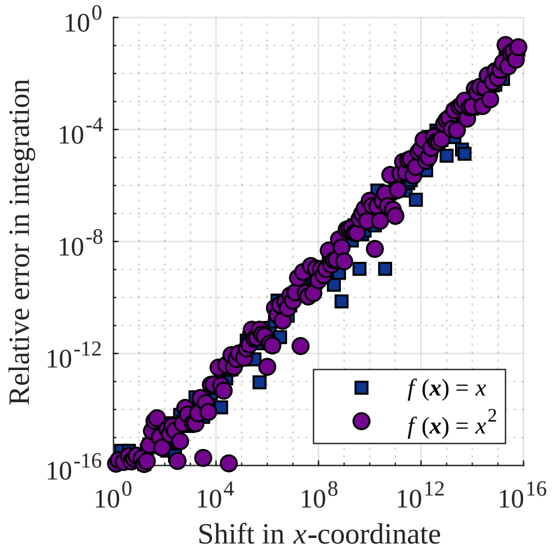


Figure 8: Integration error over a shifting cube with unshifted coordinates.

and $z$-coordinate remains at 0. The integrals of $x$ and $x^2$ are computed in both the unshifted ($\boldsymbol{x}$) and the shifted ($\tilde{\boldsymbol{x}}$) coordinate system. For all integrals in the shifted coordinates, relative integration error is $O(10^{-16})$ or better, whereas in the unshifted coordinate system, integration error scales proportionally to the amount of the shift in the $x$-coordinate. Relative integration error versus the shift in the $x$-coordinate is plotted in Fig. 8.

## 4. Integration over curved regions

### *4.1. Region bounded by Bézier triangles*

Bézier surface triangles are a natural, three-dimensional extension to Bézier curves. As with Bézier curves, Bézier triangles utilize Bernstein polynomials defined in barycentric coordinates to define geometry. Many appealing properties and tools of Bézier curves carry over to Bézier surface triangles, such as interpolation at the corners of the triangle, the convex hull property, subdivision, degree elevation, and the de Casteljau algorithm (Farin, 1986; Farouki, 2008). The surface of a degree-$n$ triangle is given by

$$\boldsymbol{b}(u, v, w) = \sum_{\substack{0 \le i,j,k \le n \\ i+j+k=n}} \boldsymbol{p}_{ijk} b^n_{ijk}(u, v, w), \tag{33a}$$

where

$$b^n_{ijk}(u, v, w) = \frac{n!}{i!j!k!} u^i v^j w^k, \tag{33b}$$

and $u$, $v$, and $w$ are barycentric coordinates of the triangle that satisfy the properties

$$\begin{gathered} 0 \le u, v, w \le 1 \\ u + v + w = 1. \end{gathered} \tag{34}$$

An example Bézier surface triangle and its control net are plotted in Fig. 9. Also, rational Bézier surface patches can be constructed similarly to rational Bézier curves through introduction of weights at each of the control points:

$$\boldsymbol{b}(u,v,w) = \frac{\sum_{\substack{0\le i,j,k\le n\\ i+j+k=n}} w_{ijk}\boldsymbol{p}_{ijk} b^n_{ijk}(u,v,w)}{\sum_{\substack{0\le i,j,k\le n\\ i+j+k=n}} w_{ijk} b^n_{ijk}(u,v,w)}. \tag{35}$$

We define barycentric unit basis vectors $\boldsymbol{\alpha} = [1,0,-1]^T$ and $\boldsymbol{\beta} = [0,1,-1]^T$. These basis vectors are used to define directional derivatives on the surface:

$$\frac{\partial \boldsymbol{b}}{\partial \alpha} = \frac{\partial \boldsymbol{b}}{\partial \boldsymbol{u}} \cdot \boldsymbol{\alpha} \qquad \text{and} \qquad \frac{\partial \boldsymbol{b}}{\partial \boldsymbol{\beta}} = \frac{\partial \boldsymbol{b}}{\partial \boldsymbol{u}} \cdot \boldsymbol{\beta}, \tag{36}$$

where $\boldsymbol{u} := [u,v,w]^T$ is a vector of components of the barycentric coordinate. With (7), integration over a region $R \subset \mathbb{R}^3$ is reduced to integration over its boundary faces. When these boundary faces are defined using Bézier surface triangles (rational or otherwise), we map the face to a triangle defined in barycentric coordinates. The Jacobian of the mapping is provided by the norm of the cross product of the directional derivatives in (36), leading to

$$\int_R f(\boldsymbol{x})\,d\boldsymbol{x} = \frac{1}{2}\left(\frac{1}{3+q}\right)\sum_{i=1}^{m}\int_{\triangle}\big(\boldsymbol{b}_i \cdot \boldsymbol{n}(\boldsymbol{b}_i)\big) f(\boldsymbol{b}_i)\left\|\frac{\partial \boldsymbol{b}_i}{\partial \alpha} \times \frac{\partial \boldsymbol{b}_i}{\partial \boldsymbol{\beta}}\right\| d\boldsymbol{u}, \tag{37}$$

where $\boldsymbol{b}_i$ is the Bézier surface triangle defining the $\boldsymbol{i}$-th boundary face. The normal can be computed explicitly as

$$\boldsymbol{n}(\boldsymbol{b}_i) = \frac{\frac{\partial \boldsymbol{b}_i}{\partial \alpha} \times \frac{\partial \boldsymbol{b}_i}{\partial \beta}}{\left\|\frac{\partial \boldsymbol{b}_i}{\partial \alpha} \times \frac{\partial \boldsymbol{b}_i}{\partial \beta}\right\|}, \tag{38}$$

which can be substituted into (37), giving

$$\int_R f(\boldsymbol{x})\,d\boldsymbol{x} = \frac{1}{2}\left(\frac{1}{3+q}\right)\sum_{i=1}^{m}\int_{\triangle}\left[\boldsymbol{b}_i \cdot \left(\frac{\partial \boldsymbol{b}_i}{\partial \alpha} \times \frac{\partial \boldsymbol{b}_i}{\partial \boldsymbol{\beta}}\right)\right] f(\boldsymbol{b}_i)\,d\boldsymbol{u}. \tag{39}$$

Next, on applying the identity $(\boldsymbol{T}\cdot\boldsymbol{c})\times(\boldsymbol{T}\cdot\boldsymbol{d}) = \boldsymbol{T}^{-T}\cdot(\boldsymbol{c}\times\boldsymbol{d})\det\boldsymbol{T}$ for $\boldsymbol{c} = \boldsymbol{\alpha}$, $\boldsymbol{d} = \boldsymbol{\beta}$, and $\boldsymbol{T} = \partial\boldsymbol{b}_i/\partial\boldsymbol{u}$, we obtain

$$\int_R f(\boldsymbol{x})\,d\boldsymbol{x} = \frac{1}{2}\left(\frac{1}{3+q}\right)\sum_{i=1}^{m}\int_{\triangle}\left[\boldsymbol{b}_i \cdot \left(\frac{\partial \boldsymbol{b}_i}{\partial \boldsymbol{u}}\right)^{-T} \cdot \boldsymbol{1}\right] f(\boldsymbol{b}_i)\det\left(\frac{\partial \boldsymbol{b}_i}{\partial \boldsymbol{u}}\right) d\boldsymbol{u}, \tag{40}$$

where $\boldsymbol{\alpha}\times\boldsymbol{\beta} = \boldsymbol{1} = [1,1,1]^T$. Computing the adjugate of $\partial\boldsymbol{b}_i/\partial\boldsymbol{u}$ and combining determinants results in the final expression:

$$\begin{aligned}\int_R f(\boldsymbol{x})\,d\boldsymbol{x} = \frac{1}{2}\left(\frac{1}{3+q}\right)\sum_{i=1}^{m}\int_{\triangle}\Bigg((b_1)_i \det\left[\boldsymbol{1}, \frac{\partial(b_2)_i}{\partial\boldsymbol{u}}, \frac{\partial(b_3)_i}{\partial\boldsymbol{u}}\right] + \\ (b_2)_i \det\left[\frac{\partial(b_1)_i}{\partial\boldsymbol{u}}, \boldsymbol{1}, \frac{\partial(b_3)_i}{\partial\boldsymbol{u}}\right] + (b_3)_i \det\left[\frac{\partial(b_1)_i}{\partial\boldsymbol{u}}, \frac{\partial(b_2)_i}{\partial\boldsymbol{u}}, \boldsymbol{1}\right]\Bigg) f(\boldsymbol{b})\,d\boldsymbol{u},\end{aligned} \tag{41}$$

where $(b_1)_i$, $(b_2)_i$, and $(b_3)_i$ refer to components of the surface $\boldsymbol{b}_i(u, v, w)$. Note that (41) is not specific to a particular type of barycentric triangular surface, and is equally applicable to both rational and polynomial Bézier surface triangles, as well as any other barycentric surface representation.

Numerical integration of (41) can be accomplished using a triangular cubature rule. A cubature rule formulated in barycentric coordinates, such as the rule by Dunavant (1985), is particularly appealing since it can be directly applied to (41) without either further mapping or eliminating one of the variables. For a polynomial Bézier surface triangle, the Dunavant (1985) rule can integrate (41) exactly. For the integrands required to compute the inertia tensor (monomials of degree 2 or less), the maximum polynomial degree in (41) where $\boldsymbol{b}_i$ is a degree-$n$ Bézier surface triangle is $3n + 2(n - 1)$.

### *4.1.1. Numerical example: Region bounded by a quadratic Bézier surface triangle*

The integration scheme is verified by integrating over the region bounded by the quadratic polynomial Bézier surface triangle pictured in Fig. 9 and the planes $x = 0$, $y = 0$, and $z = 0$. The points in the control net are

$$\begin{gathered} \boldsymbol{p}_{200} = [1, 0, 0]^T , \quad \boldsymbol{p}_{110} = [0.2, 0.2, 0]^T , \quad \boldsymbol{p}_{101} = [0.2, 0, 0.2]^T , \\ \boldsymbol{p}_{020} = [0, 1, 0]^T , \quad \boldsymbol{p}_{011} = [0, 0.2, 0.2]^T , \quad \boldsymbol{p}_{002} = [0, 0, 1]^T . \end{gathered} \tag{42}$$

Since the Bézier surface is polynomial, exact numerical integration of monomials using (41) is possible, provided a cubature rule of sufficient accuracy is available. For monomials up to degree 2, on a quadratic Bézier surface triangle, a seventh-order Dunavant (1985) rule gives the needed number of cubature points. The thirteen cubature points for the Dunavant rule with $p = 7$ are plotted on the Bézier surface in Fig. 9. For constant and linear monomials, Dunavant cubature rules with $p = 3$ and $p = 5$, respectively, are needed for exact integration. These correspond to 4- and 7-point cubature rules. Using these integration rules, machine precision accuracy, $\mathcal{O}(10^{-15})$, is obtained.

### *4.1.2. Numerical example: Integration over a sphere*

A quartic rational Bézier surface can exactly represent the octant of a sphere (Farin et al., 1988), as well as any other quadric surface (Farin, 2002). As a second verification problem, we use (41) to integrate monomials up to degree 2 over the first octant of a unit sphere. Numerical integration of (41) is performed using a Dunavant (1985) integration rule, which is not exact in this case, since the integrand is rational. However, cubature convergence can be investigated by increasing the order of the Dunavant rule and plotting the resulting integration error. This is demonstrated for all monomials up to degree 2 in Fig. 10. Though ten different monomials are integrated, only three distinct convergence curves are obtained. These three curves are plotted in Fig. 10. In general, integration error reduces exponentially with increasing cubature points. At least nine digits of accuracy are achieved for all integrals with 61 cubature points.

## *4.2. Region bounded by B-spline surface patches*

Non-uniform rational B-spline (NURBS) surfaces are among the most popular surface descriptions in CAD applications. NURBS surfaces are based on B-spline basis functions, which are

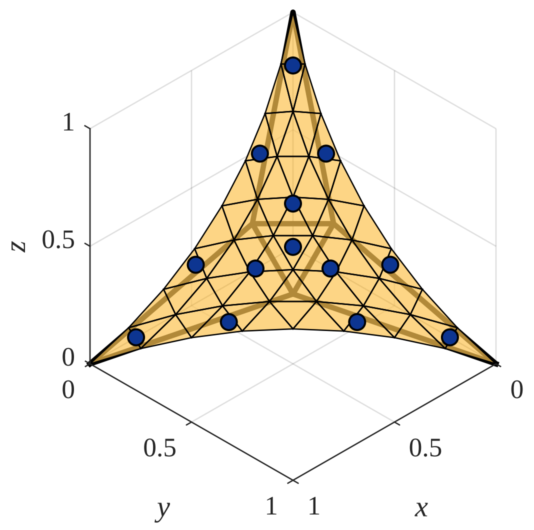


Figure 9: A quadratic non-rational Bézier surface triangle and its control net. Cubature points corresponding to a seventh-order Dunavant (1985) cubature rule are shown as filled circles on the surface.

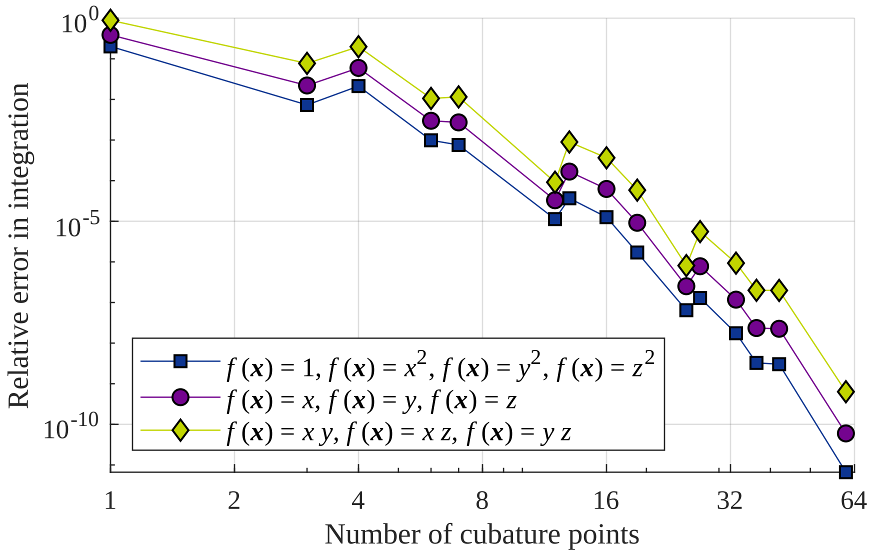


Figure 10: Relative integration error versus number of cubature points in the Dunavant (1985) rule for integrating monomials over the first octant of a unit sphere.

defined by the convex hull property, the compact support property, and $C^{n-1}$-smoothness for a degree-$n$ curve (Farouki, 2008). A B-spline basis function is defined recursively via the expression

$$B_k^r(t) = \frac{t - t_k}{t_{k+r} - t_k} B_k^{r-1}(t) + \frac{t_{k+r+1} - t}{t_{k+r+1} - t_{k+1}} B_{k+1}^{r-1}(t), \tag{43}$$

which is initiated with

$$B_k^0(t) = \begin{cases} 1 & \text{if } t_k \le t < t_{k+1} \\ 0 & \text{otherwise} \end{cases} \tag{44}$$

and continues until $r = n$. In (43), $t_k$ represents the location of the $k$-th knot in the knot vector. Values of $t_k$ are non-decreasing, such that $t_{k-1} \le t_k \le t_{k+1}$. The knot vector defines spacing between control points in parametric space and a total of $N + n + 2$ knots are required with $N + 1$ control points. Repeating the same point in the knot vector $r$ times (i.e., $t_k = t_{k+1} = \ldots = t_{k+r}$) reduces the continuity in $B_k^n(t)$ to $C^{n-r}$ at $t_k$ in all basis functions with support at this point. B-spline basis functions that interpolate the first and last control points can be generated with a knot vector with $n + 1$-fold multiplicity of values at the beginning and end of the knot vector. All B-splines in the examples that follow will be of this type.

A B-spline surface can be generated using a tensor-product construction of basis functions:

$$\boldsymbol{b}(u, v) = \sum_{i=0}^{M} \sum_{i=0}^{N} \boldsymbol{p}_{ij} B_i^m(u) B_j^n(v), \tag{45}$$

where the B-spline basis in the $u$-direction is degree-$m$, and in the $v$-direction the B-spline basis is degree-$n$. Also, in (45), $\boldsymbol{p}_{ij}$ defines the $(M + 1)(N + 1)$ control points of the surface. Similar to the rational extension to the Bézier surface triangle, a NURBS surface is given by

$$\boldsymbol{b}(u, v) = \frac{\sum_{i=0}^{M} \sum_{i=0}^{N} w_{ij} \boldsymbol{p}_{ij} B_i^m(u) B_j^n(v)}{\sum_{i=0}^{M} \sum_{i=0}^{N} w_{ij} B_i^m(u) B_j^n(v)}, \tag{46}$$

where weights have been introduced at each of the control points. To compute points on a B-spline surface, the numerically stable Cox-de Boor algorithm can be used (Cox, 1972; de Boor, 1972).

Starting with (7), we wish to parameterize the surface in $\boldsymbol{u}$-coordinates where the region of integration is a rectangle. Assuming the entire surface is covered with B-spline patches and introducing the Jacobian of the transformation, (7) becomes

$$\int_R f(\boldsymbol{x})\, d\boldsymbol{x} = \frac{1}{d+q} \sum_{i=1}^{m} \int_{\square} (\boldsymbol{b}_i \cdot \boldsymbol{n}(\boldsymbol{b}_i)) f(\boldsymbol{b}_i) \frac{\left\| \frac{\partial \boldsymbol{b}_i}{\partial u} \times \frac{\partial \boldsymbol{b}_i}{\partial v} \right\|}{(u_{N+n+i} - u_0)(v_{N+n+i} - v_0)}\, d\boldsymbol{u}. \tag{47}$$

The unit normal of the B-spline patch as a function of $\boldsymbol{u}$ is given by

$$\boldsymbol{n}(\boldsymbol{b}_i) = \left( \frac{(u_{N+n+i} - u_0)(v_{N+n+i} - v_0)}{\left\| \frac{\partial \boldsymbol{b}_i}{\partial u} \times \frac{\partial \boldsymbol{b}_i}{\partial v} \right\|} \right) \frac{\partial \boldsymbol{b}_i}{\partial u} \times \frac{\partial \boldsymbol{b}_i}{\partial v}. \tag{48}$$

Substituting (48) into (47) yields the final result:

$$\int_R f(\boldsymbol{x})\, d\boldsymbol{x} = \frac{1}{d+q} \sum_{i=1}^{m} \int_{\Box} \boldsymbol{b}_i \cdot \left( \frac{\partial \boldsymbol{b}_i}{\partial u} \times \frac{\partial \boldsymbol{b}_i}{\partial v} \right) f(\boldsymbol{b}_i)\, d\boldsymbol{u}. \tag{49}$$

A tensor-product Gauss cubature rule can be used to integrate (49). In the $u$-direction (resp. the $v$-direction), the highest order polynomial term when integrating a degree-2 monomial on a B-spline surface is $3m + 2(m-1)$ (resp. $3n + 2(n-1)$). Using these values, the number of cubature points can be optimized. A tensor-product Gauss rule can also be used to efficiently integrate NURBS surfaces, though the rule is not exact. Though this section specifically discusses B-spline surfaces, we note that (49) can be applied to any tensor-product surface, such as a bicubic Hermite patch and tensor-product spline surfaces in the cardinal basis.

Note when $\boldsymbol{b}_i$ is a polynomial B-spline surface, (49) contains a polynomial integrand, which can be decomposed into a summation of homogeneous polynomials. Therefore, (11) can be applied again to transform integration to the boundary of the rectangular patch. This idea can be used to integrate over trimmed B-spline patches.

#### *4.2.1. Numerical example: Region bounded by a B-spline surface*

Monomials of degree 2 and less are integrated over a region bounded by a bi-quadratic non-rational B-spline surface and planar facets using (49). Since a B-spline surface is polynomial, the integrand in (49) is polynomial and, therefore, integration is exact given a tensor-product Gauss cubature rule with a sufficient number of points. Given a bi-quadratic patch, polynomial integrands of order 4, 6, and 8 are expected in each direction for constant, linear, and quadratic monomials, respectively. This requires Gauss rules of order 3, 4, and 5, respectively, in each direction. With additional integration points, the computed value of the integral is expected to remain the same.

The B-spline patch used in this example is defined by the knot vectors

$$u_{\text{knot}} = \left\{0, 0, 0, \frac{1}{2}, 1, 1, 1\right\}, \qquad v_{\text{knot}} = \{0, 0, 0, 1, 1, 1\}, \tag{50}$$

and the control points

$$\begin{aligned}
\boldsymbol{p}_{00} &= [-1.5, -1.0, 0.7]^T, & \boldsymbol{p}_{01} &= [-1.3, 0.0, 0.3]^T, & \boldsymbol{p}_{02} &= [-1.5, 1.0, 0.3]^T,\\
\boldsymbol{p}_{10} &= [-0.5, -0.8, 0.4]^T, & \boldsymbol{p}_{11} &= [-0.5, 0.0, 0.7]^T, & \boldsymbol{p}_{12} &= [-0.5, 0.8, 0.3]^T,\\
\boldsymbol{p}_{20} &= [0.5, -1.0, 0.7]^T, & \boldsymbol{p}_{21} &= [0.3, 0.0, 0.4]^T, & \boldsymbol{p}_{22} &= [0.0, 0.5, 1.0]^T,\\
\boldsymbol{p}_{30} &= [1.5, -1.0, 0.7]^T, & \boldsymbol{p}_{31} &= [1.5, 0.0, 0.4]^T, & \boldsymbol{p}_{32} &= [1.5, 0.5, 1.0]^T.
\end{aligned} \tag{51}$$

The B-spline surface and its control net are presented in Fig. 11. The monomials 1, $x$, and $y^2$, are integrated over the region bounded by the surface using (49). Tensor-product Gauss rules ranging from $1\times1$ point to $6\times6$ points over every knot interval are used to numerically integrate (49). The results are listed in Table 1. As expected, the integrals become constant with $3\times3$, $4\times4$, and $5\times5$ Gauss rules for constant, linear, and quadratic monomials.

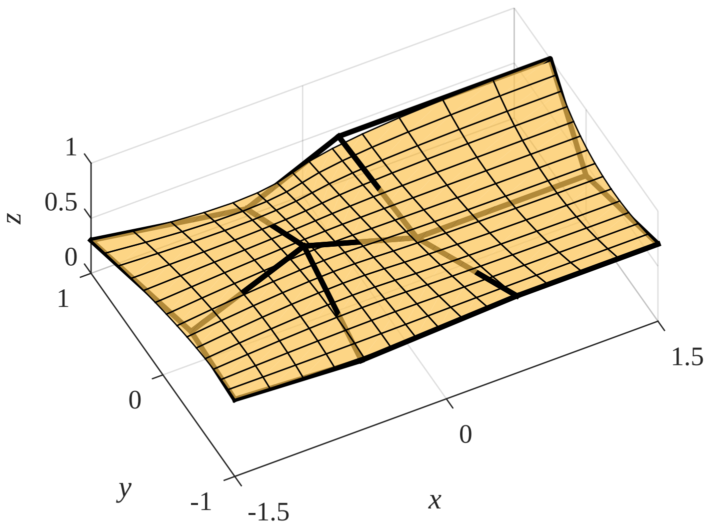


Figure 11: A quadratic non-rational B-spline surface and its control net.

Table 1: Results from numerically integrating monomials over the region bounded by the B-spline patch illustrated in Fig. 11 using a tensor-product Gauss rule with (49).

| Gauss points | $\int_R d\boldsymbol{x}$ | $\int_R x\, d\boldsymbol{x}$ | $\int_R y^2\, d\boldsymbol{x}$ |
|---|---|---|---|
| 1 | 1.379820312500000 | −0.031140106201172 | 0.005417441940308 |
| 2 | 1.279365740740741 | −0.017929898485725 | 0.185262011702675 |
| 3 | 1.279552083333333 | −0.014459059453125 | 0.172001609802917 |
| 4 | 1.279552083333333 | −0.014473543367347 | 0.172069310793788 |
| 5 | 1.279552083333333 | −0.014473543367347 | 0.172069067015660 |
| 6 | 1.279552083333333 | −0.014473543367347 | 0.172069067015660 |

#### 4.2.2. *Numerical example: Integration over a torus*

In this numerical example, monomials in the triquadratic basis are integrated over a torus defined by a circle of radius $a = 1$ swept around a circular path of radius $R = 3$. This geometry (and the NURBS control net) is presented in Fig. 12. The control points and weights for a NURBS patch of a torus are given in Hughes et al. (2005). Exact integrals can be calculated by introducing the change of coordinates,

$$x = \cos\varphi\,(R + r\cos\theta), \tag{52a}$$
$$y = \sin\varphi\,(R + r\cos\theta), \tag{52b}$$
$$z = r\sin\theta, \tag{52c}$$

where $(r, \theta)$ are polar coordinates with the origin located at the center of the circle with radius $a$ and $\varphi$ is the angle of revolution about the $xy$-plane centered at the origin. The values of the non-zero

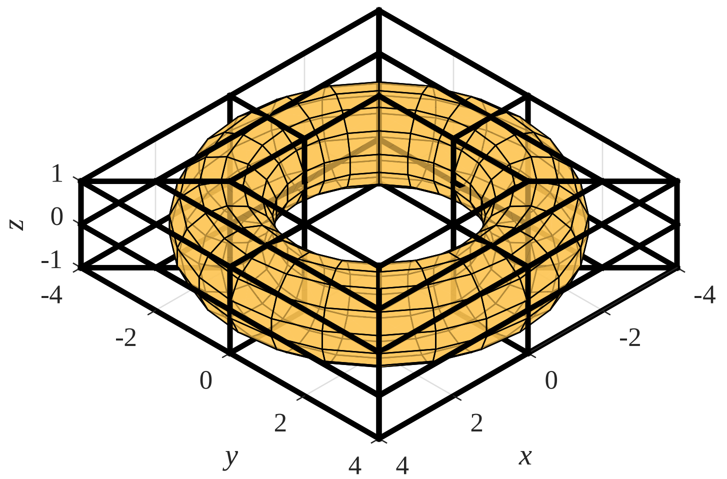


Figure 12: A torus represented as a NURBS surface and its control net.

integrals are:

$$
\begin{aligned}
\int_R d\boldsymbol{x} &= 2a^2\pi^2 R = 6\pi^2, \\
\int_R x^2 d\boldsymbol{x} = \int_R y^2 d\boldsymbol{x} &= \frac{a^2\pi^2}{4}(3a^2R + 4R^3) = \frac{117}{4}\pi^2, \\
\int_R z^2 d\boldsymbol{x} &= \frac{a^4\pi^2 R}{2} = \frac{3}{2}\pi^2, \\
\int_R x^2 y^2 d\boldsymbol{x} &= \frac{a^2\pi^2 R}{32}\left(5a^4 + 20a^2R^2 + 8R^4\right) = \frac{2499}{32}\pi^2, \\
\int_R x^2 z^2 d\boldsymbol{x} = \int_R y^2 z^2 d\boldsymbol{x} &= \frac{a^4\pi^2 R}{8}\left(a^2 + 2R^2\right) = \frac{57}{8}\pi^2, \\
\int_R x^2 y^2 z^2 d\boldsymbol{x} &= \frac{a^4\pi^2 R}{768}\left(15a^4 + 80a^2R^2 + 48R^4\right) = \frac{4623}{256}\pi^2.
\end{aligned}
$$

Using a tensor-product Gauss cubature rule with (49), the quadratic NURBS representation of the boundary of the torus is used to numerically integrate monomials. The NURBS description of the torus contains four non-zero spans in the knot vector in both the $u$- and $v$-directions, resulting in 16 total regions of integration. Gauss cubature rules ranging from $1 \times 1$ point to $15 \times 15$ points are used to compute the integral in (49) over each of the 16 integration regions (producing a total of $1 \times 1 \times 16 = 16$ to $15 \times 15 \times 16 = 3600$ number of cubature points). Monomials integrated include those in the triquadratic basis. The relative error in integration versus the total number of cubature points on the torus is plotted in Fig. 13 for unique and nontrivial monomial integrands. Using a $14 \times 14$ rule in each region, the integration error is $O(10^{-15})$ for all integrals.

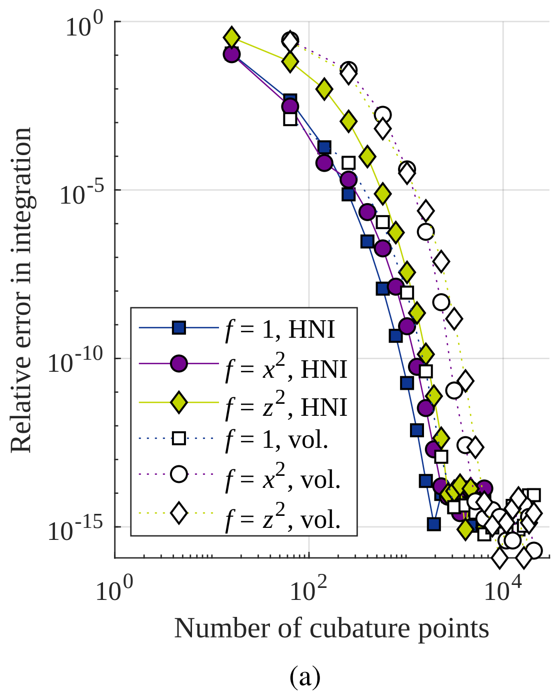
As a measure of comparison, Gauss cubature is also directly applied to the volume integrals defined in the coordinates given in (52). Volume integration is simplified using (52), which provides an exact parametric description of the geometry. In most instances, an exact parametric mapping is not known for arbitrary NURBS surfaces, so a comparison of this type would not be feasible. While Gauss cubature can approximate the volume integral with just a single region of integration, we choose to subdivide the torus into 16 regions to provide a more fair comparison to integration using (49). The 16 regions are formed as Cartesian products of the intervals $\theta = [0, \pi/4], [\pi/4, \pi], [\pi, 4\pi/3]$, and $[4\pi/3, 2\pi]$ and $\varphi = [0, \pi/3], [\pi/3, \pi], [\pi, 5\pi/4]$, and $[5\pi/4, 2\pi]$. Non-symmetric regions of integration are selected to avoid error cancellation, which artificially produces machine-precision integration accuracy of non-polynomial integrands. Tensor-product rules ranging from 64 points ($1\times1\times4\times16$) to 12,544 points ($14\times14\times4\times16$) are used to integrate the monomials in the triquadratic basis. The number of cubature points in the $\theta$ and $\varphi$ directions are varied, while the number of cubature points in the $r$ direction is held constant at 4, which is sufficient to integrate the monomials in the triquadratic basis. The results of the volume integrals are presented in Fig. 13. Reducing the relative error in integration below $O(10^{-14})$ using (49) requires approximately $2\times$ to $4\times$ fewer cubature points as compared to the parameterized volume integral.

Exponential rates of convergence are observed with both methods. Using (49), this is due to the cubature rule essentially being two-dimensional (in a three-dimensional integration domain). With the parameterized volume integral, convergence rates comparable to HNI are obtained because the transformed integral is polynomial in the $r$-direction. Therefore, machine precision integration is obtained with only a few cubature points in the $r$ direction, and integration error above machine precision is only in the $\theta$ and $\varphi$ directions. Importantly, exponential convergence in integration error is expected for any region bounded by NURBS patches when using (49) to integrate, while commensurate rates of convergence are the exception for the tensor-product Gauss volume integration scheme.

## 5. Concluding remarks

In this paper, we introduced the HNI method for integrating monomials over polyhedra and regions bounded by curved, parametric surfaces. Several numerical examples were presented that demonstrated the accuracy and efficiency of the method for the integration of monomials over a large collection of convex and nonconvex polyhedra, and regions bounded by curved surfaces that map to barycentric triangles and rectangles defined by tensor-product patches. Monomial and polynomial integration over these domains is a fundamental requirement in applications in both CAGD and in emerging methods for numerically solving PDEs. Existing methods for computing these quantities either require some form of approximation or require non-trivial definition of antiderivatives of the integrands. The method introduced in this paper does not introduce approximation of the integrals and does not require defining antiderivatives. For affine surfaces and curved surfaces defined by parametric polynomials, numerical integration is exact, given an appropriate cubature rule. For curved surfaces defined by rational parameterizations, cubature convergence is exponential.

For computing monomials over polyhedra, two algorithms were presented. The first efficiently integrates monomials up to a given order and the second provides exact, simplified formulas for



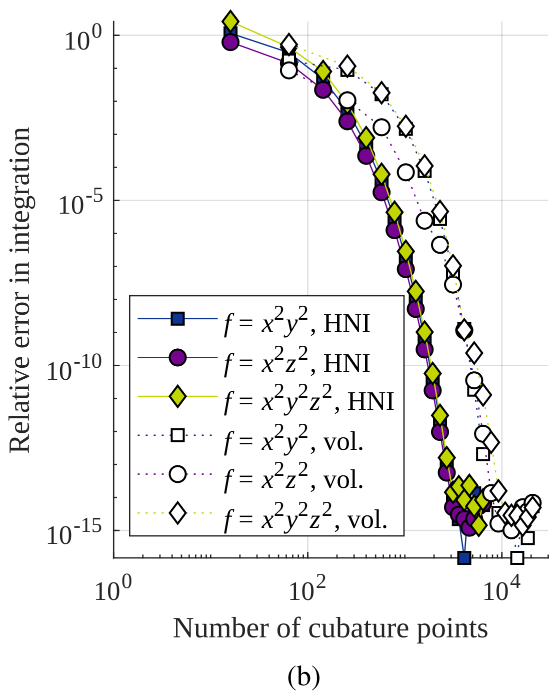


Figure 13: Relative integration error versus number of cubature points for integrating monomials over the torus in Fig. 12 using a tensor-product Gauss rule with (49) (filled markers, solid line, labeled HNI) and using tensor-product Gauss volume integration using the parametric description of a torus in (52) (open markers, dotted line, labeled vol). (a) Constant and quadratic integrands and (b) fourth-order and sixth-order integrands.

integrating monomials up to degree 2, which are required to compute polyhedral mass properties. In the first algorithm, efficiency is realized by applying the integrals of lower order monomials to eliminate the need to integrate partial derivatives of the monomial, as required by the method when reducing integration to the edges and vertices of the polyhedron. Further efficiency gains are the result of storing integrals on the edges and vertex values, such that these quantities do not need to be computed multiple times. In the second algorithm, explicit formulas were derived for integrating up to quadratic monomials over a polyhedron. With these formulas, all monomial integrals are computed by looping once over the faces of the polyhedron and the vertices of each face. C++11 codes that implement both of these algorithms are provided in the supplementary material.

While this paper has introduced an improved numerical integration method for geometry described by curved surfaces, further research is needed to apply these methods to more elaborate surface descriptions. For example, trimming is an important operation on NURBS patches, but presents difficulties in applying the HNI method, due to the integrands on the surface being rational. In this case, the HNI method does reduce integration to the boundary of the domain, though tensor-product cubature on the surface fails since the region of integration is no longer rectangular. Marussig and Hughes (2018) summarize several approaches to deal with trimmed patches, such as reconstructing the surface using Bézier triangles. An additional potential direction for future research is in coupling these ideas to level set methods and implicit descriptions of geometry. The HNI method requires an explicit parametric surface description to compute integrals, and hence developing robust methods to generate a parametric representation of a surface from an implicit surface is worthy of pursuit.

## Disclaimer

## Acknowledgements

The authors gratefully acknowledge the research support of Sandia National Laboratories to the University of California at Davis. Additional financial support to EBC from the ARCS Foundation Northern California is also acknowledged.

## Appendix A. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cagd.2020.101914.

## References

Al-Daccak, M., Angeles, J., 1993. The calculation of the volumetric properties of sweep-generated solids via line integration. Journal of Mechanical Design 115, 110–118.

Antonietti, P. F., Houston, P., Pennesi, G., 2018. Fast numerical integration on polytopic meshes with applications to discontinuous Galerkin finite element methods. Journal of Scientific Computing 77 (3), 1339–1370.

Baldoni, V., Berline, N., De Loera, J. A., Dutra, B., Köppe, M., Moreinis, S., Pinto, G., Vergne, M., Wu, J., October 2014. A User's Guide for `LattE integrale` v1.7.2. Department of Mathematics, University of California, Davis, CA 95616, available at http://www.math.ucdavis.edu/~latte.

Bassi, F., Botti, L., Colombo, A., Petro, D. A. D., Tesini, P., 2012. On the flexibility of agglomeration based physical space discontinuous Galerkin discretizations. Journal of Computational Physics 231 (1), 45–65.

Beirão da Veiga, L., Brezzi, F., Cangiani, A., Manzini, G., Marini, L. D., Russo, A., 2013. Basic principles of virtual element methods. Mathematical Models and Methods in Applied Sciences 23, 199.

Beirão da Veiga, L., Lipnikov, K., Manzini, G., 2014. The Mimetic Finite Difference Method for Elliptic Problems. Springer-Verlag, New York.

Benvenuti, E., Chiozzi, A., Manzini, G., Sukumar, N., 2019. Extended virtual element method for the Laplace problem with singularities and discontinuities. Computer Methods in Applied Mechanics and Engineering 356, 571–597.

Cangiani, A., Georgoulis, E. H., Houston, P., 2014. *hp*-version discontinuous Galerkin methods on polygonal and polyhedral meshes. Mathematical Models and Methods in Applied Sciences 24 (10), 2009–2041.

Chin, E. B., Lasserre, J. B., Sukumar, N., 2015. Numerical integration of homogeneous functions on convex and nonconvex polygons and polyhedra. Computational Mechanics 56 (6), 967–981.

Chin, E. B., Lasserre, J. B., Sukumar, N., 2017. Modeling crack discontinuities without element-partitioning in the extended finite element method. International Journal for Numerical Methods in Engineering 110 (11), 1021–1048.

Chin, E. B., Sukumar, N., 2019. Modeling curved interfaces without element-partitioning in the extended finite element method. International Journal for Numerical Methods in Engineering 120 (5), 607–649.

Cox, M. G., 1972. The numerical evaluation of B-splines. IMA Journal of Applied Mathematics 10 (2), 134–149.

de Almeida, Y. S. J. P. M., Wall, W. A., 2014. An accurate, robust, and easy-to-implement method for integration over arbitrary polyhedra: Application to embedded interface methods. Journal of Computational Physics 273, 393–415.

de Boor, C., 1972. On calculating with B-splines. Journal of Approximation Theory 6 (1), 50–62.

De Loera, J. A., Dutra, B., Köppe, M., Moreinis, S., Pinto, G., Wu, J., 2013. Software for exact integration of polynomials over polyhedra. Computational Geometry 46 (3), 232–252.

Dunavant, D. A., 1985. High degree efficient symmetrical Gaussian quadrature rules for the triangle. International Journal for Numerical Methods in Engineering 21 (6), 1129–1148.

Düster, A., Parvizian, J., Yang, Z., Rank, E., 2008. The finite cell method for three-dimensional problems of solid mechanics. Computer Methods in Applied Mechanics and Engineering 197 (45–48), 3768–3782.

Farin, G., 1986. Triangular Bernstein-Bézier patches. Computer Aided Geometric Design 3, 83–127.

Farin, G., 2002. Curves and Surfaces for CAGD: A Practical Guide, 5th Edition. Morgan Kaufmann Publishers, San Francisco, CA.

Farin, G., Piper, B., Worsey, A., 1988. The octant of a sphere as a non-degenerate triangular Bézier surface patch. Computer Aided Geometric Design 4 (4), 329–332.

Farouki, R. T., 2008. Pythagorean-Hodograph Curves: Algebra and Geometry Inseparable. Springer, Berlin, Germany.

Fries, T. P., Belytschko, T., 2010. The extended/generalized finite element method: An overview of the method and its applications. International Journal for Numerical Methods in Engineering 84 (3), 253–304.

Guendelman, E., Bridson, R., Fedkiw, R., 2003. Nonconvex rigid bodies with stacking. ACM Transactions on Graphics 22 (3), 871–878.

Hughes, T. J. R., Cottrell, J. A., Bazilevs, Y., 2005. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. Computer Methods in Applied Mechanics and Engineering 194 (39–41), 4135–4195.

Krishnamurthy, A., McMains, S., 2011. Accurate GPU-accelerated surface integrals for moment computation. Computer Aided Design 43, 1284–1295.

Lasserre, J. B., 1998. Integration on a convex polytope. Proceedings of the American Mathematical Society 126 (8), 2433–2441.

Lasserre, J. B., 1999. Integration and homogeneous functions. Proceedings of the American Mathematical Society 127 (3), 813–818.

Lee, Y. T., Requicha, A. A. G., 1982. Algorithms for computing the volume and other integral properties of solids. II. A family of algorithms based on representation conversion and cellular approximation. Communications of the ACM 25 (9), 642–650.

Legay, A., Wang, H. W., Belytschko, T., 2005. Strong and weak arbitrary discontinuities in spectral finite elements. International Journal for Numerical Methods in Engineering 64, 991–1008.

Lipnikov, K., Morgan, N., 2019. A high-order discontinuous Galerkin method for level set problems on polygonal meshes. Journal of Computational Physics 397, 108834.

Marussig, B., Hughes, T. J. R., 2018. A review of trimming in isogeometric analysis: Challenges, data exchange and simulation aspects. Archives of Computational Methods in Engineering 25, 1059–1127.

Messner, A. M., Taylor, G. Q., 1980. Algorithm 550: Solid polyhedron measures [Z]. ACM Transactions on Mathematical Software 6 (1), 121–130.

Min, C., Gibou, F., 2007. Geometric integration over irregular domains with application to level-set methods. Journal of Computational Physics 226, 1432–1443.

Mirtich, B., 1996. Fast and accurate computation of polyhedral mass properties. Journal of Graphics Tools 1 (2), 31–50.

Moës, N., Dolbow, J., Belytschko, T., 1999. A finite element method for crack growth without remeshing. International Journal for Numerical Methods in Engineering 46, 131–150.

Schillinger, D., Reuss, M., 2015. The finite cell method: A review in the context of higher-order structural analysis of CAD and image-based geometric models. Archives of Computational Methods in Engineering 22 (3), 391–455.

Sudhakar, Y., Wall, W. A., 2013. Quadrature schemes for arbitrary convex/concave volumes and integration of weak form in enriched partition of unity methods. Computer Methods in Applied Mechanics and Engineering 258, 39–54.

Sukumar, N., Dolbow, J. E., Moës, N., 2015. Extended finite element method in computational fracture mechanics: a retrospective examination. International Journal of Fracture 196 (1–2), 189–206.

Taylor, M. E., 1996. Partial Differential Equations: Basic Theory. Springer-Verlag, New York.

Thomson, W., 1887. On the division of space with minimal partitional area. Acta Mathematica 11, 121–134.

Timmer, H. G., Stern, J. M., 1980. Computation of global geometric properties of solid objects. Computer-Aided Design 12 (6), 301–304.

Weaire, D., Phelan, R., 1994. A counter-example to Kelvin's conjecture on minimal surfaces. Philosophical Magazine Letters 69 (2), 107–110.