SIAM J. SCI. COMPUT.
Vol. 39, No. 3, pp. B571–B587

ON THE USE OF COMPRESSED POLYHEDRAL QUADRATURE FORMULAS IN EMBEDDED INTERFACE METHODS[*]

Y. SUDHAKAR[†], ALVISE SOMMARIVA[‡], MARCO VIANELLO[‡], AND WOLFGANG A. WALL[§]

Abstract. The main idea of this paper is to apply a recent quadrature compression technique to algebraic quadrature formulas on complex polyhedra. The quadrature compression substantially reduces the number of integration points but preserves the accuracy of integration. The compression is easy to achieve since it is entirely based on the fundamental methods of numerical linear algebra. The resulting compressed formulas are applied in an embedded interface method to integrate the weak form of the Navier–Stokes equations. Simulations of flow past stationary and moving interface problems demonstrate that the compressed quadratures preserve accuracy and rate of convergence and improve the efficiency of performing the weak form integration, while preserving accuracy and order of convergence.

Key words. quadrature compression, algebraic quadrature, complex polyhedra, embedded interface methods

AMS subject classifications. 65D32, 65M60

DOI. 10.1137/16M1085206

1. Introduction.

Accurate numerical integration over arbitrary complex polyhedra is an essential component in various classes of finite element methods (FEM): embedded interface methods (EIM), polyhedral FEM, boundary element methods, and virtual element methods, to name a few. In EIMs, the interface, which is embedded within the nonbody confirming finite element mesh, cuts several elements of the background mesh, and as a result, splits these elements into two or more nonoverlapping polyhedra. The elements that are cut by the interface are called cut-elements, and the polyhedra produced by the interface cut are referred to as volume-cells. These cut-elements offer significant difficulties—in order to extract the stiffness matrix for these elements, the weak form has to be integrated over the arbitrary polyhedral shaped volume-cells. Moreover, in many problem classes, one faces strict requirements for such integrations.

Accuracy of the weak form integration over the volume-cells significantly influences the overall solution accuracy in EIMs. In certain simulations, a small error introduced in the integration can even hinder the overall convergence behavior. The inapplicability of the standard Gauss quadrature rules for a general polyhedron, together with the importance of accurate weak form integration, have motivated researchers to develop various methods for numerical integration over polyhedra [6, 21, 23, 24, 25, 36, 37, 42]. (Refer to [36] for an overview of the most important methods.)

[*]Submitted to the journal's Computational Methods in Science and Engineering section July 8, 2016; accepted for publication (in revised form) May 8, 2017; published electronically June 27, 2017.
http://www.siam.org/journals/sisc/39-3/M108520.html

Funding: The second and third authors were partially supported by the BIRD163015, CPDA143275, and DOR funds of the University of Padova and by the GNCS-INdAM.

[†]Institute for Computational Mechanics, Technical University of Munich, Germany. Current address: Linné Flow Centre, Department of Mechanics, KTH Royal Institute of Technology, Sweden (sudhakar.yogaraj@mech.kth.se).

[‡]Department of Pure and Applied Mathematics, University of Padova, Italy (alvise@math.unipd.it, marcov@math.unipd.it).

[§]Institute for Computational Mechanics, Technical University of Munich, Germany (wall@lnm.mw.tum.de, www.lnm.mw.tum.de).

One of the major drawbacks of the methods developed to perform integration over polyhedra is the large number of integration points they yield. Often we get thousands of integration points to achieve an accurate integral value [36, 37], and this leads to drastic computational overheads. Numerical integration on the cut-elements is a key factor that tremendously slows down the execution speed of EIMs. As an attempt to improve the computational efficiency of such methods, in this paper, we apply the recently developed multivariate quadrature compression techniques [3, 34, 35] to reduce the number of integration points required for the polyhedral quadrature rules.

Quadrature compression techniques operate on an existing quadrature rule by selecting a subset of nodes and modify the corresponding weights in such a way that the accuracy of the original quadrature rule is preserved. In the compressed quadrature rule, the number of integration points is always equal to the dimension of the considered polynomial space. For example, in three dimensions, the cardinality of fifth-order basis functions is 56. This means that the compressed quadrature rule will have 56 integration points, *irrespective of the shape of the polyhedra*, to integrate any fifth-order polynomial.

Multivariate quadrature compression is a fairly new area of research in applied mathematics. The existing studies, performed until now, report quadrature compression only on two-dimensional geometries [3, 34, 35]. In this work, for the first time, we carry out the quadrature compression over arbitrary complex polyhedra in three dimensions, and use the resulting quadrature schemes to integrate the weak form of the EIMs. The embedded interfaces are represented as straight line segments, i.e., interface curvature is not treated in this work.

This paper is organized as follows. The theory of multivariate quadrature compression and a practical method to perform compression together with the results of applying the method to arbitrary polyhedra are presented in section 2. In section 3, the application of compressed quadrature rules to perform weak form integration of EIMs is presented, with emphasis on accuracy, rate of convergence, and improved computational efficiency.

2. Compression of quadrature formulas.

Multivariate algebraic quadrature formulas on complicated geometries have often a huge number of nodes, much larger than the dimension of the exactness polynomial space [36, 37]. The possibility of compressing a multivariate quadrature formula by node selection and reweighting rests in principle on the well-known Tchakaloff's theorem, a deep result of quadrature theory, which ensures existence of positive algebraic formulas of low cardinality. Such a theorem was originally stated and proved by Tchakaloff [39] for compactly supported absolutely continuous measures with respect to the Lebesgue measure and afterward generalized to arbitrary measures with finite moments, even discrete measures (cf., e.g., [27, Theorem 1]). We report a fully discrete version of the theorem, relevant for the compression of quadrature formulas considered as discrete measures (compression of weighted sums).

THEOREM 2.1. *Consider a multivariate discrete measure supported at $X = \{x_i\} \subset \mathbb{R}^d$ with positive masses (weights) $\boldsymbol{w} = \{w_i\}$, $1 \le i \le M$, and let n be a fixed positive integer. Then there are $m \le N = \dim(\mathbb{P}_n^d)$ points $\{x_{i_s}\} \subseteq X$ and positive real numbers $\boldsymbol{\lambda} = \{\lambda_s\}$, $1 \le s \le m$, such that*

$$\sum_{i=1}^{M} w_i \, p(x_i) = \sum_{s=1}^{m} \lambda_s \, p(x_{i_s}) \ \ \forall p \in \mathbb{P}_n^d \, . \tag{1}$$

Now, consider an algebraic quadrature formula for the Lebesgue measure on a compact set $K \subset \mathbb{R}^d$,

$$\mathcal{Q}_{BQ} = (X, \boldsymbol{w}) \ , \ \ \mathrm{card}(X) = M > N = \dim\left(\mathbb{P}^d_n\right) \ , \tag{2}$$

with polynomial degree of exactness n (that we shall term in what follows the "base quadrature rule"). By the discrete version above of Tchakaloff's theorem, we get

$$\int_K p(x)\,dx = \sum_{i=1}^{M} w_i\, p(x_i) = \sum_{s=1}^{m} \lambda_s\, p(x_{i_s}) \ \ \forall p \in \mathbb{P}^d_n \ , \tag{3}$$

with $m \leq N$, that is the base quadrature rule can be compressed preserving the degree of polynomial exactness.

On the other hand, Tchakaloff's theorem is an existence result that does not provide directly a constructive approach for the node subset selection and reweighting. Nevertheless, for this purpose we can use the following approach based on discrete moments. Given a polynomial basis $\boldsymbol{\tau}(x) = (\tau_1(x), \ldots, \tau_N(x))$ of $\mathbb{P}^d_n$, we can consider the underdetermined system

$$T\boldsymbol{u} = \boldsymbol{\nu} \ , \ \ T = (t_{sk}) = (\tau_s(x_k)) \in \mathbb{R}^{N\times M} \ , \tag{4}$$

$$\boldsymbol{u} = \{u_k\} \in \mathbb{R}^M \ , \ \ \boldsymbol{\nu} = \{\nu_s\} = T\boldsymbol{w} \in \mathbb{R}^N \ , \ \ \boldsymbol{w} = \{w_k\} \in \mathbb{R}^M \ ,$$

where $\boldsymbol{\nu}$ is the vector of discrete moments of the basis, and seek a sparse nonnegative solution, i.e., a nonnegative solution vector with (at least) $M - N$ null components. In order to have a not too ill-conditioned matrix T (an unavoidable fact with the standard monomial basis), we can start from the total-degree product Chebyshev basis (with graded lexicographic order, cf. [10]) of the minimal Cartesian rectangle containing the nodes and possibly perform a discrete orthonormalization of the basis by the QR algorithm applied to the Vandermonde-like matrix T^t (the superscript t denotes matrix transpose); see [35] for an implementation.

Looking for positive weights, sparsity can be achieved by reformulating equation (4) as the nonnegative least squares (NNLS) problem $\min_{\boldsymbol{u}\geq \boldsymbol{0}} \|T\boldsymbol{u} - \boldsymbol{\nu}\|_2$ and solving it by some suitable quadratic programming method, such as the active set method by Lawson and Hanson [17]. The nonzero components of the NNLS solution, say, $\boldsymbol{u}^*$, give a subset of indexes $(i_1, \ldots, i_m)$ that allow one to compress the formula by selecting a subset of nodes $\{x_{i_s}\}$ with positive weights $\{w_s\} = \{u^*_{i_s}\}$. See [14] for the univariate case and [35] for the multivariate extension.

On large problems, however, the NNLS approach has a high computational cost, and as an alternative we can seek a classical solution to the underdetermined system (with $M - N$ zero components), corresponding to a Fekete-like extremal subset of X. This approach consists in selecting a subset of N colums of T, trying to maximize the (absolute value of) the determinant of the resulting square submatrix. It is known that such a maximization is an NP-hard problem [7], which requires heuristic algorithms. One is computing the well-known QR factorization with column pivoting of T, cf. [4], which corresponds to a greedy maximization of submatrix "volumes" (a concept that can be easily understood thinking to the extraction of three vectors from a bunch of three-dimensional (3D) vectors to maximize the volume of the resulting parallelepiped). The computed subset of indexes $(i_1, \ldots, i_N)$ selects from the original nodes the so-called approximate Fekete points; cf. [34].

The approach adopted in this paper works instead on a greedy maximization of the subdeterminants of the Vandermonde-like matrix T^t by its LU factorization with row pivoting, obtained by the resulting subset of indexes $(i_1, \dots, i_N)$ the so-called discrete Leja points extracted from the original set of nodes; cf. [3] for a full description of the algorithm and of the properties of discrete Leja points (for example, the fact that they form a sequence). For the theoretical connections of these Fekete-like and Leja-like extremal sets to multivariate polynomial interpolation theory, we refer the reader to [3]. From the computational point of view, again at high degree n a preliminary orthonormalization of the basis is needed that works until the conditioning of the Vandermonde-like matrix is not much higher than the reciprocal of machine precision; cf. [3, 34].

Once the discrete Leja points $\mathcal{L} = \{x_{i_1}, \dots, x_{i_N}\}$ have been extracted, the quadrature weights $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_N)$ are computed essentially by solving the square system

$$T^* \boldsymbol{\lambda} = \boldsymbol{\nu} \ , \ \ T^* = (t^*_{sj}) = (t_{s i_j}) = \big(\tau_s(x_{i_j})\big) \in \mathbb{R}^{N \times N} \ , \tag{5}$$

where T^* is the square submatrix of T corresponding to a subset of column indices $(i_1, \dots, i_N)$, which has been obtained by the pivoting in the LU factorization. The solution of the above system leads to a compressed quadrature formula

$$\mathcal{Q}_{CQ} = (\mathcal{L}, \boldsymbol{\lambda}) \ , \ \ \mathrm{card}(\mathcal{L}) = N \ . \tag{6}$$

In this case (and also with the approximate Fekete points) the weights are not all positive, in general. However, it turns out that with good distributions of starting nodes, like those typical of quadrature on polygons or on polyhedra that are the object of the present paper, the negative weights are few and of relatively small size, so that the quadrature sensitivity parameter

$$\rho_n = \frac{\sum_{s=1}^N |\lambda_s|}{\left|\sum_{s=1}^N \lambda_s\right|} = \frac{\sum_{s=1}^N |\lambda_s|}{meas(K)} \tag{7}$$

remains close to 1 or grows very slowly with n, ensuring accuracy and stability to the quadrature process; cf. [11, 35]. In fact, if the sampled values are affected by an error at most ε, i.e., we apply the quadrature formula with function values $\tilde{f}(x_{i_s})$ such that $|\tilde{f}(x_{i_s}) - f(x_{i_s})| \le \varepsilon$, $1 \le s \le N$, we can easily derive the accuracy/stability error estimate

$$\begin{aligned} \left| \int_K f(x)\, dx - \sum_{s=1}^N \lambda_s\, \tilde{f}(x_{i_s}) \right| &\le \left(meas(K) + \sum_{s=1}^N |\lambda_s| \right) E_n(f) + \varepsilon \sum_{s=1}^N |\lambda_s| \\ &\le meas(K)\ \left((1+\rho_n)\, E_n(f) + \varepsilon\, \rho_n \right) \quad \forall f \in C(K)];, \end{aligned} \tag{8}$$

where $E_n(f) = \min_{p \in \mathbb{P}_n^d} \|f - p\|_{\infty,K}$ is the best uniform approximation error to f in $\mathbb{P}_n^d$. As is known, the order of $E_n(f)$ can be estimated by the regularity of f on the so-called Jackson compact sets, that are compact sets admitting a Jackson-like inequality (such as, e.g., polygons and polyhedra as union of triangles/tetrahedra); cf. [26].

In summary, the method of constructing the compressed quadrature rule $\mathcal{Q}_{CQ} := (\mathcal{L}, \boldsymbol{\lambda})$ involves four steps that are presented in Algorithm 1. The first step is to construct the base quadrature $\mathcal{Q}_{BQ} := (X, \boldsymbol{w})$ with M number of points, over the considered polyhedron. The present work uses the direct divergence method [36] to construct $\mathcal{Q}_{BQ}$, but any other available technique can be used for this purpose. It should be noted that M can be on the order of thousands [36, 37] for a complex polyhedron. The next step is to construct the Vandermonde-like matrix T (of dimension $N \times M$) on X using the considered total-degree product Chebyshev basis functions $\boldsymbol{\tau}(x)$ (see (4)). The cardinality of the considered base functions is N, and usually $N < M$. Then, we extract $\mathcal{L} \subset X$ by performing LU decomposition with row-pivoting of T^t; the set $\mathcal{L}$ contains N number of points. As a next step, we form the Vandermonde-like matrix T^* (of dimension $N \times N$) on $\mathcal{L}$ using $\boldsymbol{\tau}(x)$. Finally, we compute the modified weights $\boldsymbol{\lambda}$ on $\mathcal{L}$ by solving $T^*\boldsymbol{\lambda} = \boldsymbol{\nu}$. (See (5) for more information on $\boldsymbol{\nu}$.) Thus the compressed quadrature rule $\mathcal{Q}_{CQ} := (\mathcal{L}, \boldsymbol{\lambda})$ is constructed.

Algorithm 1 Construction of compressed quadrature $\mathcal{Q}_{CQ} := (\mathcal{L}, \boldsymbol{\lambda})$ over a polyhedron.

```
1: Construct base quadrature $\mathcal{Q}_{BQ} := (X, \boldsymbol{w})$ over the polyhedron
2: Form Vandermonde-like matrix $T$ on $X$ using the base functions (see (4))
3: Extract discrete Leja points $\mathcal{L} \subset \mathbf{X}$ by performing LU-decomposition of $T^t$
4: Form square submatrix $T^*$ from $T$ on $\mathcal{L}$ using the base functions
5: Get modified weights $\boldsymbol{\lambda}$ on $\mathcal{L}$ by solving $T^*\boldsymbol{\lambda} = \boldsymbol{\nu}$
```

At this point, it is interesting to discuss the similarities of the proposed method with moment fitting methods [15, 21, 23, 37]: in both methods the quadrature schemes are obtained by solving an optimization problem. In its original form, the moment fitting methods solve a nonlinear optimization problem to get the location and weights of the integration points. However, since this is not feasible when applied to embedded interface methods, integration points are distributed within the domain and a linear optimization, as in the present work, is solved to obtain the quadrature weights. The problem with this approach is that a general method for proper distribution of points within an arbitrary polyhedron is not available (see [15, 37] for two available methods), and poorly distributed points give rise to an ill-conditioned system matrix, which results in reduced accuracy of resulting quadrature schemes [37]. By replacing the distribution of integration points with the construction of base quadrature rule, the present method guarantees well-conditioned Vandermonde matrix, which leads to accurate quadrature rules, as will be shown in the next section. The direct divergence method is the best choice for construction of $\mathcal{Q}_{BQ}$ because it can construct accurate quadrature schemes with least amount of computational time [36].

2.1. Polyhedral quadrature formulas. The quadrature compression algorithm, if it is to be applied in an EIM framework, must be highly robust to handle complex polyhedra. In order to test the robustness, we perform quadrature compression over some selected polyhedra and study the accuracy of integrations performed using the compressed quadrature. Purposefully, we chose very complex shapes so that the accuracy comparison is meaningful.

The polyhedra considered are shown in Figures 1 and 2, which illustrate the distribution of discrete Leja points and the $M - N$ points that are eliminated during the LU-decomposition process. The combination of these two sets of points define the integration points of the base quadrature rule.

Note. It can be seen from Figures 1 and 2 that some of the integration points are outside the polyhedra for concave shapes. This is not the result of the quadrature compression. The only reason for this is because $\mathcal{Q}_{BQ}$ contains points that are outside the polyhedra [36]. As mentioned earlier, the quadrature compression always selects a subset of points from $\mathcal{Q}_{BQ}$.

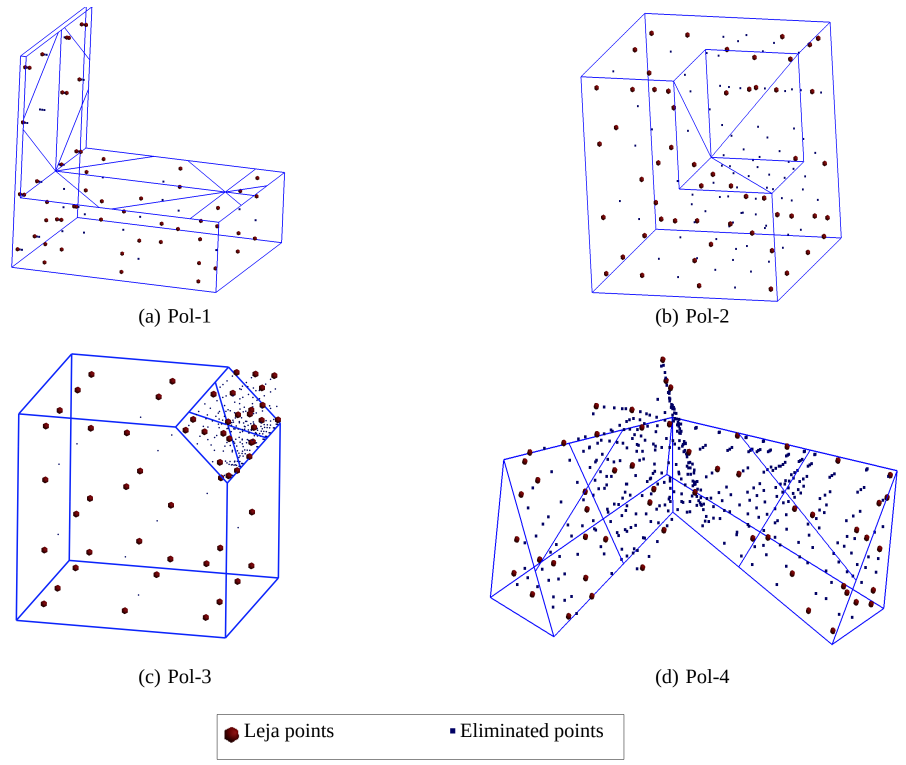


FIG. 1. *Quadrature compression over arbitrary polyhedra. Discrete Leja points represent the compressed quadrature and the collection of both Leja points and the eliminated points define the base quadrature.*

In all the examples presented in this work, we consider fifth-order quadrature rules. The number of points in the base quadrature (N_{BQ}) is dictated by the complexity of the polyhedra (Table 1). However, by construction of the method, the number of compressed quadrature points (N_{CQ}) is equal to the cardinality of the basis functions considered. In three dimensions, the cardinality of fifth-order basis functions is 56. This means that irrespective of the shape of the polyhedra, each volume will have 56 integration points in $\mathcal{Q}_{CQ}$.

Accuracy of the compressed quadrature rule is quantified by the quadrature error, which is defined as

$$e_{CQ} = \left| \frac{I_{BQ} - I_{CQ}}{I_{BQ}} \right|, \tag{9}$$

where I_{BQ} and I_{CQ} are the integral values obtained using the base quadrature and the compressed quadrature rule, respectively.

First, the errors incurred in the integration of basis functions, which are used to perform the quadrature compression, are quantified. In order to do so, e_{CQ} of all the 56 basis functions are computed, and the maximum value among them for each of the considered polyhedra is reported in Table 1. Two methods are considered for the construction of $\mathcal{Q}_{BQ}$. Tessellation involves decomposition of the polyhedron into a number of tetrahedra, and the direct divergence method makes use of the divergence theorem [36]. It can be seen from Table 1 that both methods produce accurate re-

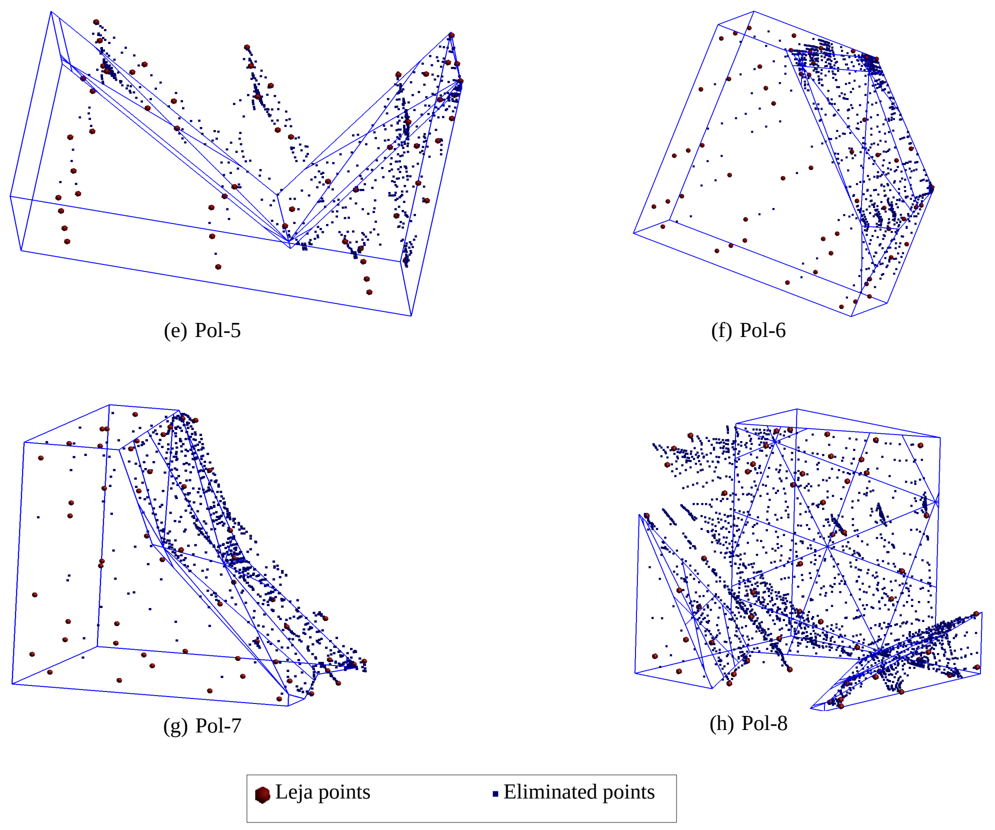


FIG. 2. *Additional polyhedra over which quadrature compression is performed.*

TABLE 1

Number of integration points in base quadrature, quadrature sensitivity parameter (ρ_n), and the error in integration of basis functions (e_{CQ}) while using two different base quadratures: tessellation and direct divergence. Irrespective of the base quadrature used or the complexity of polyhedra, the comprssed quadrature have 56 *integration points.*

	Tessellation			Direct divergence		
	N_{BQ}	ρ_n	e_{CQ}	N_{BQ}	ρ_n	e_{CQ}
Pol-1	1032	1.84	6.73×10^{-14}	96	4.24	4.64×10^{-14}
Pol-2	528	1.64	5.25×10^{-14}	144	1.63	6.89×10^{-14}
Pol-3	528	1.19	3.42×10^{-13}	288	1.89	7.04×10^{-13}
Pol-4	432	1.64	9.56×10^{-14}	504	2.45	1.32×10^{-13}
Pol-5	1344	1.53	5.74×10^{-12}	1044	2.78	1.45×10^{-11}
Pol-6	1680	2.67	1.39×10^{-13}	1044	2.56	2.46×10^{-13}
Pol-7	1440	1.78	6.66×10^{-13}	1152	1.98	9.82×10^{-13}
Pol-8	2376	1.81	9.38×10^{-14}	2520	7.85	1.75×10^{-13}

sults with the largest error being 10^{-11}. Though tessellation produces slightly more accurate $\mathcal{Q}_{BQ}$, the direct divergence method is preferred for its superior robustness characteristics and for the fast construction of $\mathcal{Q}_{BQ}$ [36]. Hence, all the results presented hereafter makes use of $\mathcal{Q}_{BQ}$ constructed using the direct divergence method.

In the next step, it is checked whether $\mathcal{Q}_{CQ}$ is accurate to integrate a general polynomial. In order to check this, the following polynomials are integrated over the

Table 2
Error incurred in integrating the polynomial functions using the compressed quadrature.

	$p_0(\mathbf{x})$	$p_1(\mathbf{x})$	$p_2(\mathbf{x})$	$p_3(\mathbf{x})$	$p_4(\mathbf{x})$	$p_5(\mathbf{x})$
Pol-1	2×10^{-16}	8×10^{-16}	3×10^{-15}	1×10^{-15}	2×10^{-15}	1×10^{-14}
Pol-2	1×10^{-16}	1×10^{-15}	1×10^{-15}	1×10^{-14}	3×10^{-15}	9×10^{-15}
Pol-3	0	6×10^{-15}	9×10^{-14}	3×10^{-15}	5×10^{-15}	3×10^{-15}
Pol-4	5×10^{-16}	6×10^{-16}	9×10^{-15}	2×10^{-15}	7×10^{-15}	3×10^{-15}
Pol-5	2×10^{-16}	6×10^{-15}	5×10^{-15}	5×10^{-15}	3×10^{-15}	1×10^{-14}
Pol-6	3×10^{-16}	2×10^{-16}	9×10^{-14}	5×10^{-15}	5×10^{-16}	6×10^{-15}
Pol-7	4×10^{-16}	1×10^{-14}	5×10^{-15}	6×10^{-15}	1×10^{-14}	2×10^{-14}
Pol-8	5×10^{-16}	2×10^{-16}	2×10^{-15}	1×10^{-14}	1×10^{-13}	8×10^{-15}

considered polyhedra, and the corresponding error is reported in Table 2:

$$\begin{aligned}
p_0(\mathbf{x}) &= 1\,,\\
p_1(\mathbf{x}) &= x + 2y + 3z\,,\\
p_2(\mathbf{x}) &= x^2 - 2y^2 + z^2\,,\\
p_3(\mathbf{x}) &= -x^3 + xyz + y^3 + z^3\,,\\
p_4(\mathbf{x}) &= x^4 - 4y^4 + 7xz^3 + z^4\,,\\
p_5(\mathbf{x}) &= x^5 + 5xyz^3 - 10xy^3z + 5x^3yz + y^5 + z^5\,.
\end{aligned}$$

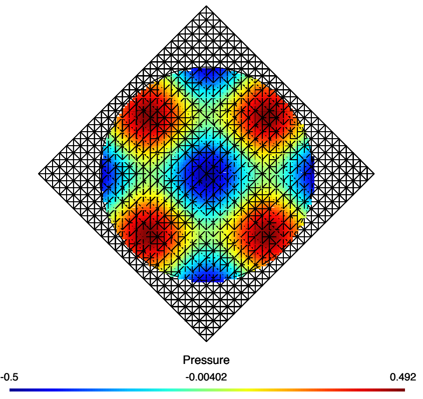
It is clearly evident from Table 2 that the compressed quadrature is accurate enough to integrate all the considered polynomials over all the polyhedra; the maximum error is of the order of 10^{-13}. This is despite the fact that the considered polyhedra include highly complex and concave shapes. Another remarkable observation is that for Pol-8, more than 2500 points in $\mathcal{Q}_{BQ}$ are reduced to 56 points in $\mathcal{Q}_{CQ}$. Even after such a drastic reduction, e_{CQ} stays as low as 10^{-13} for all the polynomials. Therefore, it can be concluded that the quadrature compression procedure is robust enough to be used in EIMs.

We can expect that owing to the reduced number of integration points, the compressed quadrature can lead to efficient integration of the weak forms associated with EIMs. This feature is explored in the next section.

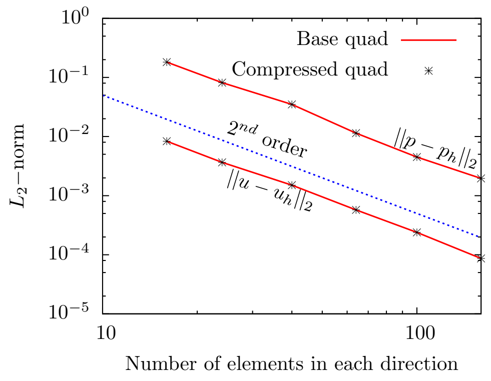
3. Application to embedded interface methods.

In the previous section, the quadrature compression is performed over various complex shaped polyhedra, and the results of integrating predefined polynomials are reported. This method is implemented to integrate weak forms of the Navier–Stokes equations discretized by using an embedded interface method [30, 31, 32]. The performance of the quadrature compression algorithm in EIM framework is studied here. We consider two classes of problems for which EIMs are ideally suited: flow over multiple immersed objects and flow past arbitrary moving structures.

In all the test cases, two simulations are performed: one simulation involving base quadrature obtained from the direct divergence method [36] and another utilizing the compressed version of the base quadrature to perform weak form integration over the cut-elements. The comparison of these two simulations enables us to quantify the influence of using the compressed quadratures on the accuracy, rate of convergence, and the computational efficiency of EIM simulations. In sections 3.2–3.4, we introduce the setup of three examples as well as quantify the accuracy of compressed quadratures wherever possible. Later in section 3.5, these examples are used to discuss the computational efficiency of using quadrature compression in EIMs.



(a) Pressure distribution



(b) Convergence plot

FIG. 3. *Taylor–Green vortex.*

3.1. Taylor–Green vortex. In the last section, it is shown that the compression of quadrature rules introduced errors as low as 10^{-11} even for very complex polyhedra. However, in order to prove the usefulness of the compressed quadratures, it is essential to show that these errors do not affect the rate of convergence of finite element simulations. In order to do so, we consider the stationary Taylor–Green vortex problem [16], for which the analytical solutions are readily available.

The velocity and pressure solution are given as

$$u_x(x,y) = -\cos(2\pi x)\sin(2\pi y)\,, \tag{10}$$

$$u_y(x,y) = \sin(2\pi x)\cos(2\pi y)\,, \tag{11}$$

$$p(x,y) = -\frac{1}{4}\left(\cos(4\pi x) + \cos(4\pi y)\right)\,, \tag{12}$$

which satisfy the continuity equation $\nabla \cdot \mathbf{u} = 0$. The numerical solution is computed on a circular domain of radius 0.45 units, whose center is located at (0.5,0.5). This circular domain is immersed into the background fluid mesh defined on $[0,1]^2$, rotated by 45^o (see Figure 3(a)). The fluid mesh contains one element in the depth direction, so that the interface cut produces 3D polyhedra.

It should be mentioned that on the boundary of the circular domain, the boundary conditions for velocity (given by the analytical solution) are enforced weakly by the recently derived variant of Nitche's method [30, 31, 32]. Since this is a pure Dirichlet problem the pressure value, given by (12), is fixed at the center node (0.5,0.5).

The background fluid mesh is discretized with uniform wedge elements. The pressure distribution within the circular domain is presented in Figure 3(a), and the vortex distribution qualitatively resemble the expected flow behavior. In order to quantify the errors and to predict the rate of convergence, the L_2-norm of pressure and velocity errors for various mesh densities are computed. The plot of L_2-norm with increasing mesh density is given in Figure 3(b). Owing to the smoothness of the solution field, as expected, we obtain optimal order of convergence rate for both pressure and velocity. It can be seen that the simulations with quadrature compression

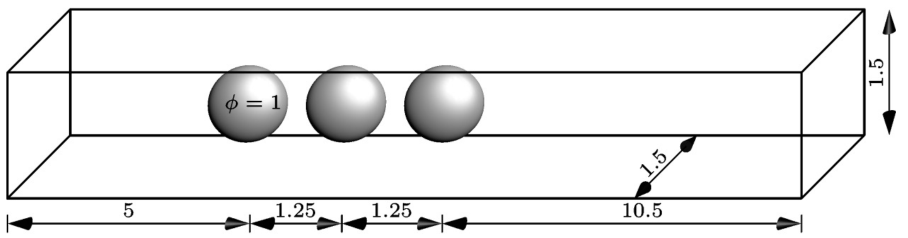


FIG. 4. *Geometric details of the multiple stationary interfaces problem.*

also maintain the optimal convergence rate. Moreover, the magnitude of L_2-norm is also exactly the same for both the base quadrature and the compressed quadrature.

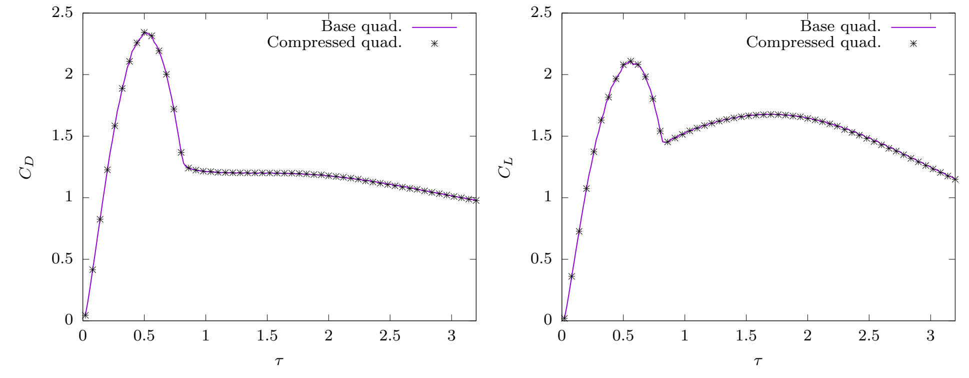
From the results presented in this section, it can be concluded that the compressed quadrature does not introduce additional errors into the computation. In addition, it also enables us to achieve the same rate of convergence as that of the base quadrature.

3.2. Flow past 3D multiple stationary interfaces. This example considers flow past 3D multiple stationary interfaces. Three spheres of diameter $D_s = 1m$ are placed inside a pipe of dimensions $18m \times 1.5m \times 1.5m$. The configuration of the simulation is depicted in Figure 4. A quadratic velocity profile with maximum velocity $u_{\max} = 1m/s$ is specified in the inflow, and a "do-nothing" condition is specified on the outflow. On all other surfaces, no-slip condition is enforced. The kinematic viscosity of the fluid, $\nu = 0.01m^2/s$, results in Reynolds number $Re = 100$ by taking D_s as the length scale. Unsteady simulations with time step $\Delta t = 0.1$ are performed for 1250 steps.

Since the present simulation involves flow over stationary interfaces, the base quadrature construction using the direct divergence method and its compression need to be carried out only once in the beginning of the simulation. The computational efficiency of employing the quadrature compression technique for this example is discussed in section 3.5.

3.3. Flow over a rapidly accelerating airfoil. The previous section considered flow past stationary interfaces. Since the interfaces are held stationary in that example, the quadrature compression is performed only once in the beginning of the simulation. However, EIMs are mainly used and are most challenging in practical applications in which the interface changes its shape and position with time. This leads to different polyhedral shapes at each time step over which the numerical integration has to be carried out. Owing to this fact, the geometrical cutting operations and the compression of the quadrature schemes need to be performed at each time step; also, all kinds of critical and challenging polyhedral shapes will appear naturally. The present and the next examples test the accuracy of the simulations using quadrature compression in such situations.

The present example resolves the flow field around a rapidly accelerated airfoil in a still fluid, as described in [13]. The NACA0012 airfoil is used in our simulations. It accelerates from rest to a velocity $U = 1$, and then it traverses with this constant velocity. Throughout the simulation, the airfoil maintains a constant angle of attack of 35^o. This test case is a pseudo-3D simulation in which only one finite element is used in the depth direction. The comparison of time-dependent forces acting on the airfoil enables us to perform quantitative analysis of the accuracy of compressed quadrature rules.



(a) Drag coefficient (b) Lift coefficient

FIG. 5. *Comparison of drag and lift coefficients for a rapidly accelerated airfoil.*

The unsteady displacement of the airfoil is given as follows:

$$x(\tau) = -\frac{1}{2}\left[\tau - \frac{\tau_a}{\pi}\sin\left(\frac{\pi\tau}{\tau_a}\right)\right] \qquad \text{for } 0 \leq \tau \leq \tau_a \tag{13a}$$

$$x(\tau) = \frac{\tau_a}{2} - \tau \qquad \text{for } \tau > \tau_a\ , \tag{13b}$$

where $\tau = \frac{tU}{c}$ is the nondimensional time, $\tau_a = 0.8$ is the acceleration duration, and c is the chord length of the airfoil. The Reynolds number, the nondimensional number that characterizes the flow field, $Re = \frac{Uc}{\nu}$ is set to be 100, where ν is the kinematic viscosity of fluid. The computational domain is taken to be $24c \times 20c$, where c is the chord length of the airfoil. The complete subdomain within which the airfoil traverses is discretized with a very fine mesh, and a coarse mesh is used away from this region. The time step used in the simulation is $\Delta\tau = 0.02$.

As in the previous example, two simulations are performed: one with the base quadrature and another with the compressed quadrature rule. The main objective of this simulation is to investigate whether the use of the compressed quadrature rule for the weak form integration leads to accurate quantitative results in fluid flow modeling. The drag (C_D) and lift (C_L) coefficients computed from the above two simulations are compared for this purpose.

The time evolution of C_D and C_L are shown in Figure 5. Lift and drag start raising during the acceleration phase and reach their peak values. After this, they fall down and settle almost to a constant value during the constant velocity translation phase. It is directly apparent from the figure that the drag- and lift-curves produced from both the simulations fall exactly one over the another. This demonstrates that the compressed quadrature rules are accurate enough to compute the time-dependent forces acting on the airfoil.

3.4. Fluid-structure interaction of a bending beam. The last section investigated the accuracy of the compressed quadrature for moving boundary simulations. However, the problem considered was pseudo-3D, and hence the resulting polyhedral shapes are of limited complexity. In order to study the influence of the proposed algorithm, a real 3D problem involving a coupled fluid-structure interaction of a flexible bending beam is simulated.

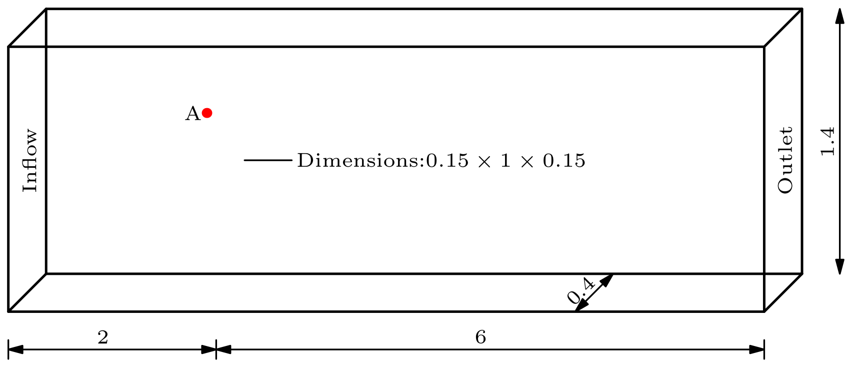


FIG. 6. *Configuration of fluid-structure interaction of a bending beam.*

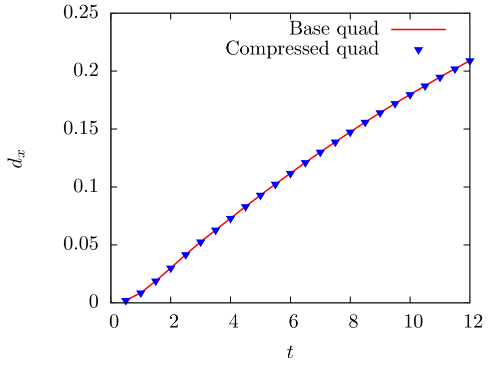


FIG. 7. *Comparison of displacement component in x-direction at point A in Figure* 6 with time.

The complete configuration of the simulation is shown in Figure 6. A flexible beam, whose bottom part is fixed, is immersed in the fluid flow. The material properties of the beam are Young's modulus, $E = 100Pa$, Poisson's ratio $\nu = 0.1$, and density $\rho_s = 10kg/m^3$. A quadratic velocity profile with maximum velocity $u_{\max} = 1m/s$ is specified in the inflow, and a do-nothing condition is specified on the outflow. The top, bottom, and side walls are no-slip boundaries. The kinematic viscosity of the fluid $\nu_f = 0.05m^2/s$. This corresponds to a Reynolds number of Re = 20, where the length scale is the height of the beam. The FSI method proposed in [12] is used to simulate this test case.

The flexible beam starts bending due to the fluid loads acting on the beam. In order to quantify the accuracy of the compressed quadrature, the x-component of displacement of the beam at point A (Figure 6) is given in Figure 7. This plot shows that the simulation with compressed quadrature produces exactly the same result as the base quadrature.

This result is encouraging due to the following reason. In FSI examples, the interface movement at each step leads to a different orientation and relative position of the interface with respect to the cut-elements. This means that at each new time step, the cut configuration is different and we get different polyhedral shaped volume-cells over which the quadrature compression is carried out. EIM-based fluid dynamic simulations are sensitive to the accuracy of quadrature rules: especially in coupled

TABLE 3

Number of points in the base quadrature ($\bar{N}_{BQ}$) for the considered element, time spent in performing quadrature compression (t_{CQ}), evaluation times ($\bar{E}_{BQ}$, and $\bar{E}_{BQ}$), and efficiency gain ($\bar{\eta}_E$). All the reported times are in seconds.

N_{BQ}	t_{CQ}	E_{BQ}	E_{CQ}	$\bar{\eta}_E$ (%)
72	3.15×10^{-4}	5.68×10^{-4}	7.53×10^{-4}	-32.57
252	5.56×10^{-4}	1.90×10^{-3}	9.96×10^{-4}	47.59
600	1.22×10^{-3}	4.49×10^{-3}	1.66×10^{-3}	63.07
816	1.52×10^{-3}	6.09×10^{-3}	2.02×10^{-3}	66.78
1440	2.73×10^{-3}	1.07×10^{-2}	3.24×10^{-3}	93.42

problems, the sensitivity is much more pronounced because even a small error in the integrals will change the pressure distribution, and this will affect the structural movement, which in-turn changes the fluid flow. Accurate matching of the time-dependent quantities imply that for all the polyhedra generated in our simulations, the accuracy of compressed quadratures is close to that of the base quadratures.

3.5. Discussion on computational efficiency. The use of quadrature compression leads to reduction in the number of integration points, which can help to reduce the simulation time. However, some amount of computational time is spent in performing the compression over each polyhedron (Table 1). This section studies the overall influence of using the compressed quadrature rules on the simulation time for the above simulations.

The base quadrature for volume-cells may contain more than 1000 points, but the compressed quadrature, by construction, needs only 56 points for the same accuracy. Due to this reduction in number of points, the evaluation time might be reduced when using the compressed quadrature. As a first step, we consider a few cut elements from the stationary interfaces example (section 3.2) to quantify the effect of quadrature compression on the evaluation time. We take several cut elements that have different N_{BQ} and present evaluation times in Table 3. The efficiency gain is quantified by $\bar{\eta}_E$, which is defined as

$$\bar{\eta}_E = \frac{\bar{E}_{BQ} - \bar{E}_{CQ}}{\bar{E}_{BQ}}\,, \tag{14}$$

where $\bar{E}_{BQ}$ is the evaluation time over the considered domain using the base quadrature, and $\bar{E}_{CQ}$ includes evaluation time using the compressed quadrature in addition to the time spent in performing the compression operations. It should be mentioned that the evaluation time does not include the time required to construct the base quadrature.

The quadrature compression algorithm requires performing the LU-decomposition of the underdetermined system presented in (4) and the solution of the matrix system given in (5). It can be seen from Table 1 that the time required to execute the proposed method (t_{CQ}) increases when $\bar{N}_{BQ}$ is higher. This is due to the fact that the dimension of the Vandermonde-like matrix increases when $\bar{N}_{BQ}$ is higher. As a result, the time required to perform LU factorization of the matrix goes up, and hence t_{CQ}. This is directly evident from Table 3.

When N_{BQ} is small, the time required to perform quadrature compression is larger than the gain in computational time we get by the use of compressed quadrature. This is the reason for having negative $\bar{\eta}_E$ when $N_{BQ} = 72$. However, when N_{BQ} is increased, the quadrature compression yields improvements in evaluation time, which

Table 4

Ratio of number of points in base quadrature (N_{BQ}) to the number of points in compressed quadrature (N_{CQ}) in the mesh, evaluation times for the entire simulation and efficiency gain (η_E). For moving interface simulations, N_{cut}/N_{total} changes with time, and the ratio at the initial configuration is reported.

	N_{BQ}/N_{CQ}	Evaluation time (s)		η_E (%)
		E_{BQ}	E_{CQ}	
Stationary spheres	3.726	3.853×10^4	2.895×10^4	24.86
Accelerating airfoil	1.027	1.138×10^4	0.990×10^4	12.98
Bending beam	1.037	2.735×10^4	2.087×10^4	23.67

Table 5

Time required for quadrature compression (t_{CQ}), geometrical and computational efficiency gain due to the use of $\mathcal{Q}_{CQ}$ (η_T).

	t_{CQ} (s)	t_{cut} (s)	Total simulation time (s)		η_T (%)
			T_{BQ}	T_{CQ}	
Stationary spheres	1.4004	19.43	4.188×10^4	3.191×10^4	23.80
Accelerating airfoil	20.996	210	4.350×10^4	3.882×10^4	10.76
Bending beam	427.78	2791	7.893×10^4	7.296×10^4	7.566

is reflected in Table 3 as positive $\bar{\eta}_E$. Even when $N_{BQ} \simeq 250$, which is quite common in EIMs, $\bar{\eta}_E$ is around 47%.

In EIMs, only cut-elements require a special quadrature rule with a large number of points, and in the remaining noncut elements the efficient traditional Gauss quadrature rules can be directly employed. In order to study the effect of quadrature compression on EIM simulations, we compute all the quantities given in Table 3 for the whole fluid mesh (including noncut elements). The results are presented in Table 4. The overbar is dropped from all quantities to indicate that the results reported are for the whole mesh and for the entire simulation. In contrast to the results presented for single element, η_E is not consistent with the ratio N_{BQ}/N_{CQ}. This is because the cut elements in our simulations are not always equally distributed among the processors. Despite this fact, we can see from Table 4 that for all the three simulations reported here, the efficiency of performing weak form integration is improved due to the use of quadrature compression.

The most important quantity is the total simulation time, which is given in Table 5. By observing η_T (defined exactly like η_E but with total simulation times T_{BQ} and T_{CQ}), it is directly evident that for the stationary interface problem, the computational efficiency gain in η_E directly translates to η_T. However, for moving boundary simulations, this is not the case. One of the reasons is that since the interface is nonstationary, the geometrical cutting operations required to form the volume-cells and the quadrature compression need to be performed at each time step, and this contributes to reduction in η_T. For bending beam simulation, η_T is reduced when compared to η_E. Since this example is an FSI simulation, it involves solving the structural dynamics equations and, moreover, the time required to perform the geometrical cutting operations (t_{cut}) as well as the quadrature compression are also larger. The time invested in solving the structural dynamics equations and t_{cut} are the same irrespective of using $\mathcal{Q}_{BQ}$ or $\mathcal{Q}_{CQ}$. Owing to these operations, η_T is smaller when compared to η_E, which is defined only based on the time required to compute the stiffness matrix associated with the governing equations of fluid flow. However, despite all these points, by simply applying certain standard linear algebraic methods, we obtain as much as 7.56% gain in efficiency. From these observations, it can

TABLE 6
Efficiency gain in evaluation time on cut-elements.

	$\eta_{cutE}(\%)$
Stationary spheres	58.25
Accelerating airfoil	93.83
Bending beam	93.75

be concluded that the use of quadrature compression leads to improvements in total simulation time.

An interesting extension of the present work is to apply the quadrature compression to polygonal and polyhedral finite element methods [8, 18, 28, 38, 43] in which all the elements in the mesh are of polyhedral shape. Therefore special quadrature schemes need to be used for all the elements in the computational domain. As a result, the application of the quadrature compression in such methods might lead to significant gain in computational efficiency when compared to EIMs in which the quadrature compression is applied only to a few elements in the mesh. In order to have a preliminary impression of the advantage of using quadrature compression in polyhedral FEM, we define an efficiency parameter (η_{cutE}), similar to η_E in (14). While η_E is based on the evaluation time over the whole computational domain, η_{cutE} is defined by considering the evaluation time only on the cut-elements, which requires integration over arbitrary polyhedra. The estimated η_{cutE} for the considered simulations are shown in Table 6, and it can be seen that it is significantly larger than η_E. η_{cutE} can be considered as the parameter characterising the efficiency gain in evaluation time for polyhedral FEM. This efficiency gain in evaluation time will directly reflect as a significant reduction in total simulation time.

Additional efficiency gain, in the context of polyhedral finite element methods, can be achieved by combining the present approach with the node elimination technique proposed in [22, 44]. In this technique, nonlinear moment fitting equations are solved using Newton's method to arrive at the location of the quadrature points and their weights. From this initial quadrature rule, the point of least significance is eliminated and the Newton's method is called once again to arrive at the new integration rule, which contains N-1 points. This approach can be applied recursively until the Newton method cease to converge. As an extension of the present work, the compressed quadrature rules can be taken as the initial condition and node elimination can be performed on them. This can help us to reduce the number of integration points even further, thus resulting in even higher computational efficiency. Moreover, the node elimination technique may also be used to eliminate the points with negative quadrature weights in the compressed quadrature, and this improves the stability of the quadrature. However, the time required to perform node elimination is very high, since reduction of one point requires solving the nonlinear system of equations by Newton's method. So they are not suitable for EIMs. Since in polyhedral FEMs, the quadrature schemes are constructed only once in the beginning of the simulations, it will be interesting to see how much efficiency one can gain with this combined approach.

In addition, the quadrature compression techniques can be of help in variety of other fields: level set based methods [19, 29], electronic structure calculations [1, 2], aerospace applications [9, 33, 41], and computational geometry [5, 20, 40], to name a few.

4. Conclusion.

We apply the multivariate quadrature compression to the algebraic quadrature formulas on complex polyhedra. The quadrature compression is very simple to implement since it is completely based on standard numerical algebraic methods. It operates on an existing quadrature rule by selecting a subset of nodes and modifying the corresponding weights in such a way that the accuracy of the original quadrature rule is preserved. In the compressed quadrature rule, the number of integration points is always equal to the dimension of the considered polynomial space. The compressed quadratures are applied in an embedded interface method to integrate the weak form of the Navier–Stokes equations. Simulations of flow past stationary and moving interface problems demonstrate that compressed quadratures preserve accuracy as well as order of convergence and improve the computational efficiency of embedded interface methods.

REFERENCES

[1] A. Alam, S. N. Khan, B. G. Wilson, and D. D. Johnson, *Efficient isoparametric integration over arbitrary space-filling Voronoi polyhedra for electronic structure calculations*, Phys. Rev. B, 84 (2011), 045105.

[2] P. M. Boerrigter, G. Te Velde, and E. J. Baerends, *Three-dimensional numerical integration for electronic structure calculations*, Int. J. Quantum Chem., XXXIII (1988), pp. 87–113.

[3] L. Bos, S. D. Marchi, A. Sommariva, and M. Vianello, *Computing multivariate Fekete and Leja points by numerical linear algebra*, SIAM J. Numer. Anal., 48 (2010), pp. 1984–1999.

[4] P. Businger and G. Golub, *Linear least-squares solutions by householder transformations*, Numer. Math., 7 (1965), pp. 269–276.

[5] C. Cattani and A. Paoluzzi, *Boundary integration over linear polyhedra*, Comput.-Aided Des., 22 (1990), pp. 130–135.

[6] E. B. Chin, J. B. Lasserre, and N. Sukumar, *Numerical integration of homogeneous functions on convex and nonconvex polygons and polyhedra*, Comput. Mech., 56 (2015), pp. 967–981.

[7] A. Civril and M. Magdon-Ismail, *On selecting a maximum volume submatrix of a matrix and related problems*, Theoret. Comput. Sci., 410 (2009), pp. 4801–4811.

[8] G. Dasgupta, *Integration within Polygonal Finite Elements*, J. Aerosp. Eng., 16 (2003), pp. 9–18.

[9] A. R. Dobrovolskis, *Inertia of Any Polyhedron*, Icarus, 124 (1996), pp. 698–704.

[10] C. F. Dunkl and Y. Xu, *Orthogonal Polynomials of Several Variables*, Cambridge University Press, Cambridge, 2001.

[11] M. Gentile, A. Sommariva, and M. Vianello, *Polynomial interpolation and cubature over polygons*, J. Comput. Appl. Math., 235 (2011), pp. 5232–5239.

[12] A. Gerstenberger and W. A. Wall, *An extended finite element method/Lagrange multiplier based approach for fluid-structure interaction*, Comput. Methods Appl. Mech. Engrg., 197 (2008), pp. 1699–1714.

[13] H. Hamdani and M. Sun, *Aerodynamic forces and flow structures of an airfoil in some unsteady motions at small reynolds number*, Acta Mech., 145 (2000), pp. 173–187.

[14] D. Huybrechs, *Stable high-order quadrature rules with equidistant points*, J. Comput. Appl. Math., 231 (2009), pp. 933–947.

[15] M. Joulaian, S. Hubrich, and A. Düster, *Numerical integration of discontinuities on arbitrary domains based on moment fitting*, Comput. Mech., 57 (2016), pp. 979–999.

[16] J. Kim and P. Moin, *Application of a fractional-step method to incompressible Navier–Stokes equations*, J. Comput. Phys., 59 (1985), pp. 308–323.

[17] C. Lawson and R. Hanson, *Solving Least Squares Problems*, SIAM, Philadelphia, 1995.

[18] P. Milbradt and T. Pick, *Polytope finite elements*, Internat. J. Numer. Methods Engrg., 73 (2008), pp. 1811–1835.

[19] C. Min and F. Gibou, *Geometric integration over irregular domains with application to level-set methods*, J. Comput. Phys., 226 (2007), pp. 1432–1443.

[20] B. Mirtich, *Fast and accurate computation of polyhedral mass properties*, J. Graph. GPU Game Tools, 1 (1996), pp. 31–50.

[21] S. E. Mousavi and N. Sukumar, *Numerical integration of polynomials and discontinuous functions on irregular convex polygons and polyhedrons*, Comput. Mech., 47 (2011), pp. 535–554.

[22] S. E. Mousavi, H. Xiao, and N. Sukumar, *Generalized Gaussian quadrature rules on arbitrary polygons*, Internat. J. Numer. Methods Engrg., 82 (2010), pp. 99–113.

[23] B. Mueller, F. Kummer, M. Oberlack, and Y. Wang, *Simple multidimensional integration of discontinuous functions with application to level set methods*, Internat. J. Numer. Methods Engrg., 92 (2012), pp. 637–651.

[24] S. Natarajan, S. Bordas, and D. R. Mahapatra, *Numerical integration over arbitrary polygonal domains based on Schwarz–Christoffel conformal mapping*, Internat. J. Numer. Methods Engrg., 80 (2009), pp. 103–134.

[25] J. P. Pereira, C. A. Duarte, D. Guoy, and X. Jiao, *hp-Generalized FEM and crack surface representation for non-planar 3-D cracks*, Internat. J. Numer. Methods Engrg., 77 (2009), pp. 601–633.

[26] W. Plesniak, *Multivariate Jackson inequality*, J. Comput. Appl. Math., 233 (2009), pp. 815–820.

[27] M. Putinar, *A note on Tchakaloff's theorem*, Proc. Amer. Math. Soc., 125 (1997), pp. 2409–2414.

[28] M. M. Rashid and M. Selimotic, *A three-dimensional finite element method with arbitrary polyhedral elements*, Internat. J. Numer. Methods Engrg., 67 (2006), pp. 226–252.

[29] U. Rasthofer, F. Henke, W. A. Wall, and V. Gravemeier, *An extended residual-based variational multiscale method for two-phase flow including surface tension*, Comput. Methods Appl. Mech. Engrg., 200 (2011), pp. 1866–1876.

[30] B. Schott, U. Rasthofer, V. Gravemeier, and W. A. Wall, *A face-oriented stabilized Nitsche-type extended variational multiscale method for incompressible two-phase flow*, Internat. J. Numer. Methods Engrg., 104 (2015), pp. 721–748.

[31] B. Schott, S. Shahmiri, R. Kruse, and W. Wall, *A stabilized Nitsche-type extended embedding mesh approach for 3D low- and high-Reynolds-number flows*, Internat. J. Numer. Methods Fluids, 82 (2016), pp. 289–315.

[32] B. Schott and W. A. Wall, *A new face-oriented stabilized XFEM approach for 2D and 3D incompressible Navier–Stokes equations*, Comput. Methods Appl. Mech. Engrg., 276 (2014), pp. 233–265.

[33] B. Singh and D. Guptasarma, *New method for fast computation of gravity and magnetic anomalies from arbitrary polyhedra*, Geophysics, 66 (2001), pp. 521–526.

[34] A. Sommariva and M. Vianello, *Computing approximate Fekete points by QR factorizations of Vandermonde matrices*, Comput. Math. Appl., 57 (2009), pp. 1324–1336.

[35] A. Sommariva and M. Vianello, *Compression of multivariate discrete measures and applications*, Numer. Funct. Anal. Optim., 36 (2015), pp. 1198–1223.

[36] Y. Sudhakar, J. P. M. de Almeida, and W. A. Wall, *An accurate, robust, and easy-to-implement method for integration over arbitrary polyhedra: Application to embedded interface methods*, J. Comput. Phys., 273 (2014), pp. 393–415.

[37] Y. Sudhakar and W. A. Wall, *Quadrature schemes for arbitrary convex/concave volumes and integration of weak form in enriched partition of unity methods*, Comput. Methods Appl. Mech. Engrg., 258 (2013), pp. 39–54.

[38] N. Sukumar and A. Tabarraei, *Confirming polygonal finite elements*, Internat. J. Numer. Methods Engrg., 61 (2004), pp. 2045–2066.

[39] Tchakaloff, *Formules de cubature mécaniques à coefficients nonnégatifs*, Bull. Sci. Math., 81 (1957), pp. 123–134.

[40] H. G. Trimmer and J. M. Stern, *Computation of global geometric properties of solid objects*, Comput.-Aided Des., 12 (1980), pp. 301–304.

[41] D. Tsoulis, *Analytical computation of the full gravity tensor of a homogeneous arbitrarily shaped polyhedra source using line integrals*, Geophysics, 77 (2012), pp. F1–F11.

[42] G. Ventura, *On the elimination of quadrature subcells for discontinuous functions in the eXtended Finite-Element Method*, Internat. J. Numer. Methods Engrg., 66 (2006), pp. 761–795.

[43] M. Wicke, M. Botsch, and M. Gross, *A finite element method on convex polyhedra*, Comput. Graph. Forum, 26 (2007), pp. 355–364.

[44] H. Xiao and Z. Gimbutas, *A numerical algorithm for the construction of efficient quadrature rules in two and higher dimensions*, Comput. Math. Appl., 59 (2008), pp. 663–676.

ERRATUM

The abstract for this article should read as follows.

Abstract. The main idea of this paper is to apply a recent quadrature compression technique to algebraic quadrature formulas on complex polyhedra. The quadrature compression substantially reduces the number of integration points but preserves the accuracy of integration. The compression is easy to achieve since it is entirely based on the fundamental methods of numerical linear algebra. The resulting compressed formulas are applied in an embedded interface method to integrate the weak form of the Navier–Stokes equations. Simulations of flow past stationary and moving interface problems demonstrate that the compressed quadratures improve the efficiency of performing the weak form integration, while preserving accuracy and order of convergence.