Adaptively Weighted Numerical Integration in the Finite Cell Method

Vaidyanathan Thiagarajan*, Vadim Shapiro

Spatial Automation Laboratory, University of Wisconsin-Madison, Madison, Wisconsin 53706, United States

Received 27 February 2016; received in revised form 14 July 2016; accepted 18 August 2016
Available online 29 August 2016

Abstract

With Adaptively Weighted (AW) numerical integration, for a given set of quadrature nodes, order and domain of integration, the quadrature weights are obtained by solving a system of suitable moment fitting equations in least square sense. The moments in the moment equations are approximated over a simplified domain that is homeomorphic to the original domain, and then are corrected for the deviation from the original domain using shape sensitivity analysis.

In this paper, we demonstrate the application of AW integration scheme in the context of the Finite Cell Method which must perform numerical integration over arbitrary domains without meshing. The standard integration technique employed in FCM is the characteristic function method that converts the continuous integrand over a complex domain into a discontinuous integrand over a simple (box) domain. Then, well known integrand adaptivity techniques are employed to integrate the resulting discontinuous integrand over the box domain. Although this method is simple to implement, it becomes computationally very expensive for realistic complex 3D domains such as sculptures, bones and engines.

In contrast, in AW scheme the quadrature weights directly adapt to the complex geometric domain without the need to making the integrand discontinuous leading to superior computational properties. In this paper, we demonstrate the computational efficiency of AW over the characteristic function method as it requires fewer subdivisions and less time to achieve a given accuracy in both two and three dimensions. In addition, AW offers a number of advantages including flexibility in the choice of quadrature points and basis functions.

Keywords: Numerical integration; Octree; Finite cell method; Moment fitting equations; Shape sensitivity analysis; Divergence theorem

1. Introduction

Mesh generation is a major problem in simulation-based engineering, involving considerable computational resources, especially for complex geometries. Approximately 80% of the total time invested for a FEA is expended in geometric modeling and mesh generation [1]. In addition, mesh generation suffers from several other drawbacks such as lack of robustness, lack of guarantees and inability to robustly handle small geometric/topological details. These
Fig. 1. The fictitious domain approach: the physical domain $\Omega$ is extended by the fictitious domain $\Omega_{\text{fict}}$ into an embedding domain $\Omega_e$ to allow easy meshing of complex geometries. The influence of $\Omega_{\text{fict}}$ is penalized by $\alpha$ [31].

challenges in mesh generation motivated the research in several meshless approaches such as the Generalized Finite Element Method (GFEM) [2] and Extended Finite Element Method (XFEM) [3]. A family of such meshfree methods, known as Immersed Boundary (IB) methods [4,5], have gained popularity in the recent past owing to its simplicity and wide applicability.

Unlike classical FEM, numerical integration is a major challenge in IB methods that immerses (or embeds) the original domain into a geometrically simpler domain (usually box-like domain). This simpler domain is then easily discretized to allocate basis functions. These basis functions are then used to discretize the variational (or weak) form of the boundary value problem under consideration resulting in a linear system (for linear problems) which when solved produces an approximation to the unknown field (such as displacements). Examples of IB methods include fictitious domain method [6–8], embedded domain method [9,10], solution structure method [11–15], certain versions of XFEM [16–21], Finite Cell Method (FCM) [22,23] and fixed-grid method [24–28].

The use of a structured voxel grid in IB methods eliminates most of the problems of conforming-mesh based methods. However, the use of a voxel grid makes numerical integration and enforcement of boundary conditions challenging. To understand this, let us consider the Finite Cell Method (FCM) [22,23] as an example. The FCM was originally introduced in [22] as an extension to the $p$-version of the finite element method [29,30]. FCM combines the fictitious domain approach with a higher-order approximation basis, the representation of the geometry by adaptive quadrature based on recursive bisection, and the weak imposition of unfitted boundary conditions [31]. FCM can operate on any geometry as long as the geometry supports point membership classification (PMC) query i.e. whether a point is located inside or outside the physical domain. Fig. 1 illustrates the fictitious domain concept that lies at the heart of the FCM.

The embedding domain $\Omega_e$ consists of the physical domain of interest $\Omega$ and the fictitious domain extension $\Omega_{\text{fict}}$. Analogous to classical FEM, the first step in FCM is the discretization of the variational (or weak) form for the boundary value problem under consideration. Let us consider the standard elastostatics problem for the purpose of illustration. The FCM variational form for this problem reads as follows:

$$\text{find } u \in H^1(\Omega_e) \text{ such that}$$

$$\int_{\Omega_e} \alpha \sigma : \delta \varepsilon d\Omega_e - \int_{\Omega_e} \alpha \delta u \cdot b \, d\Omega_e - \int_{\Gamma_N} \delta u \cdot t \, d\Gamma_N \forall \delta u \in H^1(\Omega_e)$$

$$u = u_0 \forall x \in \Gamma_D$$

(1)

(2)

$\sigma$, $b$ and $t$ are respectively the Cauchy stress tensor, the body force vector and the surface traction vector. Neumann boundary conditions are also specified over the boundary of the embedding domain $\partial \Omega$, where $\sigma \cdot n = 0$ by definition. Dirichlet boundary conditions are specified over $\Gamma_D$ of the physical domain by the prescribed value of the field variable $u_0$. The scalar factor $\alpha$ is defined as follows

$$\alpha(x) = \begin{cases} 
1.0 & \forall x \in \Omega \\
10^{-q} & \forall x \in \Omega_{\text{fict}}.
\end{cases}$$

(3)

In $\Omega_{\text{fict}}$, $\alpha$ must be chosen as small as possible, but large enough to prevent extreme ill-conditioning of the stiffness matrix [22,23]. Typical values of $\alpha$ range between $10^{-5}$ and $10^{-10}$. This scalar parameter $\alpha$ ensures that we are actually solving the boundary value problem on $\Omega$ and not on the extended domain $\Omega_e$. Using a structured grid of higher-order elements (as shown in Fig. 1), the field variable $u$ and its variation $\delta u$ are discretized using polynomial basis functions resulting in a linear system $[K][u] = [f]$. The global stiffness $[K]$ and global force vector $[f]$ are
obtained by assembling the following cell stiffness matrix \((\mathbf{K}_c)\) and force vector \((\mathbf{f}_c)\) for all the cells \((c = 1 \text{ to } n_c)\):

\[
\mathbf{k}_c = \int_{\Omega_c} \mathbf{B}_c^T \alpha \mathbf{C}_c \mathbf{B}_c d\Omega_c, 
\]

\[
\mathbf{f}_c = \int_{\Omega_c} \alpha \mathbf{N}_c^T \mathbf{b}_c d\Omega_c + \int_{\Gamma_y} \mathbf{N}_c^T \mathbf{t}_c d\Gamma_y.
\]

\(\mathbf{B}_c, \mathbf{C}_c, \mathbf{N}_c, \mathbf{t}_c\) and \(\mathbf{b}_c\) are respectively the strain–displacement matrix, constitutive matrix, shape function matrix, traction vector and body force vector associated with the cell \(c\). Solving the above linear system results in the nodal displacements \(\mathbf{u}\). As can be seen from the above equations, the imposition of Neumann boundary conditions requires the computation of integrals over the surface of \(\Omega\). This can be easily accomplished by local surface meshing. However, enforcement of Dirichlet boundary conditions (Eq. (2)) is not straightforward primarily owing to the non-conformity of the mesh and/or the basis functions not satisfying the Kronecker delta property\(^1\) [32]. Lot of research have gone into the enforcement of Dirichlet boundary conditions in meshfree methods. Thus, there are a number of schemes such as the Lagrangian Multiplier Method [33,34], Penalty Method [35,2,36], Augmented Lagrangian Method [32], Nitsche’s Method [37–40] and Kantorovich’s Approach [15,14,12,13] that are available to weakly or strongly enforce Dirichlet boundary conditions.

Notice that the formulation of stiffness matrix and the load vector requires volumetric integration of discontinuous functions (due to \(\alpha\)) over a box domain (see Eqs. (4) and (5)). In other words, due to the introduction of the \(\alpha\) factor in the weak form of the FCM, we have effectively transformed the integral of a continuous integrand over the arbitrary original domain \(\Omega\) into an integral of a discontinuous integrand over a box domain \(\bigcup_{i=1}^{n_c} \Omega_i\). Then, the integral over each of these cells can be computed by employing the classical Gaussian quadrature in combination with a hierarchical partitioning scheme [23,41] such as a quadtree [42,43] (2D) or octree [44,23,42,43] (3D). This approach of integrating a continuous integrand over an arbitrary domain by transforming it into an integral of a discontinuous integrand over a regular domain is called as the characteristic function approach and is the popular integration scheme employed in FCM [45]. Although this transformation is mathematically perfectly legitimate and sensible, it poses a serious problem when evaluated numerically owing to the discontinuity of \(\alpha\). This is because classical Gauss quadrature schemes are designed assuming that the integrand is a polynomial of some degree \([46]\). For instance, \(n\) point Gauss quadrature gives accurate results for integral of polynomials of degree up to \(2n - 1\) in 1D. If the same integration rule is applied to a non-polynomial function, it gives a value of the integral of a polynomial approximation to the integrand. Thus, the accuracy of the integrand by polynomials determines the integration accuracy. It is also well known that polynomial approximation of discontinuous functions tend to oscillate in the neighborhood of the discontinuity. For example, plots in Fig. 2 illustrate polynomial approximation of a Heaviside function by polynomials of 4th (Fig. 2(a)) and 20th (Fig. 2(b)) degree on the segment \([1, 1]\). If we raise the degree of an approximating polynomial, the amplitude of oscillations raises as well \([47]\). This implies that for the characteristic function method to produce accurate results one often needs a very fine partitioning of the domain leading to excessive boundary cell fragmentation as shown in Fig. 3. This makes the method prohibitively expensive for realistic 3D CAD models such as sculptures, engines and bones. Thus, numerical integration \emph{is one of the major bottlenecks of FCM (or any IB method for that matter) and hence will be the focus of this paper.}

In this context, the authors have recently developed a novel integration scheme called the ‘Adaptively Weighted Numerical Integration’ (AW) [48] that could be advantageously employed in any IB method. In AW scheme, given an arbitrary domain \(\Omega \) with a set of appropriately chosen quadrature nodes and order of integration, we compute the quadrature weights by solving a system of linear moment fitting equations \([49–51]\) for an appropriate set of basis functions in the least square sense. Setting up the moment-fitting equations involves computing integrals of the basis functions (known as moments) over an arbitrary geometric domain. This task itself is non-trivial, because it either entails some kind of domain decomposition, or can be reduced to repeated boundary integration as was proposed in [52] for bivariate domains. Adaption of the latter approach avoids excessive domain fragmentation, but leads to a significant computational overhead, particularly in 3D. We overcome this challenge by first computing the moments over a simpler domain \(\Omega_0\), usually a polygon/polyhedron, that is homeomorphic to and is a reasonable approximation

\(^1\) A set of basis functions \(\eta_i(x, j)\), associated to a set of nodes or particles \(x_i\), is said to satisfy the Kronecker delta property if \(\eta_i(x_j) = \delta_{ij} \forall i, j \) [32].
(in a Hausdorff distance sense) of the original domain $\Omega$. The computed moments are then corrected for the deviation from $\Omega$ based on first-order Shape Sensitivity Analysis (SSA) [53]. The moments over approximate domain are further reduced to boundary integrals by the application of divergence theorem [54–58]. The approximate moment fitting equations, thus obtained, can then be easily solved for quadrature weights that adapt to the original geometric domain—hence the name Adaptively Weighted Numerical Integration (AW). The resulting weights are not exact, but are accurate enough to integrate functions over any arbitrary domain when they are well approximated by the chosen set of basis functions.

In this paper, we examine the suitability/efficacy of the AW method in the context of FCM. Specifically, we compare the convergence/performance of AW and the characteristic function method [45] in solving 2D/3D elastostatic problems using the FCM. For this purpose, we have implemented the AW algorithm in the open source FCM framework in MATLAB called the FCMLab [59]. From the experiments, we observe that AW has superior order of convergence and also requires fewer subdivisions/less time to achieve a given accuracy compared to the characteristic function method. In addition, AW is found to offer a great deal of flexibility in the choice of quadrature points and basis functions that is not usually available for other integration schemes such as the characteristic function method.

1.1. Outline

The rest of the paper is organized as follows, In Section 2, we present a detailed discussion of different integration schemes that are available to perform integration over arbitrary geometric domains using non-conforming meshes. In
Section 3, we review the AW scheme. In Section 4, we discuss the implementation aspects of AW in both 2D and 3D. Section 5 presents a comparison between AW and the characteristic function method in solving linear elastostatic and polynomial integration problems in two and three dimensions. It is followed by conclusions and scope for further research in Section 6.

2. Background—numerical integration

As already pointed out in the introduction, formulation of stiffness matrix and the load vector requires volumetric integration of continuous/discontinuous functions over arbitrary domains. In certain IB methods such as the FCM [22, 23], embedded domain method [9, 10] and fictitious domain method [6–8], it is required to integrate discontinuous function over a simple (box-like) domain. In other IB methods such as the solution structure method [11–15] it is required to integrate continuous functions over arbitrary domains. However, current methods to integrate discontinuous functions over regular domains and continuous functions over an arbitrary domain are both computationally expensive for complex geometric domains (especially in 3D). Thus, there is a necessity for the integration procedure to effectively adapt to the geometry of the domain in addition to adapting to the integrand. The issues related to integrand adaptivity is well understood (e.g., see [46]). On the other hand, geometric adaptivity is far from complete and will be the main focus here. Hence, in this section, we will survey relevant ideas and literature on numerical integration techniques with emphasis on geometric adaptivity. We will closely follow [48] for the classification of currently existing integration schemes and its exposition.

2.1. Polygonal/Polyhedral domains

2.1.1. Simple shapes

Numerical integration over a regular n-dimensional interval, i.e. a box, is well understood (see for e.g. [60]). Stroud [61] describes cubature rules for a number of different simple domains (such as hexagon, octahedron and cubical shell) in his book four decades ago. Since then, many additional specialized cubature formulas have been developed for a variety of simple shapes and spaces [62–64].

2.1.2. A priori known functions—Divergence Theorem

Several authors [54, 56, 55, 57, 58] have advocated the use of divergence theorem to integrate symbolically integrable functions over polygonal/polyhedral domains. Here, application of divergence theorem (once for polygonal domain and twice for polyhedron domain) converts the 2D/3D integral into an 1D integral over the edges of the polygon/polyhedron. The resulting integral is then efficiently evaluated using 1D Gauss quadrature rules. For the purpose of illustration let us consider the integration of a priori known function \( b_i(X, Y) \) over the polygonal domain \( \Omega_0 \subset \mathbb{R}^2 \). Our objective is to simplify \( \iiint_{\Omega_0} b_i(X, Y) dX dY \) to an integral over the polygonal boundary. In order to apply the divergence theorem, we first require a vector function \( \phi^i = \phi^i_X(X, Y) \hat{i} + \phi^i_Y(X, Y) \hat{j} \) such that the following is satisfied

\[
\iint_{\Omega_0} b_i(X, Y) dX dY = \iint_{\Omega_0} \nabla \cdot (\phi^i) dX dY.
\]  

(6)

It can be easily proved [58] that for \( \phi^i(X, Y) = \oint b_i(X, Y) dX \hat{i} + 0 \hat{j} \) the above is satisfied. Now, application of divergence theorem to the RHS of Eq. (6) yields

\[
\iint_{\Omega_0} b_i(X, Y) dX dY = \int_{\Gamma_0} \phi^i_X N_X d\Gamma_0
\] 

(7)

\( N_X = \vec{N} \cdot \hat{n} \) is the X-component of the normal to the boundary. For a polygon, the normal is constant over its edges. Hence, for a polygon with \( n_e \)-edges (i.e. \( \Gamma_0 = \bigcup_{k=1}^{n_e} E_0^k \)) one could rewrite the above more explicitly as

\[
\iint_{\Omega_0} b_i(X, Y) dX dY = \sum_{k=1}^{n_e} \int_{E_0^k} \phi^i_X(X, Y) N_X^k dE_0^k.
\]  

(8)
The same equations may be written in a more compact form as

\[ \frac{\partial}{\partial t} \frac{\partial}{\partial x} \frac{\partial}{\partial y} \frac{\partial}{\partial z} \text{ where} \]

\[ \text{with} \]

Domains. In general, a quadrature in \( R \) basis functions to obtain a meshless cubature method for convex, nonconvex and even multiply connected polygonal

\[ 2.2. \text{A priori unknown functions—moment fitting} \]

For a priori unknown functions, Sommariva and Vianello [65] used Green’s integral formula with thin-plate splines basis functions to obtain a meshless cubature method for convex, nonconvex and even multiply connected polygonal domains. In general, a quadrature in \( R^d \) is a formula of the form:

\[ \sum_{i=1}^{n} w_i f(x_i) \approx \int_{\Omega} W(x) f(x) d\Omega \]

where \( \Omega \subset \mathbb{R}^d \) is the integration region, \( f \) is an integrand defined on \( \Omega \), and \( W(x) \) is a non-negative weight function \( (W(x) = 1 \text{ in this paper}) \). A quadrature rule is determined by points \( x_i \in \mathbb{R}^d \) that are usually called quadrature nodes, and the quadrature weights \( w_i \) [51]. A standard technique for constructing quadrature rules is to solve the following system of moment fitting equations [49–51]

\[
\begin{bmatrix}
\int_{\Omega} W(x)b_1(x) d\Omega \\
\int_{\Omega} W(x)b_2(x) d\Omega \\
\vdots \\
\int_{\Omega} W(x)b_m(x) d\Omega \\
\end{bmatrix} = 
\begin{bmatrix}
b_1(x_1) & b_1(x_2) & \cdots & b_1(x_n) \\
b_2(x_1) & b_2(x_2) & \cdots & b_2(x_n) \\
\vdots & \vdots & \ddots & \vdots \\
b_m(x_1) & b_m(x_2) & \cdots & b_m(x_n) \\
\end{bmatrix} 
\begin{bmatrix}
w_1 \\
w_2 \\
\vdots \\
w_n \\
\end{bmatrix} .
\]

The same equations may be written in a more compact form as

\[ [M]_{m \times 1} = [A]_{m \times n} [W]_{n \times 1} \]

with

\[ [M] = \begin{bmatrix}
\int_{\Omega} W(x)b_1(x) d\Omega \\
\int_{\Omega} W(x)b_2(x) d\Omega \\
\vdots \\
\int_{\Omega} W(x)b_m(x) d\Omega \\
\end{bmatrix} \]
where \( \{ b_i \}_{i=1}^m \) is the set of basis functions\(^2\) and \( \{ \mathbf{M} \} \) is the vector of moments defined over the domain of integration \( \Omega \).

The quadrature nodes and the corresponding weights \( \{ x_i, w_i \}_{i=1}^n \) can be determined numerically by solving the above set of moment fitting equations. The resulting quadrature rule can be used to integrate any function that is in the function space spanned by these basis functions, assuming that the integral of basis functions and solution to moment fitting equations were computed exactly. Each integration point in \( d \)-dimensions contributes \( d + 1 \) unknowns: \( d \) coordinate components and one weight. Thus, \( \lceil \frac{m}{d+1} \rceil \) could serve as an estimate for the number of points required to integrate \( m \) basis functions in \( d \) dimensions [67]. Further, Xiao and Gimbutas [51] devised a numerical algorithm for constructing efficient, higher-order quadratures in two and higher dimensions by using moment fitting equations along with node elimination scheme. This algorithm was successfully applied to integrate general functions over triangle and square [51].

Mousavi et al. [68,69] used the same algorithm to construct efficient quadrature rules for bivariate polynomials to integrate functions over convex and non-convex polygons. Specifically, Mousavi and Sukumar [67] constructed efficient quadrature for integration over convex polygons and polyhedra based on moment fitting equations and monomial basis functions. They used Lasserre’s method [70,71] for the integration of homogeneous functions arising in the moment computations. The authors also observed that integration of the known basis functions may be transformed to boundary integrals using divergence theorem. Recently, Joulaian et al. [72] applied this idea to the leaf cells of octree and demonstrated its application in the context of FCM. However, in practice, repeated and accurate integration over the boundary of an arbitrary domain is computationally expensive.

### 2.3. Adaptation by hierarchical partitioning

A common approach to geometric adaptation is hierarchical partitioning. In its simplest form, a quadtree [73] or octree [44] subdivision breaks the domain into non-conforming cells recursively. This kind of subdivision primarily leads to two types of integration cells: interior cells and boundary cells (see Fig. 4(a)). Interior cells can be easily handled using ordinary lattice rules (also called the Cartesian product rule). Several approaches to dealing with boundary cells have been proposed. For example:

(a) The simplest method is to randomly include/exclude boundary cells (see Fig. 4(b)) and then use ordinary lattice rules on the randomly included cells [74].

(b) Alternatively, one could map the boundary cells to smoothly deformed rectangles [14] or other canonical regions [75] and then use ordinary lattice rules on it.

(c) More recently, convenient space parametrization of boundary cells was found to produce good results [76,11,47].

(d) Monte-Carlo based methods [46] can be used to accurately integrate the boundary cells.

(e) Finally, level-set based approaches [77,78] to deal with boundary cells.

All of the proposed approaches to dealing with boundary cells are problematic. Specifically, referring to the above, (a) lacks accuracy, (b) does not generalize to 3D, (b) and (d) are computationally expensive, (b) requires function evaluation outside of the domain, (e) is specific to a particular representation of the domain and (c) involves heuristic steps.

### 2.4. Adaptation by characteristic function

The problem of geometric adaptivity can be reduced in to integrand adaptivity over regular rectangular/box domain. Thus, well known integrand adaptivity techniques (over regular domains) can be used to compute the integral over arbitrary domains. To be precise, let \( f \) be the function to be integrated over the arbitrary domain \( \Omega \subset D \), where \( D \) is a regular domain. Let us define a characteristic (Heaviside) function \( H(x) \):

\[
H(x) = \begin{cases} 
0 & : x \not\in \Omega \\
1 & : x \in \Omega 
\end{cases}
\]

so that \( \int_{\Omega} f(x) d\Omega = \int_{D} f(x) H(x) dD \).

\(^2\)Popular choices of basis functions include the bivariate polynomials \( x^p y^r \) \( (p + r \leq o) \) and trivariate polynomials \( x^p y^r z^s \) \( (p + r + s \leq o) \) in 2D and 3D respectively (for the given integration order \( o \)). Other recommended choices of basis functions include Legendre and Chebyshev polynomials [66].
Now we have transformed the original integration domain $\Omega$ of the integral (Fig. 5(a)) into a regular box domain $D$ (Fig. 5(b)). Thus, the required integral can be computed by sampling the modified function $f(x)H(x)$ over the extended domain $D$ using the ordinary lattice rules. Although this transformation is mathematically perfectly legitimate and sensible, it poses a serious problem when evaluated numerically owing to the discontinuity of the Heaviside function ($H(x)$). Variations of this method is particularly popular in IB methods. In particular, in FCM, the Heaviside function of Eq. (15) is replaced by the $\alpha$ function as given by Eq. (3) in order to avoid ill-conditioning of the stiffness matrix as explained in Section 1. The integration is then performed by hierarchical partitioning of the arbitrary domain. A higher order quadrature is usually employed in the leaf cells to resolve the discontinuity introduced by the $\alpha$ function. However, in general, this turns out to be computationally expensive. Hence, a recent survey [45] discusses a number of strategies employed in Finite Cell Method [22] in dealing with the discontinuity of the $\alpha$ function. These include local meshing [79–81], heuristic modification of integration weights [82,45], smoothed step function approach [83,84], and equivalent polynomial approximation combined with moment fitting approach [85,45]. These methods help alleviate the problem but at additional computational cost and without solving the underlying fundamental issue.

3. Adaptively weighted numerical integration

Adaptively weighted numerical integration (AW) method [48] was developed by the authors specifically for integration of a priori unknown functions over arbitrary domains using a non-conforming grid. We begin by noting that the moment equations (Eq. (12)) are not easy to solve as they form a system of non-linear equations. This is because $[A]$ is non-linear in the position of quadrature points and the unknowns of these equations include the position of quadrature points in addition to the weights. Thus, these unknowns can only be determined by solving a non-linear system of equations using non-linear solution techniques (such as the Newton–Raphson method [46]) that are computationally expensive. However, if we fix the position of the integration points as in [67], the equations become linear in weights and hence can be solved easily. Knowing $\{M\}$, one could solve the moment equations for weights using linear least squares as

$$\{W\}_{n \times 1} = [A^\dagger]_{n \times m} \{M\}_{m \times 1}$$

(16)

$[A^\dagger]$ is the Moore–Penrose pseudo inverse [86]. One popular stable algorithm to numerically compute $[A^\dagger]$ is based on the QR factorization [87].

Thus, given an arbitrary domain $\Omega$ with a set of appropriately chosen quadrature nodes and order of integration, quadrature weights are obtained in this method by solving a system of linear moment fitting equations [49–51] for an
appropriate set of basis functions in the least square sense. Setting up the moment-fitting equations involves computing integrals of the basis functions (known as moments) over an arbitrary geometric domain. This task itself is non-trivial, because it either entails some kind of domain decomposition, or can be reduced to repeated boundary integration as was proposed in [52, 72]. Adaption of the latter approach avoids excessive domain fragmentation, but leads to a significant computational overhead, particularly in 3D.

This challenge is overcome by first computing the moments over a simpler domain \( \Omega_0 \), usually a polygon/polyhedron, that is homeomorphic to and is a reasonable approximation (in a Hausdorff distance sense) of the original domain \( \Omega \) (see Fig. 6). The computed moments are then corrected for the deviation from \( \Omega \) based on first-order Shape Sensitivity Analysis (SSA) [53]. Thus,

\[
M_i \approx \int_{\Omega_0} b_i(X) \, d\Omega_0 + C
\]

with the correction term \( C \) defined as:

\[
C = t \int_{\Omega_0} \nabla \cdot (b_i(X) \hat{V}(X)) \, d\Omega_0 \\
= \int_{\Omega_0} \nabla \cdot (b_i(X)V(X)) \, d\Omega_0 \\
= \int_{\Gamma_0} b_i(X)V_N(X) \, d\Gamma_0,
\]

where \( V = t\hat{V} \) is the design velocity and \( V_N(X) = V(X) \cdot N(X) \) is the normal component of the design velocity at a point \( X \) on the boundary \( \Gamma_0 \) of the reference (piecewise linear) domain \( \Omega_0 \). The correction term \( C \) can be easily derived by the straightforward application of Shape Sensitivity Analysis [53] (for details refer to [48]). The moments over approximate domain are further reduced to boundary integrals by the application of divergence theorem [54, 56, 55, 57, 58]. Thus, the final approximation of \( M_i \) is given by replacing the first term of Eq. (17) by an equivalent boundary integral (Eq. (10)) and the second term by its equivalent expression from Eq. (18) to obtain an approximation to the moment over arbitrary domain \( \Omega \) as

\[
M_i \bigg|_{\Omega} \approx \sum_{k=1}^{n_w} \frac{1}{I_0^k} \int_{\Gamma_0^k} [\beta_X^i(X)N_X^k + b_t(X)V_N(X)] \, d\Gamma_0^k \quad \text{for } i \in \{1, 2, \ldots, m\}.
\]

The resulting boundary integral can be evaluated accurately and efficiently. For 2D domains, approximate \([M]\) could be obtained from Eq. (19) simply by employing the usual Gaussian quadrature over the edges of the polygon. Similarly, integration over 3D domains bounded by triangles or convex quadrilaterals is straightforward. For more general 3D domains, we proceed in two steps. First, we compute the appropriate weights for a chosen set of quadrature nodes for each of the polygonal faces of the approximating polyhedron by solving the moment fitting equations (Eq. (16)) with...
exact \{M\} given by Eq. (10). Then, using the determined weights for the polygonal faces, we obtain approximate \{M\} for the arbitrary 3D domain by numerically evaluating Eq. (19) over the faces of the polyhedron. Using this approximate \{M\}, the approximate weights for the arbitrary domain are obtained by solving the moment fitting equations i.e. Eq. (16).

The second term in the integrand of Eq. (19) is the shape sensitive term that accounts for the deviation of the original domain \(\Omega\) from the polygonal/polyhedral domain \(\Omega_0\). In other words, this term ensures that the weights determined using the moment fitting equations (Eq. (16)) adapt to the domain of integration. In fact, the exact moments for a polygonal/polyhedral domain can be recovered from Eq. (19) by simply omitting the second term. In general, as the polygonal/polyhedral approximation \(\{M\}_0\) approaches (in Hausdorff sense) the boundary of the original domain \(\{M\}\), the second term goes to 0, and the first term approaches the exact moments for the original domain.

Note that the second term requires the computation of \(V_N\). For our purposes, it is sufficient to assume that the design velocity vector \(\hat{V}(X)\) is piecewise continuous over every edge/face \(f^k\) of \(\Omega_0\). Then, the perturbation of the domain by distance \(\gamma\) in the direction normal to the boundary is \(x - X = \gamma N(X)\), which implies that

\[
V_N(X) = V(X).N(X) = t \hat{V}(X).N(X) = t \frac{x - X}{t}.N(X) = \gamma N(X).N(X) = \gamma.
\]

Thus, \(V_N(X)\) is simply the shortest distance (\(\gamma\)) from \(X\) to the original boundary \(\{M\}\) as measured in the normal direction.

The approximate moment fitting equations, thus obtained, can then be easily solved for quadrature weights that adapt to the original geometric domain—hence the name Adaptively Weighted Numerical Integration (AW). The resulting weights are not exact, but are accurate enough to integrate functions over any arbitrary domain when they are well approximated by the chosen set of basis functions. We summarize the AW algorithm in the next subsection.

3.1. AW algorithm

The AW algorithm follows essentially the same steps for 2D and 3D domains, and produces a set of integration nodes and weights that can be used to integrate any sufficiently smooth function over domain \(\Omega\). For the purpose of analysis, we assume that the shape is bounded by a degree \(k\) curve/surface and that \(k\) is known a priori (or at least can be estimated a posteriori). Conceptually, the procedure consists of the following three subtasks that we describe below in detail.

Initialization involves setting the data structures required for solving the system of moment equations (Eq. (16)). Note that the dimensions of the matrix \([A]^{T}\) depend on the choice of basis functions and on the number of integration nodes, both of which depend on the required order of integration. Specifically, initialization consists of the following steps:

1. Choose a piecewise linear domain \(\Omega_0\) (polygon in 2D or polyhedron in 3D) that is homeomorphic to and is a reasonable approximation of the original domain \(\Omega\). There are many ways to accomplish this and we will see some examples in the following section.
2. Determine the order of integration \((o)\) based on the function to be integrated. Choose a set of basis functions \(\{b_i\}_{i=1}^m\) of order up to \(o\). A simple choice is the trivariate polynomials \(x^p y^q z^r\) (3D) or bivariate polynomials \(x^p y^r\) (2D).
3. For the chosen order of integration \((o)\), determine the number and location of quadrature nodes. The minimum number of quadrature points required is dictated by the solvability of moment fitting equations. By having as many equations (basis functions) as the number of unknowns (weights), we make matrix \([A]\) invertible and thus eliminate the need for least squares solution. Hence, the minimum number of quadrature points required is equal to the number of basis functions \((m)\) chosen in the previous step. However, in general, choosing more quadrature points \((n)\) than this minimum number would result in a rectangular system that when solved in a least square sense results in \(n - m\) quadrature points with zero weights. Moment fitting equations also offer a lot of flexibility in positioning the quadrature points. There are a number of ways to choose the quadrature points. One simple scheme is to use the tensor-product rule ensuring all points are inside the domain as explained in [47] (for e.g., see Fig. 5(a)).
In addition, it is best to ensure that all the quadrature nodes lie within the approximate polygon/polyhedron \( \Omega_0 \) as choosing points outside \( \Omega_0 \) will lead to dealing with the discontinuity of an otherwise continuous integrand (as in characteristic function approach) and thereby deteriorating AW’s accuracy.

**Compute moments.** The left hand side of Eq. (13) is a vector of \( m \) moments, one moment for each basis function. The moment computation is a two step procedure and can be summarized as follows

4. Determine the quadrature nodes and weights for each edge (2D) or face (3D) of the simplified domain \( \Omega_0 \). If the edge/face approximates a surface of degree \( 2k - 1 \) in the original boundary \( \Gamma \), choose the integration order to be \( (o + k) \) (for example, \( k = 2 \) for conic sections and quadric surfaces).

5. Using the basis functions selected in step 1, evaluate the approximation to \( \{M\} \) over the original domain \( \Omega \) from Eq. (19), by summing contribution over individual edges (2D) or faces (3D) of the approximate domain \( \Omega_0 \) (using the quadrature nodes and weights determined in step 4).

**Solve for weights and evaluate.**

6. Compute \( [A^\dagger] \) for the preselected set of basis functions (step 1) and quadrature nodes (step 2) using say QR factorization [87].

7. Using the \( \{M\} \) from step 5 and \( [A^\dagger] \) from step 6, compute the approximate weights for the arbitrary domain \( \Omega \) from \( \{W\} = [A^\dagger]\{M\} \) (Eq. (16)).

8. Finally, evaluate the approximation to the required integral by using these weights in Eq. (11) i.e. \( \int_\Omega f(x)\,d\Omega \approx \sum_{i=1}^{n} W_i f(x_i) \).

It should be clear that the same algorithm may be used in most situations, with steps 4 and 5 depending strongly on type, dimension, and representations of the geometric domain \( \Omega \) and its approximation \( \Omega_0 \). This integration scheme could be advantageously employed in cell decomposition methods, for example using quadtrees [73] or octrees [44], to reduce fragmentation. The authors [48] demonstrate the efficacy of this method for integration of polynomial functions over several 2D domains. In this paper, we demonstrate its application and efficacy in the context of FCM in computing the stiffness matrix and force vector arising in 2D/3D elastostatic problems. We also demonstrate its application in general integration problems arising in three dimensions.

### 4. Implementation

The AW algorithm could be implemented in any 2D/3D meshfree FEA system. To demonstrate its application in a meshfree system, we implemented our AW method in FCMLab [59], an open source framework in MATLAB that enables 2D/3D meshfree analysis based on the Finite Cell Method (FCM) [22].

FCMLab currently supports the Penalty Method [35,2,36] and Nitsche’s Method [37–40] for the enforcement of Dirichlet boundary conditions (in weak sense). Furthermore, it supports higher order polynomial basis functions for the discretization of any given field of interest (such as displacements or temperature). Neumann boundary conditions are enforced by means of surface integration over the surface mesh. The domain integration is carried out over a separate integration mesh different from the grid of basis functions. The integration mesh is partitioned using space trees such as quadtrees (2D)/octrees (3D) and classical Gauss quadrature nodes (for boxes) are allocated in each of the integration cells. Depending on whether a quadrature node is inside or outside the domain, appropriate scaling factor \( (\alpha = 1 \text{ or } \alpha = 10^{-6}) \) is used for the domain integrands. In short, FCMLab currently supports volumetric integration based on the characteristic function method (discussed in Section 2.4). For a detailed discussion of the capabilities and design of FCMLab we refer the reader to [59].

We implemented our AW method in the FCMLab framework in order to compute the 2D/3D integrals arising in FCM and in general. In this section, we will give a brief overview of implementation aspects of AW scheme in both two and three dimensions.

#### 4.1. 2D Implementation

For 2D domains, we used a quadtree based integration to compute the desired area integrals. A quadtree decomposition of the domain \( (\Omega) \) results in three types of cells i.e. inner, outer, and leaf (magenta cells in Fig. 9) cells. For
the inner and outer\(^3\) (blue cells in Fig. 9) cells, we allocated quadrature nodes based on the classical Gauss quadrature rule for rectangles [46]. Hence, in this section, we will limit our discussion only to the integration of leaf cells using the AW method.

4.1.1. Initialization

**Approximate polygon (\(\Omega_0\)) construction.** The approximating polygon (\(\Omega_0\)) homeomorphic to the domain (\(\Omega\)) could be constructed in a number of ways including (i) ray casting [89], (ii) coarse quadtree decomposition [73] of \(\Omega\), and (iii) coarse polygonization [90] of \(\Omega\). For the sake of simplicity, we employed ray casting to construct the approximate polygon (\(\Omega_0\)). However, what is needed for applying AW algorithm in a quadtree based integration setting are the polygonal pieces \(\Omega_0^i = I_i \cap \Omega_0\) that approximates \(\Omega^i = I_i \cap \Omega\) for every integration cell \(I_i\). In other words, \(\Omega\) is the union of pieces \(\Omega^i (\Omega = \bigcup_{i=1}^{N_I} \Omega^i)\) that is approximated by \(\Omega_0 = \bigcup_{i=1}^{N_I} \Omega_0^i\). Since, the inner and outer cells are integrated using classical quadrature scheme for boxes, it is enough if we just construct approximate polygonal pieces \(\Omega_0^i\) for the leaf cells.

Specifically, first the leaf cells are classified into one of the 15 marching square cases of Fig. 8 using point membership classification (PMC). For cases 1, 2, 3, 4, 6, 7, 8, 9, 11, 12, 13 and 14, we construct approximate polygons homeomorphic to the leaf cell by casting rays parallel to integration rays and intersecting it with the original domain (\(\Omega\)) as shown in Fig. 7.

The ambiguous cases (case 5 & 10) were handled simply by recursively subdividing the cell until it falls into one of the basic cases (0, 1, 2, 3, 4, 6, 7, 8, 9, 11, 12, 13, 14 and 15) or the size of cells was consistent with the acceptable errors in integration.

**Choice of basis functions.** There are a number of choices for the basis functions in 2D. Some popular choices include bivariate, Legendre and Chebyshev polynomials. In our 2D implementation, for the desired order of integration (\(o\)), we chose a set of bivariate polynomials \(\{x^p y^r | p, r \geq 0, p + r \leq o\}\) as our set of basis functions.

**Quadrature point generation.** Quadrature nodes (in the leaf cells) were generated depending on the type of leaf cell arising in marching squares (see Fig. 8). To be precise, the quadrature nodes for cases 1, 2, 3, 4, 6, 7, 8, 9, 11, 12, 13 and 14 were allocated based on scaled Cartesian product rule via ray casting (see Fig. 7) such that the points lie within both the original domain (\(\Omega\)) and the approximate polygon (\(\Omega_0\)). This is because choosing points outside \(\Omega\) will lead to dealing with the discontinuity of an otherwise continuous integrand (as in characteristic function approach) and thereby deteriorating AW’s accuracy. Moreover, choosing points outside \(\Omega_0\) is known to slightly decrease the accuracy of AW method as was observed by us in our experiments [48]. Typical distribution of quadrature points in a 2D domain for the AW scheme is illustrated in Fig. 9.

---

\(^3\)In principle, we could ignore the outer cells as they do not contribute to the integral. However, we included outer cells in the computation in order to avoid stability issues that may arise when very little of the support of the basis function is in \(\Omega\) (in solving boundary value problems). There are better ways to improve stability as was suggested by Hillig et al. [15] and Babuśka I. and Banerjee U. [88]. However, for the sake of simplicity we did not employ such methods in our implementation.
4.1.2. Moment computations

In order to set up the moment fitting equations (Eq. (12)), it is required to compute the moments (Eq. (19)) and the $[A]$ matrix (of Eq. (12)). The moments (Eq. (19)) are basically boundary integrals over the approximate polygon ($\Omega_0$) and therefore requires computation of line integrals over the edges of $\Gamma_0$. Notice that this can be easily accomplished by employing the classical 1D Gauss quadrature scheme [46] over the edges of the polygon provided the design velocity ($V_N$) is known at each of these 1D quadrature points. Recall from Section 3 that the design velocity ($V_N$) is simply the shortest distance ($\gamma$) from the given point on $\Gamma_0$ (here the 1D quadrature points) to the original boundary ($\Gamma$) as measured in the normal direction. One simple way to compute this distance is by a simple ray casting algorithm [89] as illustrated in Fig. 10. Thus, using the generated quadrature nodes, the approximating polygon and the design velocity, the approximate moments (Eq. (19)) were computed using the bivariate ($x^p y^r$) polynomial basis functions for the desired order of integration. Then, $[A]$ matrix can easily be setup by simply evaluating the chosen set of basis functions (in this case $x^p y^r$) at all the generated quadrature points.
4.1.3. Solving for weights and evaluation

Thus, for every leaf cell, we setup and solve a separate set of moment fitting equations in least square sense (using the QR algorithm [87]) in order to obtain a set of weights that adapt to the geometry in that leaf cell. In other words, we generate a new quadrature rule on the fly for every leaf cell accounting for the type of geometry in that cell. Each of these generated quadrature is then used in Eq. (11) to estimate the desired area integral contribution of the leaf cells.

As mentioned in the beginning of this subsection, inner/outer cell’s contribution was computed using the classical Gauss quadrature scheme for rectangles [46]. Thus, summing up the contributions from all inner, outer and leaf cells gives us an estimation of the desired area integral.

4.2. 3D Implementation

For 3D domains, we used an octree based integration to compute the desired volume integrals. Similar to quadtree decomposition, an octree decomposition of the domain (Ω) results in three types of cells i.e. inner, outer, and leaf cells. The inner and outer cells can be handled by standard Gauss quadrature for boxes [46]. Hence, in this section, we will limit our discussion only to the integration of leaf cells using AW method.

4.2.1. Initialization

**Approximate polyhedron (Ω₀) construction.** The approximating polyhedron Ω₀ homeomorphic to the domain Ω can be constructed in a number of ways including (i) ray casting [89], (ii) coarse octree decomposition [44] of Ω, and (iii) coarse surface meshing [90] of Ω. For the sake of simplicity, we construct this by the polygonization of Ω through an octree decomposition. However, what is needed for applying AW algorithm in an octree based integration setting are the polyhedron pieces Ωᵢ₀ = Iᵢ₁ ∩ Ω₀ that approximates Ω = Iᵢ₁ ∩ Ω for every integration cell Iᵢ₁. In other words, Ω is the union of pieces Ωᵢ (Ω = ∪ᵢ Ωᵢ) that is approximated by Ω₀ = ∪ᵢ Ωᵢ₀. Since, the inner and outer cells are integrated using classical quadrature scheme for boxes, it is enough if we just construct approximate polyhedron pieces Ωᵢ₀ for the leaf cells. Thus, for each leaf cell Iᵢ, we generate the approximate polyhedron Ωᵢ₀ as shown in Fig. 13(b).

To be precise, first the leaf cells are classified into one of the 15 cases of Fig. 11 using point membership classification (PMC). For cases 1, 2, 5, 8, 9, 11 and 14, we first compute the polygonal boundary Γᵢ (magenta triangles in Fig. 13(b)) approximating Γᵢ based on marching cubes algorithm [91]. This polyhedral boundary (Γᵢ) forms only a portion of the approximate polyhedral boundary Γ₀. The other faces of the polyhedron are obtained by trimming the leaf cell Iᵢ by Γᵢᵢ resulting in the polygons (Γᵢᵢ) bounded by red lines in Fig. 13(b)). Thus, Ω₀ = Γᵢ₀ ∪ Γᵢᵢ and Ωᵢ₀ is the volume bounded by this boundary Γ₀. Γᵢᵢ is stored as a bunch of triangles while Γᵢᵢ is stored as a bunch of planar n-gons. The ambiguous and disjoint cases (cases 3, 4, 6, 7, 10, 12 and 13) were handled by recursively subdividing until it falls into one of the above basic cases (case 0, 1, 2, 5, 8, 9, 11 and 14) or the size of cells were consistent with the acceptable errors in integration.

**Choice of basis functions.** There are a number of choices for the basis functions in 3D. Some popular choices include trivariate, Legendre and Chebyshev polynomials. In our 3D implementation, for the desired order of integration (o), we chose a set of trivariate polynomials (\(\sum p r s | s ≥ 0, p + r + s ≤ o\)) as our set of basis functions.

**Quadrature point generation.** There is a great deal of flexibility in the choice of generating quadrature points. One way to generate quadrature points is based on the type of leaf cell arising in marching cubes. However, in 3D, we have 256 marching cube cases to handle. But, by employing rotations and symmetries this can be reduced to the 15 basic cases shown in Fig. 11. The quadrature nodes for the non-ambiguous/non-disjoint cases 1, 2, 5, 8, 9, 11 and 14 were allocated by scaled Cartesian product rule such that the points lie both within the original domain (Ω) and the approximate polyhedron (Ω₀). This is because choosing points outside Ω will lead to dealing with the discontinuity of an otherwise continuous integrand (as in characteristic function approach) and thereby deteriorating AW’s accuracy. Likewise, choosing points outside Ω₀ is known to slightly decrease the accuracy of AW method as was observed by us in [48]. Generation of quadrature nodes in a typical leaf cell is illustrated in Fig. 13(d). For a leaf cell Iᵢ, one could also generate quadrature points randomly such that the points lie both within the approximate polyhedron piece Ωᵢ₀.
Fig. 11. Cases in marching cubes [92].

and the actual domain $\Omega$ as we will see in one of the examples in Section 5.2. The ambiguous/disjoint cases (cases 3, 4, 6, 7, 10, 12 and 13) were handled as before.

Minimum number of quadrature points for AW scheme (in leaf cells) is dictated by the total number of basis functions chosen for the given order $o$. This is because, to make the moment fitting equations solvable we at least need as many equations (basis functions) as the number of unknowns (quadrature weights). For example, if we choose all trivariate polynomials of order up to 3 (i.e. $\{1, x, y, z, x^2, xy, yz, z^2, xz, x^3, x^2y, x^2z, xyz, y^2x, y^2z, y^3, z^2x, z^2y, z^3\}$) as our basis functions, then the minimum number of quadrature points required for AW (per leaf cell) is only 20. However, comparing this with the characteristic function method, we find that one requires at least 27 points ($3 \times 3 \times 3$) to integrate a polynomial of order 3. This is because quadrature nodes in characteristic function method are allocated based on the Cartesian product of 1D Gauss quadrature rule and the smallest cube greater than or equal to 20 is $3^3 = 27$.

Thus, minimum number of quadrature points for AW is equal to the number of complete polynomials of given order $(o)$ and is equal to $\binom{d+o}{d}$ [61] (where $d = 2$ for 2D and $d = 3$ for 3D domains). For characteristic method, it is simply equal to $(o+1)^d$ as it is based on the Cartesian product of 1D Gauss quadrature rule. Fig. 12 shows the savings in quadrature points per leaf cell for the AW method relative to the characteristic function method for different orders of integration. From the plot one can see that, unlike 2D, the savings in 3D is considerable especially for higher orders of integration which is critical from the context of FCM.
4.2.2. Moment computations

In order to set up the moment fitting equations (Eq. (12)), it is required to compute the moments (Eq. (19)) and the $[A]$ matrix (of Eq. (12)). The moments (Eq. (19)) are basically boundary integrals over the approximate polyhedron ($\Omega_0$) and therefore requires computation of surface integrals over the polygonal faces of $\Gamma_0$. Notice that this can be accomplished by first performing a coarse triangulation of the general polygonal faces and then applying the standard quadrature rule for triangles [49]. Further, in order to compute the moments it is required to compute the design velocity ($V_N$) at each of these surface quadrature points. Recall from Section 3 that the design velocity ($V_N$) is simply the shortest distance ($\gamma$) from the give point on $\Gamma_0$ (here the surface quadrature points) to the original boundary ($\Gamma$) as measured in the normal direction. One simple way to compute this distance is by a simple ray casting algorithm [89] as illustrated in Fig. 13(c). Thus, using the generated quadrature nodes, the approximating polyhedron and the design velocity, the approximate moments (Eq. (19)) were computed using the trivariate ($x^p y^r z^k$) polynomial basis functions for the desired order of integration. Then, $[A]$ matrix can easily be setup by simply evaluating the chosen set of basis functions (in this case $x^p y^r z^k$) at all the generated quadrature points.

4.2.3. Solving for weights and evaluation

This step is exactly the same as in 2D and hence we refer the reader to subsection Section 4.1.3 for details.

Note on PMC. There are many fast algorithms for PMC catering to different geometric representations. For geometries represented by implicit functions, PMC is straight forward and so is extremely fast. For geometries
Fig. 14. 2D annular ring problem from [31].

represented by triangular meshes, PMC by ray boundary intersection is rapid since ray–cell intersections have closed form solutions [93]. Even faster techniques take advantage of signed triangles or enclosing tetrahedra sharing a common vertex whose membership is known [94]. For non-tessellated B-reps, PMC is based on Newton-type methods that operate on the parametric geometry describing a solid’s boundary [95]. For a good discussion of PMC algorithms we refer the reader to [96,97].

5. Results and discussions

We now compare the computational properties of AW integration scheme (Section 3) and the characteristic function method (Section 2.4) for various 2D/3D elasticity and polynomial integration problems. We used FCMLab’s [59] implementation of the Finite Cell Method (FCM) to solve the 2D/3D elasticity problems. FCMLab [59] supports quadtree/octree based integration using the characteristic function method. We used this implementation of characteristic function method for the comparisons. We also extended FCMLab [59] to support quadtree/octree based integration using our AW integration scheme.

For polynomial integration problems, we compare the two methods using the relative error measure given by

$$E_r = \left| \frac{I_A - I_C}{I_A} \right| \times 100 \%$$

(21)

$I_A$ and $I_C$ are respectively the analytical and computed (using either AW or characteristic) integral values for the problem under consideration. We also compare the two methods in terms of the $L^2$-norm of the residual error vector for integration of moments up to a given degree. The residual error vector is defined as the difference between the analytical and computed moment vectors. Thus, this error measure is defined as

$$E_m = \| M_A - M_C \|_2$$

(22)

$M_A$ and $M_C$ are respectively the analytical and computed (using either AW or characteristic) moment vectors (given by Eq. (14)). For the elasticity problems, we assess accuracy and convergence of the two integration schemes in the context of FCM using the energy norm given by

$$E_e = \sqrt{\left| \frac{U_{ex} - U_{FCM}}{U_{ex}} \right|} \times 100 \%$$

(23)

$U_{ex}$ and $U_{FCM}$ being the exact and FCM computed strain energies respectively.

5.1. Annular ring

In this example, we will consider the plane stress problem over a 2D annular ring as reported in [31]. The geometry, modeling parameters and the reference solution are given in Fig. 14. The application of Dirichlet boundary conditions for these problems was done in a weak sense using the Nitsche’s method [37–40] with $\beta = 10^3$. The radial displacement and von Mises stress plot for this plane stress problem obtained using the AW integration scheme (with 3 subdivisions) is shown in Fig. 15 and Fig. 16 respectively.
Convergence w.r.t. p-refinement. First, we study the convergence w.r.t. p-refinement for the two integration schemes. Here, p-refinement refers to increasing the polynomial degree ($p$) of the basis functions uniformly from 1 to 8 for an $8 \times 8$ mesh of quad elements. We study the convergence (as measured in energy norm) of the annular ring problem w.r.t. p-refinement for the two integration schemes using various levels of integration tree (quadtree) refinement. We use a $g \times g$ quadrature in each of the integration cells where $g = p + 4$. Fig. 17 shows the convergence of the two schemes w.r.t. p-refinement for various integration tree (quadtree) depths. It is clear from the plot that AW requires just
Fig. 18. Convergence of AW and characteristic function method w.r.t. quadtree depth in energy norm.

Table 1
Rel. error in energy norm and time comparison for AW and Characteristic method w.r.t quadtree refinement.

<table>
<thead>
<tr>
<th>Depth</th>
<th>Characteristic Error (%)</th>
<th>Time (s)</th>
<th>AW Error (%)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1263.7</td>
<td>4.26</td>
<td>5.785</td>
<td>6.06</td>
</tr>
<tr>
<td>2</td>
<td>8.102</td>
<td>11.74</td>
<td>5.796</td>
<td>15.40</td>
</tr>
<tr>
<td>3</td>
<td>5.987</td>
<td>27.24</td>
<td>5.780</td>
<td>35.55</td>
</tr>
<tr>
<td>4</td>
<td>6.048</td>
<td>58.31</td>
<td>5.779</td>
<td>71.86</td>
</tr>
<tr>
<td>5</td>
<td>5.754</td>
<td>104.85</td>
<td>5.790</td>
<td>131.225</td>
</tr>
</tbody>
</table>

2 subdivisions to provide the accuracy of characteristic’s 6 subdivisions. For lower quadtree depths (dep < 5), the integration error dominates to the point that characteristic function method diverges for higher order basis functions (p ≥ 4).

Convergence w.r.t. quadtree refinement. Next, we study the effect of quadtree refinement on the two schemes. For this study, we use a 4 × 4 mesh of quad elements of order 4 (578 dofs), for which we vary the quadtree depth (dep) from 1 to 4. We use 7 × 7 quadrature in each of the integration cells. Fig. 18 shows the convergence of the two schemes w.r.t quadtree refinement in energy norm. It is very clear from Table 1 that characteristic function method requires at least 4 subdivisions more than AW for the error in energy norm to fall below 5.8%. We also measured the total time required to compute stiffness matrix and force vector for this problem. Table 1 lists the computer time and error in energy norm for the two methods. From the table we find that AW is at least 17 times faster than characteristic function method to achieve a given accuracy (of error ≤ 5.8%).

Convergence w.r.t. quadrature rule. Finally, we study the convergence w.r.t quadrature rule. For this study, we use a 4 × 4 mesh of quad elements of order 4 (578 dofs), for which we vary the quadrature rule from g = 1 to g = 10 in each of the integration cells. We use a quadtree depth of 1 for both the schemes. Thus, the total number of integration points for a given quadrature rule (g) is given by 4 × 4 × 4 × g^2 = 64g^2. Fig. 19 shows the convergence of the two schemes w.r.t quadrature rule in energy norm. From the plots, it is clear that for g ≥ 7, the error in energy norm for AW scheme is approximately 6%. However, the characteristic function method diverges for higher order quadrature with the error ranging from 25% to 1230%. This experiment demonstrates why, unlike characteristic function method, AW is a reliable method especially for higher order integration. As already pointed out in the introduction, classical Gauss quadrature schemes are designed assuming that the integrand is a polynomial of some degree [46]. Characteristic function method applies Cartesian product of this 1D integration scheme to integrate a discontinuous (i.e. non-polynomial) integrand giving the value of an integral of its polynomial approximation. However, polynomial approximation of discontinuous functions tends to oscillate in the neighborhood of the discontinuity as was explained
in the introduction (also see Fig. 2). Thus, raising the degree of an approximating polynomial, the amplitude of oscillations raises as well [47] as in Fig. 2. This explains why the characteristic function method diverges for higher order quadrature in this experiment. However, this problem does not arise in our AW integration scheme as we are always dealing with a continuous integrand. Moreover, quadrature rule is carefully generated accounting for the type of integrand and geometric domain.

5.2. Unit sphere

Volume and moment computations. In this example, we will compare the accuracy of our integration scheme in computing the integrals of trivariate polynomials \(x^p y^q z^r\) over a unit sphere using \(3 \times 3 \times 3\) integration points on all the integration cells of the octree. Figs. 20 and 21 show the relative error\(^4\) in computing the volume (\(\int d\Omega\)) and second moment (\(\int x^2 d\Omega\)) over a unit sphere using the two methods. The \(L^2\)-norm of the residual error in integrating all trivariate polynomials of order up to 3 is given in Fig. 22. It takes just 2 subdivisions for AW to drive the error below 0.01%, 0.1% and 0.001 respectively for volume, second moment and residual computations. On the other hand, the characteristic method requires 4 subdivisions to achieve the same level of accuracy clearly owing to the inability of the method to adapt to the underlying geometry.

Tables 2 and 3 list the error and the order of convergence in computing the volume and second moment using the two methods (here \(h\) is the size of the smallest leaf cell). For this problem, we find that AW has a consistent order of

\(^4\) The exact analytical volume (\(\frac{4}{3}\pi\)) and second moment (\(\frac{4}{15}\pi\)) was used as the reference solution in computing these errors.
Fig. 21. Relative error in computing $\int x^2 d\Omega$ over a unit sphere.

Fig. 22. The $L^2$-norm of the residual error in integrating all trivariate polynomials of order up to three ($x^p y^q z^s$ with $p, q, s \geq 0, p + q + s \leq 3$) over a unit sphere.

Table 2

<table>
<thead>
<tr>
<th>Depth</th>
<th>h</th>
<th>Characteristic Volume</th>
<th>Error (%)</th>
<th>Order</th>
<th>AW Volume</th>
<th>Error (%)</th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.25</td>
<td>4.2246228</td>
<td>8.55E−01</td>
<td></td>
<td>4.180840</td>
<td>1.90E−01</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.125</td>
<td>4.1699889</td>
<td>4.48E−01</td>
<td>0.930</td>
<td>4.1884232</td>
<td>8.76E−03</td>
<td>4.444</td>
</tr>
<tr>
<td>3</td>
<td>0.0625</td>
<td>4.1964163</td>
<td>1.82E−01</td>
<td>1.302</td>
<td>4.1887660</td>
<td>5.76E−04</td>
<td>3.925</td>
</tr>
<tr>
<td>4</td>
<td>0.0313</td>
<td>4.1897994</td>
<td>2.41E−02</td>
<td>2.918</td>
<td>4.1887888</td>
<td>3.44E−05</td>
<td>4.064</td>
</tr>
</tbody>
</table>

Convergence close to 4. However, the characteristic function method does not seem to possess a consistent order of convergence. For lower octree depths, it has an order of convergence close to 1 and then it increases to 3 for an octree depth of 4. This suggests that for characteristic function method to produce reliable accurate results one often requires 4 or more subdivisions leading to a substantial increase in the computational cost.

Choice of quadrature points. Finally, we also study the effect of position of quadrature points on the AW method. We consider the following three ways to generate quadrature points in the leaf cells:

1. Scaled Cartesian product rule (Fig. 24).
2. Random points lying inside the domain (Fig. 25).
3. Random points on the polygonal faces of the polyhedral boundary (Fig. 26).
Table 3
Relative error and order of convergence in second moment computations for a unit sphere.

<table>
<thead>
<tr>
<th>Depth</th>
<th>h</th>
<th>Characteristic $\int x^2 d\Omega$</th>
<th>Error (%)</th>
<th>Order</th>
<th>AW $\int x^2 d\Omega$</th>
<th>Error (%)</th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.25</td>
<td>0.8478973</td>
<td>1.210E+00</td>
<td></td>
<td>0.8279464</td>
<td>1.171E+00</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.125</td>
<td>0.8315221</td>
<td>7.444E−01</td>
<td>0.701</td>
<td>0.8371905</td>
<td>6.775E−02</td>
<td>4.112</td>
</tr>
<tr>
<td>3</td>
<td>0.0625</td>
<td>0.8402888</td>
<td>3.021E−01</td>
<td>1.301</td>
<td>0.8377194</td>
<td>4.618E−03</td>
<td>3.875</td>
</tr>
<tr>
<td>4</td>
<td>0.03125</td>
<td>0.8380973</td>
<td>4.050E−02</td>
<td>2.899</td>
<td>0.8377557</td>
<td>2.808E−04</td>
<td>4.040</td>
</tr>
</tbody>
</table>

Fig. 23. The $L^2$-norm of the residual vector in integrating all trivariate polynomials of order up to three ($x^p y^r z^s$ with $p, r, s \geq 0, p + r + s \leq 3$) over a unit sphere for three different quadrature point generation schemes.

Sufficient care is taken to ensure that all generated points lie within the original domain ($\Omega$) and the approximate polyhedral piece $\Omega_0^i$ for any given leaf cell $I_i$.

We compare the $L^2$-norm of the residual vector for the integration of polynomials of order up to 3 over the unit sphere for all the above. From Fig. 23, we find that random domain/boundary points perform slightly better than scaled Cartesian product method for higher octree depths. One main reason for this is that as the leaf cells become smaller and smaller, the quadrature points generated by the scaled Cartesian product method come closer to one another and thereby affecting the condition number of the moment matrix $[A]$. Random distribution of points within the domain/boundary avoids this problem completely resulting in better accuracy even for increased octree depths. Also, there is almost no difference in the residual plot for random domain and boundary points. This observation brings forth an important advantage of the AW method which is the freedom in the choice of quadrature points. Generating quadrature points using scaled Cartesian product rule could be time consuming as it involves ray-triangle (or ray-polygonal) intersection tests. On the other hand, it is a lot easier to generate random points lying within both the original domain ($\Omega$) and the polygonal domain ($\Omega_0^i$) or boundary ($\Gamma_0^i$) as it only involves the inside/outside (PMC) test.

5.3. Cylinder under uniaxial compression

In this example, we will compare our AW method with that of the characteristic function method in solving a 3D uniaxial compression problem. A cylinder of radius 1 m and height 0.5 m is subjected to a compressive pressure of 1 MPa on the top face with the bottom fixed (Fig. 27). We assume a linear isotropic constitutive relationship with $E = 2.05 \times 10^{11}$ N/m² and $\nu = 0.29$ and study the convergence of the two integration schemes with respect to h-refinement using FCM. We use triquadratic hexahedron elements (27 node, 81 DOFs) of various mesh sizes as tabulated in Table 4. For both the methods, $3 \times 3 \times 3$ quadrature points were used in all the integration cells. The strong penalty method [35] with a penalty value of $10^{16}$ was employed to enforce the zero Dirichlet boundary condition on
Fig. 24. Quadrature points generated using scaled Cartesian product rule.

Fig. 25. Quadrature points generated randomly inside the domain.

Fig. 26. Quadrature points generated randomly on the faces of the approximate polyhedron.

Fig. 27. Cylinder under uniaxial compression.
Table 4
Mesh sizes and corresponding DOFs for the uniaxial compression problem.

<table>
<thead>
<tr>
<th>Grid size</th>
<th>DOFs</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 × 2 × 2</td>
<td>375</td>
</tr>
<tr>
<td>3 × 3 × 3</td>
<td>1,029</td>
</tr>
<tr>
<td>4 × 4 × 4</td>
<td>2,187</td>
</tr>
<tr>
<td>5 × 5 × 5</td>
<td>3,993</td>
</tr>
<tr>
<td>7 × 7 × 7</td>
<td>10,125</td>
</tr>
<tr>
<td>9 × 9 × 9</td>
<td>20,577</td>
</tr>
<tr>
<td>10 × 10 × 10</td>
<td>27,783</td>
</tr>
<tr>
<td>12 × 12 × 12</td>
<td>46,875</td>
</tr>
<tr>
<td>14 × 14 × 14</td>
<td>73,167</td>
</tr>
</tbody>
</table>

The convergence plot with respect to global h-refinement for various octree depths in computing the stiffness matrix of FCM using the two integration schemes is shown in Fig. 28. We find that **AW exhibits monotonic convergence whereas characteristic function’s convergence is oscillatory in nature**. It is also clear from Fig. 28 that the error plot for the characteristic function method oscillates about the error plot for our AW method (blue lines in Fig. 28). Moreover, the characteristic function method requires at least 5 subdivisions (red lines in Fig. 28) for the oscillations to settle down and come closer to the accuracy of AW’s 2 subdivisions.

It is also clear from Fig. 29 that **AW requires just one subdivision to achieve the same level of accuracy as that of characteristic’s five subdivision**. Moreover, Fig. 29 also suggests the oscillatory nature of convergence of the characteristic function method w.r.t octree refinement. This is further confirmed from the von Mises stress plots obtained for the two methods for various levels of octree refinement shown in Figs. 31 and 32. The reference von Mises stress plot obtained from an overkill FEM solution in SOLIDWORKS [98] is shown in Fig. 30. Fig. 31 clearly shows spurious stresses at the edges of the top face due to the inability of the characteristic integration scheme to resolve the geometry at lower levels of refinement. On the other hand, from Fig. 32 it is clear that AW does not exhibit such a phenomena and the von Mises stress plots for all three subdivisions are quite similar to one another. The reason for this is that even though we use a coarse polyhedral approximation (Ω₀) at lower subdivisions (obtained by ray casting in 2D and polygonization in 3D), the incorporation of the shape sensitive term in the moment fitting equations
suitably accounts for the difference between $\Omega_0$ and $\Omega$ and thereby enabling a good approximation of the moment integral resulting in accurate geometrically adaptive integration weights. This in turn implies that, unlike AW method, one often requires more than 4 subdivisions for the characteristic function method to produce reliable accurate results for realistic problems with complicated geometry in 3D. Table 5 lists the time required to compute the stiffness matrix in generating the converged points of the h-refinement convergence graph (Fig. 28) for both the methods. From the table we find that AW is at least 43–60 times faster than characteristic function method for this problem. For the AW scheme to generate adaptive weights, on an average, 72% of the time is spent in quadrature point generation (via scaled Cartesian product rule), 8% in approximate polyhedra ($\Omega_0$) construction, 18.5% in setting up the linear system ($\{M\}$ and $[A]$) and 1.5% in solving the linear system. However, by using random quadrature point generation schemes (as discussed in Section 5.2) the total time for AW can be reduced even further. This suggests that, unlike the characteristic function method, AW provides a scalable/viable solution for integration over complex 3D domains.

6. Conclusions and future work

We successfully demonstrated the application of a new integration scheme called the Adaptively Weighted Numerical Integration (AW) in the context of FCM. We implemented the method in FCMLab [59] and performed convergence studies for 2D plane stress, 3D isotropic elasticity and 3D volume/moment computations. The experimental results clearly indicates that, to achieve a given accuracy, AW requires far fewer subdivisions/less time compared to the standard integration scheme (characteristic function approach) used in FCM. In addition,
Fig. 31. von Mises stress plot (top face) for the uniaxial cylinder problem obtained from (a) one (b) two and (c) three octree subdivisions using the characteristic integration scheme (4 × 4 × 4 mesh) in FCMLab [59].

Fig. 32. von Mises stress plot (top face) for the uniaxial cylinder problem obtained from (a) one (b) two and (c) three octree subdivisions using the AW integration scheme (4 × 4 × 4 mesh) in FCMLab [59].

Table 5
Time required to generate the stiffness matrix in generating the converged points for various mesh sizes.

<table>
<thead>
<tr>
<th>Grid size</th>
<th>Time (s)</th>
<th>Speed up</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Characteristic</td>
<td>AW</td>
</tr>
<tr>
<td>2 × 2 × 2</td>
<td>714.13</td>
<td>11.90</td>
</tr>
<tr>
<td>3 × 3 × 3</td>
<td>1540.32</td>
<td>29.30</td>
</tr>
<tr>
<td>4 × 4 × 4</td>
<td>2733.53</td>
<td>54.35</td>
</tr>
<tr>
<td>5 × 5 × 5</td>
<td>4199.05</td>
<td>82.41</td>
</tr>
<tr>
<td>7 × 7 × 7</td>
<td>8590.39</td>
<td>183.82</td>
</tr>
<tr>
<td>9 × 9 × 9</td>
<td>14291.23</td>
<td>327.35</td>
</tr>
<tr>
<td>10 × 10 × 10</td>
<td>19705.23</td>
<td>411.55</td>
</tr>
</tbody>
</table>

AW is found to possess superior order of convergence compared to the characteristic function approach. Moreover, the characteristic function approach is found to produce erroneous results at lower levels of refinement for certain problems in addition to exhibiting oscillatory convergence. Reliable and consistent results from characteristic function method can be expected only with 4 or more subdivisions. This is essentially because of the inability of the method to resolve the geometry at lower grid resolutions. This makes the characteristic function approach computationally very expensive for problems with complicated geometry such as the human femur model shown in Fig. 33. On the other hand, the introduction of shape sensitive term in the computation of moments in AW makes the computed quadrature weights adapt to the domain of integration. This explains why AW produces accurate results even at lower grid resolution. AW is found to exhibit monotonic convergence for integration over simple geometries and we anticipate similar behavior even for more complex geometries. Moreover, AW offers a lot of flexibility in the choice of quadrature points as was demonstrated in one of the examples. Likewise, AW offers freedom in the choice of basis functions as...
its formulation is based on the moment fitting equations. We also note that AW was formulated irrespective of how the original domain $\Omega$ is represented, and it can be used with or without an integration mesh. This suggests that AW is not only applicable to FCM but also to other immersed boundary and meshfree methods.

One limitation of the AW method is the inability of it to automatically deal with small features. The presence of small features such as voids or inclusions smaller than an integration cell (in both 2D and 3D) may not be discovered by this method. Fig. 34 illustrates this situation in which the void is completely situated inside the cell. Since all vertices of a cell are inside the geometric domain, this void is missed during the construction of approximate polygon/polyhedron, and, as a result, the integration accuracy will deteriorate. There are several ways to detect such small features. If the user knows that small features exist in the domain geometry, an initial grid with tighter spacing (Fig. 35(a)) can be provided to ensure a cell corner intersects the feature. This is a simple way to globally ensure that features down to a given size are detected and accounted for. However, because the increased grid resolution applies everywhere in the domain, including regions where it is not needed, it can unnecessarily drive up the computational cost. This approach can be optimized if an additional information about location of small geometric features is
available. In this case, as it is illustrated by Fig. 35(b) and Fig. 35(c), a denser grid is imposed within a cell or a non-uniform adaptive subdivision can be done depending on the location of small features.

Future work will focus on a priori and a posteriori error estimation for the AW method. In addition, studying the computational properties of Legendre and Chebyshev polynomial basis functions in moment fitting equations (for arbitrary 2D/3D domains) is an important step towards improving the accuracy and relevance of AW method to meshfree analysis.

Acknowledgments

This research was supported by the National Science Foundation grants CMMI-1344205 and CMMI-1361862 and National Institute of Standards and Technology grants 70NANB14H248 and 70NANB14H232. The responsibility for errors and omissions lies solely with the authors.

References


[92] MARCHING CUBES. http://users.polytech.unice.fr/~lingrand/MarchingCubes/algo.html. (Online; accessed 07.06.16).


