Journal of Scientific Computing (2018) 77:1339–1370
https://doi.org/10.1007/s10915-018-0802-y

Fast Numerical Integration on Polytopic Meshes with Applications to Discontinuous Galerkin Finite Element Methods

Paola F. Antonietti[1] · **Paul Houston**[2] · **Giorgio Pennesi**[1]

Received: 12 January 2018 / Revised: 26 July 2018 / Accepted: 31 July 2018 / Published online: 29 August 2018

Abstract
In this paper we present efficient quadrature rules for the numerical approximation of integrals of polynomial functions over general polygonal/polyhedral elements that do not require an explicit construction of a sub-tessellation into triangular/tetrahedral elements. The method is based on successive application of Stokes' theorem; thereby, the underlying integral may be evaluated using only the values of the integrand and its derivatives at the vertices of the polytopic domain, and hence leads to an exact cubature rule whose quadrature points are the vertices of the polytope. We demonstrate the capabilities of the proposed approach by efficiently computing the stiffness and mass matrices arising from hp-version symmetric interior penalty discontinuous Galerkin discretizations of second-order elliptic partial differential equations.

Keywords Numerical integration · Polygonal/polyhedral meshes · hp-discontinuous Galerkin method

Mathematics Subject Classification 65D30 · 65D32 · 65L60

Paola F. Antonietti and Giorgio Pennesi have been partially funded by the SIR Project No. RBSI14VT0S funded by MIUR - Italian Ministry of Education, Universities and Research—and by Fondazione Cariplo and Regione Lombardia research Grant No. 2015-0182. Paola F. Antonietti and Giorgio Pennesi also acknowledge the financial support given by GNCS-INdAM.

✉ Paul Houston
paul.houston@nottingham.ac.uk

Paola F. Antonietti
paola.antonietti@polimi.it

Giorgio Pennesi
giorgio.pennesi@polimi.it

[1] MOX-Laboratory for Modeling and Scientific Computing, Dipartimento di Matematica, Politecnico di Milano, Piazza Leondardo da Vinci 32, 20133 Milan, Italy

[2] School of Mathematical Sciences, University of Nottingham, University Park, Nottingham NG7 2RD, UK

1 Introduction

In recent years the exploitation of computational meshes composed of polygonal and polyhedral elements has become very popular in the field of numerical methods for partial differential equations. Indeed, the flexibility offered by polygonal/polyhedral elements allows for the design of efficient computational grids when the underlying problem is characterized by a strong complexity of the physical domain, such as, for example, in geophysical applications, fluid-structure interaction, or crack propagation problems. Moreover, the possibility to adopt computational meshes with hanging nodes is included in this framework by observing that, for example, a classical quadrilateral element with a hanging node on one of its edges can be treated as a pentagon with two aligned edges. Several conforming numerical discretization methods which admit polygonal/polyhedral meshes have been proposed within the current literature; here, we mention, for example, the Composite Finite Element Method [5,41,42], the Mimetic Finite Difference (MFD) method [4,18–21,44], the Polygonal Finite Element Method [63], the Extended Finite Element Method [37,64], the Virtual Element Method (VEM) [10,11,15–17] and the Hybrid High-Order (HHO) method [33–35]. In the non-conforming setting, we mention Discontinuous Galerkin (DG) methods [1–3,6,9,14,22–25,25], Hybridizable DG methods [29–32], non-conforming VEM [8,13,26], and the Gradient Schemes [36]; here the possibility of defining local polynomial discrete spaces follows naturally with the flexibility provided by polytopic meshes.

One of the key aspects concerning the development of efficient finite element discretizations with polygonal/polyhedral grids is the construction of quadrature formulae for the approximate computation of the terms appearing in the underlying weak formulation. Indeed, the design of efficient quadrature rules for the numerical computation of integrals over general shaped polytopes is far from being a trivial task. The classical and most widely employed technique for the integration over polytopes is the *Sub-Tessellation* method, cf. [38,51,62]; here, the domain of integration is subdivided into standard-shaped elements, such as triangular/quadrilateral elements in 2D or tetrahedral/hexahedral elements in 3D, whereby standard efficient quadrature rules are employed, cf. [50,60,70], and also [71] and [48], for an interpolation technique based on the same idea. On the one hand this technique is easy to implement, however, it is generally computationally expensive, particularly for high order polynomials, since the number of function evaluations may be very large.

For this reason, the development of quadrature rules that avoid sub-tessellation is an active research field. Several approaches have been proposed; in particular, we mention [43,54,67,68], for example. One interesting method in this direction is represented by the *Moment Fitting Equation* technique, firstly proposed by Lyness and Monegato in [49], for the construction of quadrature rules on polygons featuring the same symmetry as the regular hexagon. Generalizations to convex and non-convex polygons and polyhedra were proposed by Mousavi et al. in [53]. Here, starting from an initial quadrature rule, given, for example, by the sub-tessellation method described above, an iterative node elimination algorithm is performed based on employing the least-squares Newton method [69] in order to minimise the number of quadrature points while retaining exact integration. Further improvements of the moment fitting equation algorithm can also be found in [52] and [61]. While this method is optimal with respect to the number of function evaluations, the nodes and weights must be stored for every polygon, thus affecting memory efficiency. An alternative approach designed to overcome the limitations of the sub-tessellation approach is based on employing the generalized version of Stokes' theorem; here, the exploitation of Stokes' theorem reduces the integral over a polytope to an integration over its boundary; see [66] for details. For the

two-dimensional case, in [59], Sommariva and Vianello proposed a quadrature rule based on employing Green's theorem. In particular, if an x- or y-primitive of the integrand is available (as for bivariate polynomial functions), the integral over the polygon is reduced to a sum of line integrations over its edges. When the primitive is not known, this method does not directly require a sub-tessellation of the polygon, but a careful choice of the parameters in the proposed formula leads to a cubature rule that can be viewed as a particular sub-tessellation of the polygon itself. However, it is not possible to guarantee that all of the quadrature points lie inside the domain of integration. An alternative and very efficient formula has been proposed by Lasserre in [47] for the integration of *homogeneous functions* over convex polytopes. This technique has been recently extended to general convex and non-convex polytopes in [27]. The essential idea here is to exploit the generalized Stokes' theorem together with Euler's homogeneous function theorem, cf. [58], in order to reduce the integration over a polytope only to boundary evaluations. The main difference with respect to the work presented in [59] is the possibility to apply the same idea recursively, leading to a quadrature formula which exactly evaluates integrals over a polygon/polyhedron by employing only point-evaluations of the integrand and its derivatives at the vertices of the polytope.

In this article we extend the approach of [27] to the efficient computation of the volume/face integral terms appearing in the discrete weak formulation of second-order elliptic problems, discretized by means of high-order DG methods. We point out that our approach is completely general and can be directly applied to other discretization schemes, such as VEM, HHO, Hybridisable DG, and MFD, for example. We focus on the DG approach presented in [25], where the local polynomial discrete spaces are defined based on employing the bounded box technique [39]. We show that our integration approach leads to a considerable improvement in the performance compared to classical quadrature algorithms based on sub-tessellation, in both two- and three-dimensions. The outline of this article is as follows: in Sect. 2 we recall the work introduced in [27], and outline how this approach can be utilized to efficiently compute the integral of d-variate polynomial functions over general polytopes. In Sect. 3 we introduce the interior penalty DG formulation for the numerical approximation of a second-order diffusion–reaction equation on general polytopic meshes. In Sect. 4 we outline the exploitation of the method presented in Sect. 2 for the assembly of the mass and stiffness matrices appearing in the DG formulation. Several two- and three-dimensional numerical results are presented in Sect. 5 in order to show the efficiency of the proposed approach. Finally, in Sect. 6 we summarise the work undertaken in this article and discuss future extensions.

2 Integrating Polynomials over General Polygons/Polyhedra

In this section we review the procedure introduced by Chin et al. in [27] for the integration of homogeneous functions over a polytopic domain. To this end, we consider the numerical computation of $\int_{\mathscr{P}} g(\mathbf{x})\mathrm{d}\mathbf{x}$, where

– $\mathscr{P} \subset \mathbb{R}^d$, $d = 2, 3$, is a closed polytope, whose boundary $\partial\mathscr{P}$ is defined by m $(d-1)$-dimensional *faces* $\mathcal{F}_i$, $i = 1, \ldots, m$. Each *face* $\mathcal{F}_i$ lies in a hyperplane $\mathcal{H}_i$ identified by a vector $\mathbf{a}_i \in \mathbb{R}^d$ and a scalar number b_i, such that

$$\mathbf{x} \in \mathcal{H}_i \iff \mathbf{a}_i \cdot \mathbf{x} = b_i, \quad i = 1, \ldots, m. \tag{1}$$

We observe that $\mathbf{a}_i$, $i = 1, \ldots, m$, can be chosen as the unit outward normal vector to $\mathcal{F}_i$, $i = 1, \ldots, m$, respectively, relative to $\mathscr{P}$, cf. Figs. 1 and 2.

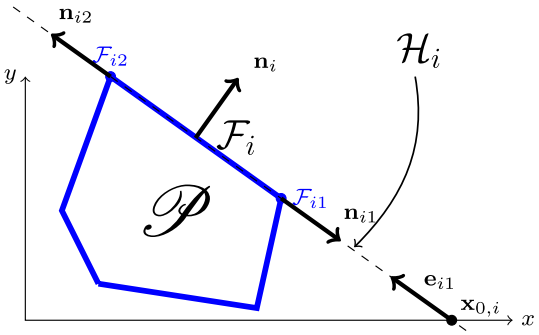

Fig. 1 Example of a two-dimensional polytope $\mathscr{P}$ and its face $\mathcal{F}_i$. The hyperplane $\mathcal{H}_i$ is defined by the local origin $\mathbf{x}_{0,i}$ and the vector $\mathbf{e}_{i1}$

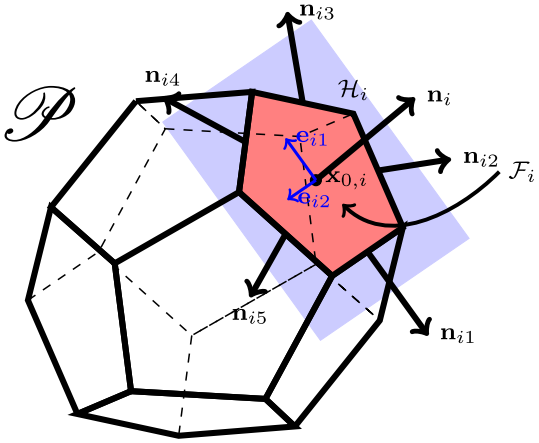

Fig. 2 The dodecahedron $\mathscr{P}$ with pentagonal faces and the face $\mathcal{F}_i \subset \partial\mathscr{P}$ with unit outward normal vector $\mathbf{n}_i$. Here, $\mathcal{F}_i$ has five edges $\mathcal{F}_{ij}$, $j = 1, \ldots, 5$, and five unit outward normal vectors $\mathbf{n}_{ij}$, $j = 1, \ldots, 5$, lying on the plane $\mathcal{H}_i$. The hyperplane $\mathcal{H}_i$ is identified by the local origin $\mathbf{x}_{0,i}$ and the orthonormal vectors $\mathbf{e}_{i1}, \mathbf{e}_{i2}$

– $g : \mathscr{P} \to \mathbb{R}$ is a *homogeneous function* of degree $q \in \mathbb{R}$, i.e., for all $\lambda > 0$, $g(\lambda\mathbf{x}) = \lambda^q g(\mathbf{x})$ for all $\mathbf{x} \in \mathscr{P}$.

We recall that Euler's homogeneous function theorem [58] states that, if g is a *homogeneous function* of degree $q \geq 0$, then the following identity holds:

$$q\ g(\mathbf{x}) = \nabla g(\mathbf{x}) \cdot \mathbf{x} \quad \forall \mathbf{x} \in \mathscr{P}. \tag{2}$$

Next we introduce the generalized Stokes' theorem, which can be stated as follows, cf. [66]: given a generic vector field $\mathbf{X} : \mathscr{P} \to \mathbb{R}^d$, the following identity holds

$$\int_{\mathscr{P}} (\nabla \cdot \mathbf{X}(\mathbf{x}))g(\mathbf{x})\mathrm{d}\mathbf{x} + \int_{\mathscr{P}} \nabla g(\mathbf{x}) \cdot \mathbf{X}(\mathbf{x})\mathrm{d}\mathbf{x} = \int_{\partial\mathscr{P}} \mathbf{X}(\mathbf{x}) \cdot \mathbf{n}(\mathbf{x})g(\mathbf{x})\mathrm{d}\sigma, \tag{3}$$

where $\mathbf{n}$ is the unit outward normal vector to $\mathscr{P}$ and $\mathrm{d}\sigma$ denotes the $(d-1)$-dimensional (surface) measure. Selecting $\mathbf{X} = \mathbf{x}$ in (3), and employing (2), gives

$$\int_{\mathscr{P}} g(\mathbf{x})\mathrm{d}\mathbf{x} = \frac{1}{d+q} \int_{\partial\mathscr{P}} \mathbf{x} \cdot \mathbf{n}(\mathbf{x})g(\mathbf{x})\mathrm{d}\sigma = \frac{1}{d+q} \sum_{i=1}^{m} b_i \int_{\mathcal{F}_i} g(\mathbf{x})\ \mathrm{d}\sigma. \tag{4}$$

Equation (4) states that if g is *homogeneous*, then the integral of g over a polytope $\mathscr{P}$ can be evaluated by computing the integral of the same function over the boundary faces $\mathcal{F}_i \subset \partial\mathscr{P}$, $i = 1, \ldots, m$. By applying Stokes' theorem recursively, we can further reduce each term $\int_{\mathcal{F}_i} g(\mathbf{x})\mathrm{d}\sigma$, $i = 1, \ldots, m$, to the integration over $\partial\mathcal{F}_i$, $i = 1, \ldots, m$, respectively. To this

end, Stokes' theorem needs to be applied on the hyperplane $\mathcal{H}_i,\ i = 1,\dots,m$, in which each $\mathcal{F}_i,\ i = 1,\dots,m$, lies, respectively. In order to proceed, let $\boldsymbol{\gamma} : \mathbb{R}^{d-1} \to \mathbb{R}^d$ be the function which expresses a generic point $\tilde{\mathbf{x}} = (\tilde{x}_1,\dots,\tilde{x}_{d-1})^\top \in \mathbb{R}^{d-1}$ as a point in $\mathbb{R}^d$ that lies on $\mathcal{H}_i,\ i = 1,\dots,m$, i.e.,

$$\tilde{\mathbf{x}} \longmapsto \boldsymbol{\gamma}(\tilde{\mathbf{x}}) = \mathbf{x}_{0,i} + \sum_{n=1}^{d-1} \tilde{x}_n \mathbf{e}_{in}, \quad \text{with } \mathbf{e}_{in} \in \mathbb{R}^d, \ \ \mathbf{e}_{in} \cdot \mathbf{e}_{im} = \delta_{nm}.$$

Here, $\mathbf{x}_{0,i} \in \mathcal{H}_i,\ i = 1,\dots,m$, is an arbitrary point which represents the origin of the coordinate system on $\mathcal{H}_i$, and $\{\mathbf{e}_{in}\}_{n=1}^{d-1}$ is an orthonormal basis on $\mathcal{H}_i,\ i = 1,\dots,m$; see Figs. 1 and 2 for two- and three-dimensional examples, respectively. Notice that $\mathbf{x}_{0,i}$ does not have to lie inside $\mathcal{F}_i,\ i = 1,\dots,m$. Let $\widetilde{\mathcal{F}}_i \subset \mathbb{R}^{d-1}$ such that $\boldsymbol{\gamma}(\widetilde{\mathcal{F}}_i) = \mathcal{F}_i,\ i = 1,\dots,m$, then the following identity holds:

$$\int_{\mathcal{F}_i} g(\mathbf{x}) \mathrm{d}\sigma = \int_{\widetilde{\mathcal{F}}_i} g(\boldsymbol{\gamma}(\tilde{\mathbf{x}})) \mathrm{d}\tilde{\mathbf{x}}, \quad i = 1,\dots,m. \tag{5}$$

Before outlining the details regarding the recursive application of the Stokes' Theorem to (4), we first require the following lemma.

Lemma 1 *Let $\mathcal{F}_{ij} \subset \partial\mathcal{F}_i\ j = 1,\dots,m_i$, be the vertices/edges of $\mathcal{F}_i,\ i = 1,\dots,m$, for $d = 2, 3$, respectively, and let $\mathbf{n}_{ij}$ be the unit outward normal vectors to $\mathcal{F}_{ij}$ lying in $\mathcal{H}_i$. Moreover, let $\widetilde{\mathcal{F}_{ij}} \subset \partial\widetilde{\mathcal{F}_i}$ be the preimage of $\mathcal{F}_{ij}$ with respect to the map $\boldsymbol{\gamma}$, and $\widetilde{\mathbf{n}}_{ij}$ be the corresponding unit outward normal vector. Then, the following holds*

$$\tilde{\mathbf{n}}_{ij} = \underline{\mathbf{E}}^\top \mathbf{n}_{ij}, \quad i = 1,\dots,m, \ \ j = 1,\dots,m_i,$$

where $\underline{\mathbf{E}} \in \mathbb{R}^{d\times(d-1)}$, whose columns are the vectors $\{\mathbf{e}_{in}\}_{n=1}^{d-1},\ i = 1,\dots,m$.

Proof We first note that employing the definition of $\boldsymbol{\gamma}$ we have that

$$\mathbf{x} = \boldsymbol{\gamma}(\tilde{\mathbf{x}}) = \mathbf{x}_{0,i} + \underline{\mathbf{E}}\tilde{\mathbf{x}}, \quad \text{i.e.,} \quad \mathbf{x} - \mathbf{x}_{0,i} = \underline{\mathbf{E}}\tilde{\mathbf{x}}. \tag{6}$$

The proof now follows immediately from simple linear algebra considerations; for full details, we refer to [7]. □

Given identity (5) and Lemma 1, we can prove the following result.

Proposition 1 *Let $\mathcal{F}_i,\ i = 1,\dots,m$, be a face of the polytope $\mathscr{P}$, and let $\mathcal{F}_{ij},\ j = 1,\dots,m_i$, be the planar/straight faces/edges such that $\partial\mathcal{F}_i = \cup_{j=1}^{m_i}\mathcal{F}_{ij}$ for some $m_i \in \mathbb{N}$. Then, for any homogeneous function g, of degree $q \geq 0$, the following identity holds*

$$\int_{\mathcal{F}_i} g(\mathbf{x}) \mathrm{d}\sigma = \frac{1}{d-1+q} \left(\sum_{j=1}^{m_i} d_{ij} \int_{\mathcal{F}_{ij}} g(\mathbf{x}) \mathrm{d}\nu \ + \ \int_{\mathcal{F}_i} \mathbf{x}_{0,i} \cdot \nabla g(\mathbf{x}) \mathrm{d}\sigma \right),$$

where d_{ij} denotes the Euclidean distance between $\mathcal{F}_{ij}$ and $\mathbf{x}_{0,i}$, $\mathbf{x}_{0,i} \in \mathcal{H}_i$, is arbitrary, $i = 1,\dots,m$, and $\mathrm{d}\nu$ denotes the $(d-2)$-dimensional (surface) measure.

Proof If we denote by $\nabla_i = \left[\frac{\partial}{\partial\tilde{x}_1},\dots,\frac{\partial}{\partial\tilde{x}_{d-1}}\right]^\top$ the gradient operator on the hyperplane $\widetilde{\mathcal{H}}_i,\ i = 1,\dots,m$, with respect to the coordinate system $(\tilde{x}_1,\dots,\tilde{x}_{d-1})$, then, upon application of Stokes' theorem, we have

$$\underbrace{\int_{\widetilde{\mathcal{F}}_i} (\nabla_i \cdot \widetilde{\mathbf{X}}) g(\boldsymbol{\gamma}(\tilde{\mathbf{x}})) \mathrm{d}\tilde{\mathbf{x}}}_{①} + \underbrace{\int_{\widetilde{\mathcal{F}}_i} \widetilde{\mathbf{X}} \cdot \nabla_i g(\boldsymbol{\gamma}(\tilde{\mathbf{x}})) \mathrm{d}\tilde{\mathbf{x}}}_{②} = \underbrace{\int_{\partial\widetilde{\mathcal{F}}_i} \widetilde{\mathbf{X}} \cdot \tilde{\mathbf{n}}\ g(\boldsymbol{\gamma}(\tilde{\mathbf{x}})) \mathrm{d}\nu(\tilde{\mathbf{x}})}_{③}, \tag{7}$$

where $\tilde{\mathbf{n}}$ is the unit outward normal vector of $\widetilde{\mathcal{F}}_i$ and $\widetilde{\mathbf{X}}$ is a vector field on $\mathbb{R}^{d-1}$. Next, we transform (7) back to the original coordinate system. To this end, denoting $\underline{\mathbf{E}} \in \mathbb{R}^{d\times(d-1)}$ to be the matrix whose columns are the vectors $\{\mathbf{e}_{in}\}_{n=1}^{d-1}$, we observe that, if we choose $\widetilde{\mathbf{X}} = \tilde{\mathbf{x}}$, then its divergence is $\nabla_i \cdot \widetilde{\mathbf{X}} = d-1$. Exploiting (6), the term $\nabla_i g(\boldsymbol{\gamma}(\tilde{\mathbf{x}}))$ can be written as follows:

$$\nabla_i g(\boldsymbol{\gamma}(\tilde{\mathbf{x}})) = \begin{bmatrix} \frac{\partial\gamma_1}{\partial\tilde{x}_1} & \frac{\partial\gamma_2}{\partial\tilde{x}_1} & \cdots & \frac{\partial\gamma_d}{\partial\tilde{x}_1} \\ \frac{\partial\gamma_1}{\partial\tilde{x}_2} & \frac{\partial\gamma_2}{\partial\tilde{x}_2} & \cdots & \frac{\partial\gamma_d}{\partial\tilde{x}_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial\gamma_1}{\partial\tilde{x}_{d-1}} & \frac{\partial\gamma_2}{\partial\tilde{x}_{d-1}} & \cdots & \frac{\partial\gamma_d}{\partial\tilde{x}_{d-1}} \end{bmatrix} \begin{bmatrix} \frac{\partial g}{\partial\tilde{x}_1} \\ \frac{\partial g}{\partial\tilde{x}_2} \\ \vdots \\ \frac{\partial g}{\partial\tilde{x}_d} \end{bmatrix} = (\underline{\mathbf{E}}^\top \nabla g)(\boldsymbol{\gamma}(\tilde{\mathbf{x}})). \tag{8}$$

Exploiting (6) and (8), we can write ① and ② as

$$① = (d-1)\int_{\widetilde{\mathcal{F}}_i} g(\boldsymbol{\gamma}(\tilde{\mathbf{x}}))\mathrm{d}\tilde{\mathbf{x}} = (d-1)\int_{\mathcal{F}_i} g(\mathbf{x})\mathrm{d}\sigma, \tag{9}$$

$$\begin{aligned} ② &= \int_{\widetilde{\mathcal{F}}_i} \tilde{\mathbf{x}}^\top \underline{\mathbf{E}}^\top \nabla g(\boldsymbol{\gamma}(\tilde{\mathbf{x}}))\mathrm{d}\tilde{\mathbf{x}} = \int_{\mathcal{F}_i} (\mathbf{x}-\mathbf{x}_{0,i})\cdot\nabla g(\mathbf{x})\mathrm{d}\sigma \\ &= q\int_{\mathcal{F}_i} g(\mathbf{x})\mathrm{d}\sigma \;-\; \int_{\mathcal{F}_i} \mathbf{x}_{0,i}\cdot\nabla g(\mathbf{x})\mathrm{d}\sigma, \end{aligned} \tag{10}$$

respectively. Employing Lemma 1, together with (6), we have that

$$\begin{aligned} ③ &= \sum_{j=1}^{m_i}\int_{\widetilde{\mathcal{F}}_{ij}} \tilde{\mathbf{x}}^\top \tilde{\mathbf{n}}_{ij} g(\boldsymbol{\gamma}(\tilde{\mathbf{x}}))\mathrm{d}\nu(\tilde{\mathbf{x}}) = \sum_{j=1}^{m_i}\int_{\mathcal{F}_{ij}} (\mathbf{x}-\mathbf{x}_{0,i})^\top \underline{\mathbf{E}}\underline{\mathbf{E}}^\top \mathbf{n}_{ij} g(\mathbf{x})\mathrm{d}\nu(\mathbf{x}) \\ &= \sum_{j=1}^{m_i}\int_{\mathcal{F}_{ij}} (\mathbf{x}-\mathbf{x}_{0,i})\cdot\mathbf{n}_{ij} g(\mathbf{x})\mathrm{d}\nu. \end{aligned} \tag{11}$$

We observe that the term $(\mathbf{x}-\mathbf{x}_{0,i})\cdot\mathbf{n}_{ij}$ is constant for any $\mathbf{x}\in\mathcal{F}_{ij}$, and that it represents the Euclidean distance between $\mathcal{F}_{ij}$ and $\mathbf{x}_{0,i}$; thereby, we define $d_{ij} = (\mathbf{x}-\mathbf{x}_{0,i})\cdot\mathbf{n}_{ij}$. From the above identities (9), (10) and (11) we deduce the statement of the Proposition. □

Using Proposition 1, together with Eq. (4), we obtain the following identity

$$\int_{\mathscr{P}} g(\mathbf{x})\mathrm{d}\mathbf{x} = \frac{1}{d+q}\sum_{i=1}^{m}\frac{b_i}{d-1+q}\Big(\sum_{j=1}^{m_i} d_{ij}\int_{\mathcal{F}_{ij}} g(\mathbf{x})\mathrm{d}\nu \;+\; \int_{\mathcal{F}_i}\mathbf{x}_{0,i}\cdot\nabla g(\mathbf{x})\mathrm{d}\sigma\Big), \tag{12}$$

where we recall that $\partial\mathscr{P} = \cup_{i=1}^{m}\mathcal{F}_i$ and $\partial\mathcal{F}_i = \cup_{j=1}^{m_i}\mathcal{F}_{ij}$, for $i = 1,\dots,m$.

Remark 1 If $d=2$, then $\mathcal{F}_{ij}$ is a point and (12) states that the integral of g on $\mathscr{P}$ can be computed by vertex-evaluations of the integrand plus a line integration of the partial derivative of g. If $d=3$ we can apply Stokes' Theorem recursively to $\int_{\mathcal{F}_{ij}} g(\mathbf{x})\mathrm{d}\nu$. Proceeding as before, we get

$$\int_{\mathcal{F}_{ij}} g(\mathbf{x})\mathrm{d}\nu = \frac{1}{d-2+q}\Big(\sum_{k=1}^{m_{ij}} d_{ijk}\int_{\mathcal{F}_{ijk}} g(\mathbf{x})\mathrm{d}\xi \;+\; \int_{\mathcal{F}_{ij}}\mathbf{x}_{0,ij}\cdot\nabla g(\mathbf{x})\mathrm{d}\nu\Big),$$

where $\partial\mathcal{F}_{ij} = \cup_{k=1}^{m_{ij}}\mathcal{F}_{ijk}$, $\mathbf{x}_{0,ij}$ is an arbitrarily chosen origin for $\mathcal{F}_{ij}$, and d_{ijk} is the Euclidean distance between $\mathcal{F}_{ijk}$ and $\mathbf{x}_{0,ij}$.

Table 1 Polytopic domains of integration $\mathcal{E}$ considered in Algorithm 1 as a function of the dimension d

	$N=3$	$N=2$	$N=1$	$N=0$
$d=3$	$\mathcal{E}=\mathscr{P}$ is a polyhedron	$\mathcal{E}=\mathcal{F}_i\subset\partial\mathscr{P}$ is a polygon	$\mathcal{E}=\mathcal{F}_{ij}\subset\partial\mathcal{F}_i$ is an edge	$\mathcal{E}=\mathcal{F}_{ijk}\subset\partial\mathcal{F}_{ij}$ is a point
$d=2$		$\mathcal{E}=\mathscr{P}$ is a polygon	$\mathcal{E}=\mathcal{F}_i\subset\partial\mathscr{P}$ is an edge	$\mathcal{E}=\mathcal{F}_{ij}\subset\partial\mathcal{F}_i$ is a point
$d=1$			$\mathcal{E}=\mathscr{P}$ is an interval	$\mathcal{E}=\mathcal{F}_i\subset\partial\mathscr{P}$ is a point

In view of the application of Proposition 1 to finite element methods, we are interested in the integration of a particular class of *homogeneous functions*, namely *polynomial homogeneous functions* of the form

$$g(\mathbf{x}) = x_1^{k_1} x_2^{k_2} \dots x_d^{k_d}, \quad \text{where } k_n \in \mathbb{N}_0 \text{ for } n = 1, \dots, d.$$

In this case, g is a *homogeneous function* of degree $q = k_1 + \cdots + k_d$, and the general partial derivative $\frac{\partial g}{\partial x_n}$ is a *homogeneous function* of degree $q-1$. With this in mind, it is possible to recursively apply formula (12) to the terms involving the integration of the derivatives of g. To this end, we write $\mathcal{E}\subset\mathbb{R}^d$, $d=2,3$, be a N-polytopic domain of integration, with $N=1,\dots,d$, and let $\partial\mathcal{E}=\cup_{i=1}^m\mathcal{E}_i$, where each $\mathcal{E}_i\subset\mathbb{R}^d$ is a $(N-1)$-polytopic domain. When $N=d$ and $d=2,3$, $\mathcal{E}_i$, $i=1,\dots,m$, will be an edge or a face, respectively; see Table 1 for details. We define the function

$$\mathcal{I}(N,\mathcal{E},k_1,\dots,k_d) = \int_{\mathcal{E}} x_1^{k_1}\dots x_d^{k_d}\,\mathrm{d}\sigma_N(x_1,\dots,x_d), \tag{13}$$

Algorithm 1 $\mathcal{I}(N,\mathcal{E},k_1,\dots,k_d) = \int_{\mathcal{E}} x_1^{k_1}\dots x_d^{k_d}\,\mathrm{d}\sigma_N(x_1,\dots,x_d)$

if $N=0$ ($\mathcal{E}=(v_1,\dots,v_d)\in\mathbb{R}^d$ is a point)

return $\mathcal{I}(N,\mathcal{E},k_1,\dots,k_d) = v_1^{k_1}\cdots v_d^{k_d}$;

else if $1\le N\le d-1$ ($\mathcal{E}$ is a point if $d=1$ or an edge if $d=2$ or a face if $d=3$)

$$\begin{aligned}\mathcal{I}(N,\mathcal{E},k_1,\dots,k_d) = \frac{1}{N+\sum_{n=1}^d k_n}\Big(&\sum_{i=1}^m d_i\,\mathcal{I}(N-1,\mathcal{E}_i,k_1,\dots,k_d)\\ &+ x_{0,1}\,k_1\,\mathcal{I}(N,\mathcal{E},k_1-1,k_2,\dots,k_d)\\ &+\cdots+\ x_{0,d}\,k_d\,\mathcal{I}(N,\mathcal{E},k_1,\dots,k_d-1)\Big);\end{aligned}$$

else if $N=d$ ($\mathcal{E}$ is an interval if $d=1$ or a polygon if $d=2$ or a polyhedron if $d=3$)

$$\mathcal{I}(N,\mathcal{E},k_1,\dots,k_d) = \frac{1}{N+\sum_{n=1}^d k_n}\Big(\sum_{i=1}^m b_i\,\mathcal{I}(N-1,\mathcal{E}_i,k_1,\dots,k_d)\Big).$$

end if

which returns the integral of the polynomial $x_1^{k_1} \dots x_d^{k_d}$ over $\mathcal{E}$, where $\mathrm{d}\sigma_N$ is the N-dimensional (surface) measure, $N = 1, 2, \dots, d$. According to Proposition 1, the recursive definition of the function $\mathcal{I}(\cdot, \cdot, \dots, \cdot)$ is given in Algorithm 1.

Remark 2 With a slight abuse of notation, when $1 \le N \le d-1$, in Algorithm 1 (and for the purposes of the following discussion), the point $\mathbf{x}_0 = (x_{0,1}, \dots, x_{0,d})^\top$ denotes an arbitrarily chosen origin for the coordinate system which defines the N-polytope $\mathcal{E}$ and d_i represents the Euclidean distance between the $(N-1)$-polytopes $\mathcal{E}_i$, which form the boundary of $\mathcal{E}$, and $\mathbf{x}_0$, $i = 1, \dots, m$. Furthermore, in Algorithm 1, b_i, $i = 1, \dots, m$, is the same constant appearing in (1). Here it can be evaluated as $b_i = \mathbf{n}_i \cdot \mathbf{v}$, where $\mathbf{v}$ is a vertex of $\mathcal{E}_i$ and $\mathbf{n}_i$ is the unit outward normal vector, $i = 1, \dots, m$.

Remark 3 We point out that in (12), cf. also (13), the shape of the underlying polytope can be general: indeed, nonconvex simply-connected domains $\mathcal{E}$ are admissable.

2.1 Integration of Bivariate Polynomials over Polygonal Domains

In order to test the performance of the method proposed in Algorithm 1, we consider the integration of bivariate *homogeneous functions* on a given polygon $\mathscr{P} \subset \mathbb{R}^2$ based on using the three different approaches:

A.1 Recursive algorithm described in Sect. 2, based on the formula (13): $\int_{\mathscr{P}} x^k y^l \mathrm{d}\mathbf{x} = \mathcal{I}(2, \mathscr{P}, k, l)$, cf. Algorithm 1.

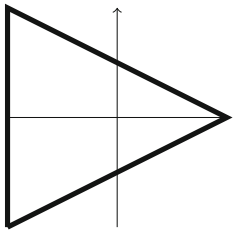

Fig. 3 Triangle ($\mathscr{P}_1$)

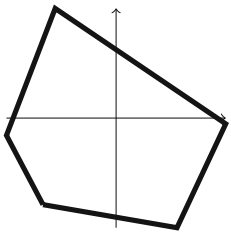

Fig. 4 Irregular polygon with 5 faces ($\mathscr{P}_2$)

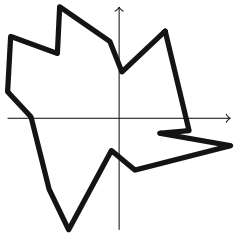

Fig. 5 Irregular polygon with 15 faces ($\mathscr{P}_3$)

A.2 Use of the formula (4) together with numerical integration employed for the evaluation of the edge integrals with known one-dimensional Gaussian quadrature rules, as recently proposed in [28];

A.3 Sub-tessellation technique: the domain of integration $\mathscr{P}$ is firstly decomposed into triangles where standard efficient quadrature rules are then employed.

We test the three different approaches for integrating bivariate polynomials of different polynomial degrees on the triangle depicted in Fig. 3 and the two irregular polygons shown in Figs. 4 and 5, cf. Table 2 for the list of coordinates for each domain; the actual values of the integrals are given in Table 3. In Table 4 we show the average CPU-time taken to evaluate the underlying integral using each method. We point out that, for each integrand and each integration domain $\mathscr{P}$, the relative errors between the output of the three different approaches are of the order of machine precision; that is, all three algorithms return the exact integral up to roundoff error. For completeness, we note that the times for **A.1** include the computation of b_i, $\mathbf{n}_i$, and d_{ij}, $j = 1, \ldots, m_i$, $i = 1, \ldots, m$. For **A.2** we take into account the evaluation of b_i, $\mathbf{n}_i$, $i = 1, \ldots, m$, and the one-time computation of the one-dimensional quadrature defined on $(-1, 1)$, consisting of $\mathcal{N}$ nodes and weights, employed for the line integrations. Here, we select $\mathcal{N} = \lceil \frac{k+l}{2} \rceil + 1$, in order to guarantee the exact integration of $x^k y^l$. The times for **A.3** include the one-time computation of the $\mathcal{N}^2$ nodes and weights on the reference tri-

Table 2 Coordinates of the polygons of Figs. 3, 4, and 5

	Vertex	x-cordinates	y-cordinates
$\mathscr{P}_1$	1	−1.000000000000000	−1.000000000000000
	2	1.000000000000000	0.000000000000000
	3	−1.000000000000000	1.000000000000000
$\mathscr{P}_2$	1	−0.666666666666667	−0.789473684210526
	2	0.555555555555556	−1.000000000000000
	3	1.000000000000000	−0.052631578947368
	4	−0.555555555555556	1.000000000000000
	5	−1.000000000000000	−0.157894736842105
$\mathscr{P}_3$	1	0.413048522141662	0.781696234443715
	2	0.024879797655533	0.415324992429711
	3	−0.082799691823524	0.688810136531751
	4	−0.533191422779328	1.000000000000000
	5	−0.553573605852999	0.580958514816226
	6	−0.972432940212767	0.734117068746903
	7	−1.000000000000000	0.238078507228890
	8	−0.789986179147920	0.012425068086110
	9	−0.627452906935866	−0.636532897516109
	10	−0.452662174765764	−1.000000000000000
	11	−0.069106265580153	−0.289054989277619
	12	0.141448047807069	−0.464417038155806
	13	1.000000000000000	−0.245698820584615
	14	0.363704451489016	−0.134079689960635
	15	0.627086024018283	−0.110940423607648

Table 3 The approximated values of the integral over the three polygons in Figs. 3, 4 and 5 obtained with approach **A.1**

	$\mathscr{P}_1$	$\mathscr{P}_2$	$\mathscr{P}_3$
$\int_{\mathcal{E}} x^5 y^5$	0	-0.0020324991	-0.002589861
$\int_{\mathcal{E}} x^{10} y^{10}$	0.0111339078	$7.4274779926 \times 10^{-5}$	$1.5738050178 \times 10^{-4}$
$\int_{\mathcal{E}} x^{20} y^{20}$	0.0030396808	$6.0738145408 \times 10^{-8}$	$1.3793481020 \times 10^{-6}$
$\int_{\mathcal{E}} x^{40} y^{40}$	$7.9534562047 \times 10^{-4}$	$2.2238524572 \times 10^{-12}$	$4.2588831784 \times 10^{-10}$
$\int_{\mathcal{E}} x^{10} y^5$	0	$-2.0911953867 \times 10^{-4}$	0.0014996521
$\int_{\mathcal{E}} x^{20} y^5$	0	$-1.3797380205 \times 10^{-5}$	$7.0356275077 \times 10^{-4}$
$\int_{\mathcal{E}} x^{40} y^5$	0	$-7.9203571311 \times 10^{-7}$	$2.5065856538 \times 10^{-4}$
$\int_{\mathcal{E}} x^5 y^{20}$	-0.005890191	$8.08469022058 \times 10^{-5}$	$-1.330384913 \times 10^{-4}$
$\int_{\mathcal{E}} x^5 y^{40}$	-0.001868889	$4.37593748009 \times 10^{-5}$	$-3.963064075 \times 10^{-5}$

Table 4 CPU times as a function of the integrand and the integration domain $\mathscr{P}$ for the three approaches **A.1**, **A.2** and **A.3**

	$\mathscr{P}_1$			$\mathscr{P}_2$			$\mathscr{P}_3$		
2pt	**A.1**	**A.2**	**A.3**	**A.1**	**A.2**	**A.3**	**A.1**	**A.2**	**A.3**
$x^5 y^5$	0.054	0.159	0.616	0.083	0.244	0.973	0.227	0.678	2.856
$x^{10} y^{10}$	0.078	0.221	1.359	0.123	0.328	2.321	0.351	0.939	7.301
$x^{20} y^{20}$	0.124	0.344	4.060	0.207	0.540	7.399	0.580	1.498	22.70
$x^{40} y^{40}$	0.208	0.578	14.79	0.377	0.934	27.24	1.073	2.671	86.63
$x^{10} y^5$	0.064	0.191	0.999	0.081	0.296	1.699	0.237	0.833	5.125
$x^{20} y^5$	0.078	0.240	1.955	0.089	0.412	3.690	0.274	1.093	10.99
$x^{40} y^5$	0.107	0.363	4.975	0.085	0.616	9.504	0.332	1.680	29.40
$x^5 y^{20}$	0.052	0.244	1.971	0.085	0.412	3.662	0.243	1.117	11.07
$x^5 y^{40}$	0.051	0.365	5.009	0.082	0.597	9.295	0.272	1.673	29.17

angle, where $\mathcal{N}$ is selected as in **A.2**, the time required for sub-tessellation, as well as the time needed for numerical integration on each sub-triangle. The results shown in Table 4 illustrate that the sub-tessellation approach **A.3** is the slowest while the proposed method **A.1** is the fastest for all of the considered cases; in particular, we highlight that, even for just a single domain of integration, the former method is between one- to two-orders of magnitude slower than the latter approach proposed in this article. Moreover, when the integration domain consists of a triangle, our algorithm **A.1** still outperforms classical quadrature rules, cf. **A.3**, even though in this case no sub-tessellation is undertaken. When comparing **A.1** and **A.2**, we observe that the former algorithm is again superior in terms of CPU time in comparison with the latter approach; this difference seems to grow when the exponents k and l of the integrand function $x^k y^l$ are very different. This is because in **A.1** we have made an optimal selection of the points $\mathbf{x}_{0,i} = (x_{0i,1}, x_{0i,2})^\top$, $i = 1, \ldots, m$, appearing in (12). Indeed, performing the geometric reduction of the edges of the domain of integration, we then choose $x_{0i,1} = 0$ or $x_{0i,2} = 0$, $i = 1, \ldots, m$, if the exponents of the integrand function $x^k y^l$ are $k \geq l$ or $k < l$, respectively. The choice $x_{0i,1} = 0$ or $x_{0i,2} = 0$, $i = 1, \ldots, m$, allows us to avoid the recursive

calls to the function $\mathcal{I}(\cdot,\cdot,\ldots,\cdot)$ related to the x- or y-partial derivatives, respectively. In this way the approach **A.1** is able to exploit the form of the integrand in order to optimize the evaluation of the corresponding integral. To explore this issue further, in the following section we consider the computational complexity of **A.1** in both the cases when an optimal and non-optimal selection of the points $\mathbf{x}_{0,i}$, $i = 1,\ldots,m$, is made.

2.2 Computational Complexity of Algorithm 1

The computational complexity of Algorithm 1, which is employed in **A.1**, depends in general on the number of recursive calls of the function $\mathcal{I}(\cdot,\cdot,\ldots,\cdot)$. In particular, using the shorthand notation introduced in Remark 2, the selection of the points $\mathbf{x}_0 = (x_{0,1},\ldots,x_{0,d})^\top$, which are used to define the origin of the coordinate system of each N-polytope $\mathcal{E}$ which defines the facets of $\mathscr{P}$ is crucial. In general, any $(d-1)$-dimensional hyperplane in $\mathbb{R}^d$ possesses a non-empty intersection with some axis of the Cartesian reference system, which means that it is always possible to choose $(d-1)$ components of $\mathbf{x}_0$ as zero. Without loss of generality we select $x_{0,r} = {}^{b_i}/_{n_{i,r}}$ and $x_{0,s} = 0$ for $s \neq r$, where b_i and $\mathbf{n}_i$ are as defined in Remark 2, and $r \in \{1,\ldots,d\}$ is chosen so that $k_r = \min\{k_1,\ldots,k_d\}$.

Remark 4 In general, if $\mathcal{E} \subset \mathcal{H}$ is a N-polytopic domain in $\mathbb{R}^d$, then at most N components of $\mathbf{x}_0 \in \mathcal{H}$ can be selected to be zero.

In this way, the selection of k_r essentially fixes the number of recursive calls of $\mathcal{I}(\cdot,\cdot,\ldots,\cdot)$ in Algorithm 1. More precisley, we write $|\mathcal{I}(N,\mathcal{E},k_1,\ldots,k_d)|_{Fl}$ to denote the number of FLOPs to perform $\mathcal{I}(N,\mathcal{E},k_1,\ldots,k_d)$, and let C_N be the number of FLOPs required by $\mathcal{I}(N,\mathcal{E},k_1,\ldots,k_d)$, without considering the recursive calls of $\mathcal{I}(\cdot,\cdot,\ldots,\cdot)$ to itself. With this in mind, let us consider the following two examples:

– Set $d = 2$ and assume $k_1 \leq k_2$, so that we can choose $x_{0,1} \neq 0$ and $x_{0,2} = 0$ on each of the edges of $\mathscr{P}$. Then, according to Algorithm 1 we have

$$|\mathcal{I}(2,\mathcal{E},k_1,k_2)|_{Fl} = C_2 + \sum_{i=1}^{m} \underbrace{|\mathcal{I}(1,\mathcal{E}_i,k_1,k_2)|_{Fl}}_{\text{(i)}}$$

and

$$\begin{aligned}
\text{(i)} &= C_1 + \sum_{j=1}^{2} |\mathcal{I}(0,\mathbf{v}_{ij},k_1,k_2)|_{Fl} + |\mathcal{I}(1,\mathcal{E}_i,k_1-1,k_2)|_{Fl} \\
&= C_1 + 2C_0 + C_1 + \sum_{j=1}^{2} |\mathcal{I}(0,\mathbf{v}_{ij},k_1-1,k_2)|_{Fl} + |\mathcal{I}(1,\mathcal{E}_i,k_1-2,k_2)|_{Fl} \\
&= \cdots = k_1(C_1 + 2C_0),
\end{aligned}$$

where we have denoted the vertices of the edge $\mathcal{E}_i$ as $\mathbf{v}_{i1}$ and $\mathbf{v}_{i2}$. Hence,

$$|\mathcal{I}(2,\mathscr{P},k_1,k_2)|_{Fl} = C_2 + mk_1(C_1 + 2C_0) \sim \mathcal{O}(k_1).$$

In general, for $d = 2$ we deduce that

$$|\mathcal{I}(2,\mathscr{P},k_1,k_2)|_{Fl} \sim \mathcal{O}(\min\{k_1,k_2\}). \tag{14}$$

– Set $d = 3$ and assume $k_1 = \min\{k_1, k_2, k_3\}$, so that we may select $x_{0,1} \neq 0$ and $x_{0,2} = x_{0,3} = 0$ on each of the faces of $\mathscr{P}$. Thereby, employing Algorithm 1 we deduce that

$$|\mathcal{I}(3, \mathcal{E}, k_1, k_2, k_3)|_{Fl} = C_3 + \sum_{i=1}^{m} \underbrace{|\mathcal{I}(2, \mathcal{E}_i, k_1, k_2, k_3)|_{Fl}}_{\text{(i)}}$$

where, for each $i = 1, \ldots, m$,

$$\begin{aligned}
\text{(i)} &= C_2 + \sum_{j=1}^{m_i} |\mathcal{I}(1, \mathcal{E}_{ij}, k_1, k_2, k_3)|_{Fl} + |\mathcal{I}(2, \mathcal{E}_i, k_1 - 1, k_2, k_3)|_{Fl} \\
&= 2C_2 + \sum_{j=1}^{m_i} |\mathcal{I}(1, \mathcal{E}_{ij}, k_1, k_2, k_3)|_{Fl} \\
&\quad + \sum_{j=1}^{m_i} (|\mathcal{I}(1, \mathcal{E}_{ij}, k_1 - 1, k_2, k_3)|_{Fl}) + |\mathcal{I}(2, \mathcal{E}_i, k_1 - 2, k_2, k_3)|_{Fl} \\
&= \cdots = k_1 C_2 + \sum_{k=1}^{k_1} \left(\sum_{j=1}^{m_i} |\mathcal{I}(1, \mathcal{E}_{ij}, k, k_2, k_3)|_{Fl} \right)
\end{aligned}$$

Here, the computational complexity of $\mathcal{I}(1, \mathcal{E}_{ij}, k, k_2, k_3)$ depends on the choice of $\mathbf{x}_0 \equiv \mathbf{x}_{0,ij}$ which defines the origin of the coordinate system for $\mathcal{E}_{ij}$, $j = 1, \ldots, m_i$, $i = 1, \ldots, m$. According to Remark 4, two components of $\mathbf{x}_{0,ij}$ can possibly be different from zero, which implies that the complexity of Algorithm 1 increases exponentially when $d = 3$. However, it is possible to modify Algorithm 1 in order to avoid the double recursive calls which cause this exponential complexity. In particular, in Sect. 2.3 we propose an alternative algorithm which exploits the same idea of Algorithm 1 and allows us to overcome this issue.

In order to confirm (14), we use the tool [56] to measure the number of FLOPs required to exactly compute $\int_{\mathscr{P}} x^{k_1} y^{k_2} \mathrm{d}\mathbf{x}$; moreover, comparisons will also be made with **A.3**. To simplify the presentation, the polygon $\mathscr{P}$ is selected to be the triangle with vertices $(-1, 0.3)$,

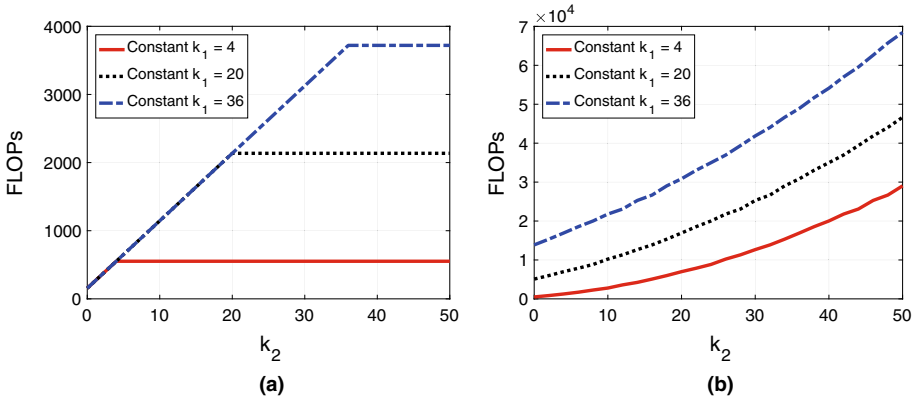

Fig. 6 Comparison of the number of FLOPs required to evaluate $\int_{\mathscr{P}} x^{k_1} y^{k_2} \mathrm{d}\mathbf{x}$, based on fixing k_1 and varying $k_2 \in \{0, \ldots, 50\}$: **a** Quadrature free method **A.1**; **b** Sub-tessellation method **A.3**

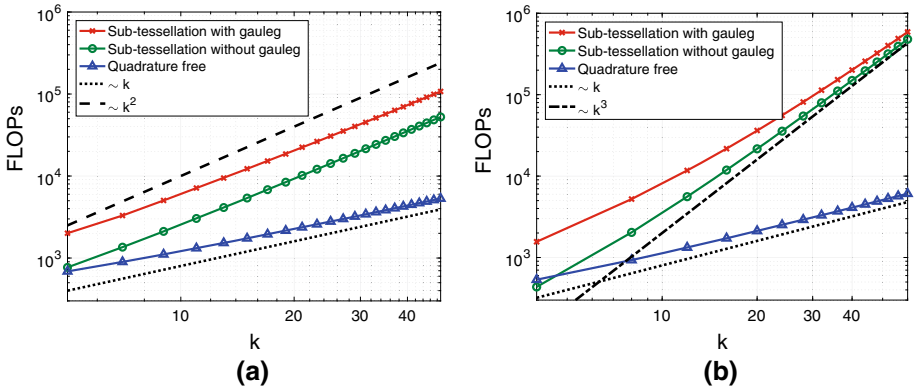

Fig. 7 Comparison of the number of FLOPs required to evaluate $\int_{\mathscr{P}} x^k y^k \mathrm{d}\mathbf{x}$ as k increases employing both the quadrature free and sub-tessellation methods: **a** Excludes function evaluations; **b** Total cost including function evaluations

$(1, -1)$, and $(0.3, 1)$; thereby, **A.3** does not require the computation of a sub-tessellation. In Fig. 6, we plot the number of FLOPs needed to evaluate $\int_{\mathscr{P}} x^{k_1} y^{k_2} \mathrm{d}\mathbf{x}$ by fixing k_1 and varying $k_2 \in \{0, \ldots, 50\}$ employing both **A.1** and **A.3**. In particular, Fig. 6a shows that the number of FLOPs required by the quadrature free method **A.1** growths linearly with respect to k_2 when $k_1 > k_2$ and becomes constant as k_2 increases when $k_1 \leq k_2$. Figure 7 confirms the asymptotic behaviour of the two algorithms in the case when $k_1 = k_2$; here, the number of FLOPs required by the sub-tessellation method is reported in both the case when the cost of the evaluation of the quadrature nodes and weights employing the function `gauleg`, cf. [55], for example, is included/excluded. In particular, we show results both in the case when the cost of the function evaluations is excluded, cf. Fig. 7a, as well as the total number of FLOPs required by each algorithm to exactly evaluate $\int_{\mathscr{P}} x^k y^k \mathrm{d}\mathbf{x}$, cf. Fig. 7b. As expected, the computational complexity of **A.3** grows as $\mathcal{O}(k^2)$ and $\mathcal{O}(k^3)$ in these two latter cases, respectively, while the cost of the quadrature free method is always $\mathcal{O}(k)$ as k increases.

Thus far, the numerical results presented for the proposed quadrature free method have assumed that the points $\mathbf{x}_0$, which are used to define the origin of the coordinate system of each N-polytope $\mathcal{E}$ which defines the facets of $\mathscr{P}$, has been chosen in an optimal manner to ensure that the number of recursive calls of $\mathcal{I}(\cdot, \cdot, \ldots, \cdot)$, cf. Algorithm 1, is minimized. Indeed, a sub-optimal choice of these points leads to an exponential growth in the number of recursive calls of the function $\mathcal{I}(\cdot, \cdot, \ldots, \cdot)$ in Algorithm 1. For example, if $d = 2$ the non-optimal choice of $\mathbf{x}_0$ implies that each call of $\mathcal{I}(\cdot, \cdot, \ldots, \cdot)$ with $N = 1$ leads to a double recursive call of $\mathcal{I}(\cdot, \cdot, \ldots, \cdot)$, up to when a zero exponent k_1 or k_2 appears as input. In particular, if $k_1 = k_2 = k$, it is possible to show that the number of FLOPs required by the quadrature free method grows as $\mathcal{O}(2^{2k-1})$, as k increases, cf. Fig. 8. In the following section, we present an alternative implementation of the quadrature free algorithm which avoids this exponential growth, irrespective of the selection of the points $\mathbf{x}_0$.

2.3 Integration of Families of Monomial Functions

In the context of employing the quadrature free approach within a finite element method, in practice we are not interested in integrating a single monomial function, but instead an entire

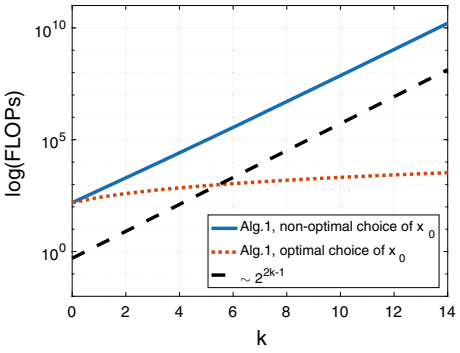

Fig. 8 Number of FLOPs required to evaluate $\int_{\mathscr{P}} x^k y^k \mathrm{d}\mathbf{x}$ as k increases, based on employing the quadrature free method, with a sub-optimal choice of $\mathbf{x}_0$

family of monomials, which, for example, form a basis for the space of polynomials of a given degree over a given polytopic element κ which belongs to the underlying computational mesh. For example, when $d = 2$, let us consider the evaluation of

$$\int_{\kappa} x^{k_1} y^{k_2} \mathrm{d}\mathbf{x} \quad \forall\, k_1, k_2 \geq 0, \quad k_1 + k_2 \leq p. \tag{15}$$

We note that even when employing the Approach **A.1** with an optimal choice of the points $\mathbf{x}_0$, the total number of FLOPs required for the computation of (15) is approximately $\mathcal{O}(p^3)$, as p increases.

To improve the dependence on p we propose an alternative approach, cf. Algorithm 2; this is based on the observation that, using the notation of Algorithm 1, if the values of $\mathcal{I}(N-1, \mathcal{E}_j, k_1, \ldots, k_d), j = 1, \ldots, m, \mathcal{I}(N, \mathcal{E}, k_1 - 1, \ldots, k_d) \ldots \mathcal{I}(N, \mathcal{E}, k_1, \ldots, k_d - 1)$, for $1 \leq N \leq d-1$, in Algorithm 1, have already been computed, then the computation of $\mathcal{I}(N, \mathcal{E}, k_1, \ldots, k_d)$ is extremely cheap. Indeed, since we must store the integrals of all the monomials on κ anyway, we can start by computing and storing $\int_{\kappa} x^{k_1} y^{k_2} \mathrm{d}x_1 \mathrm{d}x_2$ related to the lower degrees k_1, k_2 and $N = 1$, then exploit these values in order to compute the integrals with higher degrees k_1, k_2 and higher dimension N of the integration domain $\mathcal{E}$. This leads to an algorithm, whereby the number of FLOPs required to compute and store $\{\int_{\kappa} x_1^{k_1} \ldots x_d^{k_d} \mathrm{d}\sigma_d(x_1, \ldots, x_d),\ k_1, \ldots, k_d \geq 0,\ k_1 + k_2 + \cdots + k_d \leq p\}$ is of order $\mathcal{O}(p^d)$, as p increases, irrespective of the selection of choice of the points $\mathbf{x}_0$. In Fig. 9 we now compare these two approaches for $d = 2$, when the underlying element is selected to be the triangular region employed in the previous section. Here, we compare Algorithm 1, with an

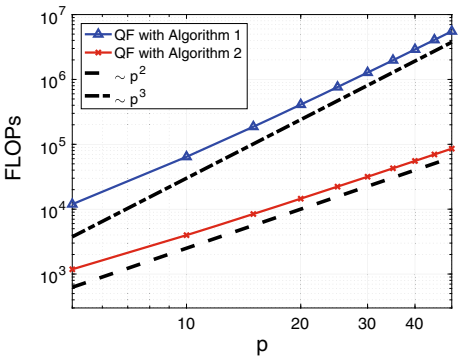

Fig. 9 Number of FLOPs required to evaluate $\{\int_{\kappa} x^{k_1} y^{k_2} \mathrm{d}\mathbf{x}\ \forall\, k_1, k_2 \geq 0,\ k_1 + k_2 \leq p\}$ based on employing Algorithm 1 (with an optimal selection of the points $\mathbf{x}_0$), and Algorithm 2

Algorithm 2 Algorithm for integrating all monomials up to order p

$\partial\mathcal{E} = \{\mathcal{E}_1, \ldots, \mathcal{E}_m\}$ where $\mathcal{E}_i \subset \partial\mathcal{E}$;
$F = FaceIntegrals(d-1, \mathcal{E}_1, \ldots, \mathcal{E}_m, k_1, \ldots, k_d)$;
for $a_1 = 0 : k_1, \ldots, a_d = 0 : k_d; k_1 + k_2 + \ldots + k_d \leq p$ **do**
  $V(a_1, \ldots, a_d) = \frac{1}{d+\sum_{n=1}^{d} a_n} \sum_{i=1}^{m} b_i F(a_1, \ldots, a_d, i)$;
end for
procedure $F = FaceIntegrals(N, \mathcal{E}_1, \ldots, \mathcal{E}_m, k_1, \ldots, k_d)$
  $F(-1 : k_1, \ldots, -1 : k_d, 1 : m) = 0$;
  for i=1:m **do**
    choose $\mathbf{x}_0$ as the first vertex of $\mathcal{E}_i$;
    $\partial\mathcal{E}_i = \{\mathcal{E}_{i1}, \ldots, \mathcal{E}_{im_i}\}$ where $\mathcal{E}_{ij} \subset \partial\mathcal{E}_i,\ j = 1, \ldots, m_i$;
    if N-1>0
      $E = FaceIntegrals(N-1, \mathcal{E}_{i1}, \ldots, \mathcal{E}_{im_i}, k_1, \ldots, k_d)$;
    else if N-1=0 ($\mathcal{E}_{ij} = (v_1, \ldots, v_d) \in \mathbb{R}^d$ is a point)
      $E(a_1, \ldots, a_d, j) = v_1^{a_1} \ldots v_d^{a_d} \quad \forall\, 0 \leq a_n \leq k_n,\ j = 1, \ldots, m_i$;
    end if
    for $a_1 = 0 : k_1, \ldots, a_d = 0 : k_d; k_1 + k_2 + \ldots + k_d \leq p$ **do**

$$F(a_1, \ldots, a_d, i) = \frac{1}{N + \sum_{n=1}^{d} a_n} \Big(\sum_{j=1}^{m_i} d_{ij} E(a_1, \ldots, a_d, j) + x_{0,1} k_1 F(a_1 - 1, \ldots, a_d, i) + \cdots + x_{0,1} k_1 F(a_1, \ldots, a_d - 1, i) \Big);$$

    end for
  end for
end procedure

optimal selection of the points $\mathbf{x}_0$, with Algorithm 2, where in the latter case the points $\mathbf{x}_0$ are simply selected to be equal to the first vertex defining each edge; here, we clearly observe the predicted increase in FLOPs of $\mathcal{O}(p^3)$ and $\mathcal{O}(p^2)$, as p increases, for each of the two algorithms, respectively.

3 Application to *hp*-Version DG Methods

We consider the following elliptic model problem, given by: find u such that

$$\begin{aligned} -\Delta u + u &= f \text{ in } \Omega\ , \\ u &= 0 \text{ on } \partial\Omega\ , \end{aligned} \tag{16}$$

where $\Omega \subset \mathbb{R}^d$, $d = 2, 3$, is a polygonal/polyhedral domain with boundary $\partial\Omega$ and f is a given function in $L^2(\Omega)$.

In order to discretize problem (16), we introduce a partition $\mathcal{T}_h$ of the domain Ω, which consists of disjoint (possibly non-convex) open polygonal/polyhedral elements κ of diameter h_κ, such that $\overline{\Omega} = \bigcup_{\kappa \in \mathcal{T}_h} \bar{\kappa}$. We denote the mesh size of $\mathcal{T}_h$ by $h = \max_{\kappa \in \mathcal{T}_h} h_\kappa$. Furthermore, we define the *faces* of the mesh $\mathcal{T}_h$ as the planar/straight intersections of the $(d-1)$-dimensional facets of neighbouring elements. This implies that, for $d = 2$, a *face* consists of a line segment, while for $d = 3$, the *faces* of $\mathcal{T}_h$ are general shaped polygons; without loss of generality, for the definition of the proceeding DG method we assume that

the faces are $(d-1)$-dimensional simplices, cf. [24,25] for a discussion of this issue. In order to introduce the DG formulation, it is helpful to distinguish between boundary and interior element faces, denoted by $\mathcal{F}_h^B$ and $\mathcal{F}_h^I$, respectively. In particular, we observe that $F \subset \partial\Omega$ for $F \in \mathcal{F}_h^B$, while for any $F \in \mathcal{F}_h^I$ we assume that $F \subset \partial\kappa^{\pm}$, where $\kappa^{\pm}$ are two adjacent elements in $\mathcal{T}_h$. Furthermore, we write $\mathcal{F}_h = \mathcal{F}_h^I \cup \mathcal{F}_h^B$ to denote the set of all mesh faces of $\mathcal{T}_h$. For simplicity of presentation we assume that each element $\kappa \in \mathcal{T}_h$ possesses a uniformly bounded number of faces under mesh refinement, cf. [24,25].

We associate to $\mathcal{T}_h$ the corresponding discontinuous finite element space V_h, defined by $V_h = \{v \in L^2(\Omega) : v|_\kappa \in \mathcal{P}_{p_\kappa}(\kappa), \kappa \in \mathcal{T}_h\}$, where $\mathcal{P}_{p_\kappa}(\kappa)$ denotes the space of polynomials of total degree at most $p_\kappa \geq 1$ on $\kappa \in \mathcal{T}_h$, cf. [24,25].

In order to define the DG method, we introduce the jump and average operators:

$$\begin{aligned}
&[\![\boldsymbol{\tau}]\!] = \boldsymbol{\tau}^+ \cdot \mathbf{n}^+ + \boldsymbol{\tau}^- \cdot \mathbf{n}^-, \qquad \{\boldsymbol{\tau}\} = \frac{\boldsymbol{\tau}^+ + \boldsymbol{\tau}^-}{2}, \qquad F \in \mathcal{F}_h^I,\\
&[\![v]\!] = v^+\mathbf{n}^+ + v^-\mathbf{n}^-, \qquad \{v\} = \frac{v^+ + v^-}{2}, \qquad F \in \mathcal{F}_h^I,\\
&[\![v]\!] = v^+\mathbf{n}^+, \qquad \{\boldsymbol{\tau}\} = \boldsymbol{\tau}^+, \qquad F \in \mathcal{F}_h^B,
\end{aligned} \tag{17}$$

where $v^{\pm}$ and $\boldsymbol{\tau}^{\pm}$ denote the traces of sufficiently smooth scalar- and vector-valued functions v and $\boldsymbol{\tau}$, respectively, on F taken from the interior of $\kappa^{\pm}$, respectively, and $\mathbf{n}^{\pm}$ are the unit outward normal vectors to $\partial\kappa^{\pm}$, respectively, cf. [12].

We then consider the bilinear form $\mathcal{A}_h(\cdot,\cdot) : V_h \times V_h \to \mathbb{R}$, corresponding to the symmetric interior penalty DG method, defined by

$$\begin{aligned}
\mathcal{A}_h(u,v) = &\sum_{\kappa\in\mathcal{T}_h}\int_\kappa \nabla u \cdot \nabla v \,\mathrm{d}\mathbf{x} - \sum_{F\in\mathcal{F}_h}\int_F \left(\{\nabla_h u\}\cdot[\![v]\!] + [\![u]\!]\cdot\{\nabla_h v\}\right)\,\mathrm{d}s\\
&+ \sum_{F\in\mathcal{F}_h}\int_F \alpha_h [\![u]\!]\cdot[\![v]\!]\,\mathrm{d}s,
\end{aligned} \tag{18}$$

where ∇_h denotes the broken gradient operator, defined elementwise, and $\alpha_h \in L^\infty(\mathcal{F}_h)$ denotes the interior penalty stabilization function, whose precise definition, based on the analysis introduced in [24,25], is given below. To this end, we first need the following definition.

Definition 1 Let $\tilde{\mathcal{T}}_h$ be the subset of elements $\kappa \in \mathcal{T}_h$ such that each $\kappa \in \tilde{\mathcal{T}}_h$ can be covered by at most $n_{\mathcal{T}}$ shape-regular simplices $\mathcal{K}_i$, $i = 1,\ldots,n_{\mathcal{T}}$, such that

$$dist(\kappa, \partial\mathcal{K}_i) < C_{as}\,\frac{diam(\mathcal{K}_i)}{p_\kappa^2}, \text{ and } |\mathcal{K}_i| \geq c_{as}|\kappa|$$

for all $i = 1,\ldots,n_{\mathcal{T}}$, for some $n_{\mathcal{T}} \in \mathbb{N}$, where C_{as} and c_{as} are positive constants, independent of κ and $\mathcal{T}_h$.

Given Definition 1, we recall the following inverse inequality, cf. [24,25].

Lemma 2 *Let $\kappa \in \mathcal{T}_h$, $F \subset \partial\kappa$ denote one of its faces, and $\tilde{\mathcal{T}}_h$ be defined as in Definition 1. Then, for each $v \in \mathcal{P}_{p_\kappa}(\kappa)$, we have the inverse estimate*

$$\|v\|^2_{L^2(F)} \leq C_{INV}(p_\kappa,\kappa,F)\frac{p_\kappa^2|F|}{|\kappa|}\|v\|^2_{L^2(\kappa)},$$

where

$$C_{INV}(p_\kappa, \kappa, F) := C_{inv} \begin{cases} \min\left\{ \frac{|\kappa|}{\sup_{\kappa_\flat^F \subset \kappa} |\kappa_\flat^F|}, p_\kappa^{2(d-1)} \right\}, & \text{if } \kappa \in \tilde{\mathcal{T}}_h, \\ \frac{|\kappa|}{\sup_{\kappa_\flat^F \subset \kappa} |\kappa_\flat^F|}, & \text{if } \kappa \in \mathcal{T}_h \setminus \tilde{\mathcal{T}}_h, \end{cases}$$

and $\kappa_\flat^F$ denotes a d-dimensional simplex contained in κ which shares the face F with $\kappa \in \mathcal{T}_h$. Furthermore, C_{inv} is a positive constant, which if $\kappa \in \tilde{\mathcal{T}}_h$ depends on the shape regularity of the covering of κ given in Definition 1, but is always independent of $|\kappa|/\sup_{\kappa_\flat^F \subset \kappa} |\kappa_\flat^F|$, p_κ and v.

Based on Lemma 2, together with the analysis presented in [24,25], the parameter α_h can be defined as follows.

Definition 2 Let $\alpha_h : \mathcal{F}_h \to \mathbb{R}_+$ be defined facewise by

$$\alpha_h(x) := C_\alpha \begin{cases} \max\limits_{\kappa \in \{\kappa^+, \kappa^-\}} \left\{ C_{INV}(p_\kappa, \kappa, F) \frac{p_\kappa^2 |F|}{|\kappa|} \right\}, & x \in F, F \in \mathcal{F}_h^I, F \subset \partial\kappa^\pm, \\ C_{INV}(p_\kappa, \kappa, F) \frac{p_\kappa^2 |F|}{|\kappa|}, & x \in F, F \in \mathcal{F}_h^B, F \subset \partial\kappa, \end{cases}$$

with $C_\alpha > C_\alpha^{min}$, where C_α^{min} is a sufficiently large lower bound.

The DG discretization of the problem (16) is given by: find $u_h \in V_h$ such that

$$\mathcal{A}_h(u_h, v_h) + \int_\Omega u_h v_h \, \mathrm{d}\mathbf{x} = \int_\Omega f v_h \, \mathrm{d}\mathbf{x} \quad \forall v_h \in V_h. \tag{19}$$

By fixing a basis $\{\phi_i\}_{i=1}^{N_h}$, N_h denoting the dimension of the discrete space V_h, (19) can be rewritten as: find $\mathbf{U} \in \mathbb{R}^{N_h}$ such that

$$(\mathbf{A} + \mathbf{M})\mathbf{U} = \mathbf{f}, \tag{20}$$

where $\mathbf{f}_i = \int_\Omega f \phi_i \mathrm{d}\mathbf{x}$ $\forall i = 1, \ldots, N_h$, $\mathbf{A}$ is the stiffness matrix, given by $\mathbf{A}_{ij} = \mathcal{A}_h(\phi_j, \phi_i)$ $\forall i, j = 1, \ldots, N_h$, $\mathbf{M}$ is the mass matrix, and $\mathbf{U}$ contains the expansion coefficients of $u_h \in V_h$ with respect to the chosen basis. In order to assemble $(\mathbf{A} + \mathbf{M})$ we need to compute the following matrices:

$$\mathbf{M}_{i,j} = \int_\Omega \phi_i \phi_j \mathrm{d}\mathbf{x}, \quad \mathbf{V}_{i,j} = \int_\Omega \nabla\phi_i \cdot \nabla\phi_j \, \mathrm{d}\mathbf{x}, \tag{21}$$

$$\mathbf{S}_{i,j} = \sum_{F \in \mathcal{F}_h} \int_F \alpha_h [\![\phi_i]\!] \cdot [\![\phi_j]\!] \mathrm{d}\sigma, \quad \mathbf{I}_{i,j} = \sum_{F \in \mathcal{F}_h} \int_F \{\nabla\phi_i\} \cdot [\![\phi_j]\!] \mathrm{d}\sigma, \tag{22}$$

for $i, j = 1, \ldots, N_h$, where as before N_h denotes the dimension of the DG space V_h. In particular, the stiffness matrix related to the DG approximation of problem (19) is defined as $\mathbf{A} = \mathbf{V} - \mathbf{I}^\top - \mathbf{I} + \mathbf{S}$.

4 Elemental Stiffness and Mass Matrices

In this section, we outline the application of Algorithm 2 for the efficient computation of the mass and stiffness matrices appearing in (20).

4.1 Shape Functions for the Discrete Space V_h

To construct the discrete space V_h we exploit the approach presented in [25], based on employing polynomial spaces defined over the bounding box of each element. More precisely, given an element $\kappa \in \mathcal{T}_h$, we first construct the Cartesian bounding box B_κ, such that $\overline{\kappa} \subset \overline{B_\kappa}$. Given B_κ, $\kappa \in \mathcal{T}_h$, it is easy to define a linear map between B_κ and the reference element $\hat{B} = (-1, 1)^d$ as follow: $\mathbf{F}_\kappa : \hat{B} \to B_\kappa$ such that $\mathbf{F}_\kappa : \hat{\mathbf{x}} \in \hat{B} \longmapsto \mathbf{F}_\kappa(\hat{\mathbf{x}}) = \mathbf{J}_\kappa \hat{\mathbf{x}} + \mathbf{t}_\kappa$, where $\mathbf{J}_\kappa \in \mathbb{R}^{d\times d}$ is the Jacobi matrix of the transformation which describes the stretching in each direction, and $\mathbf{t}_\kappa \in \mathbb{R}^d$ is the translation between the point $\mathbf{0} \in \hat{B}$ and the baricenter of the bounded box B_κ, see Fig. 10. We note that since $\mathbf{F}_\kappa$ affinely maps one bounding box to another (without rotation), the Jacobi matrix $\mathbf{J}_\kappa$ is diagonal.

Employing the map $\mathbf{F}_\kappa$, $\kappa \in \mathcal{T}_h$, we may define a standard polynomial space $\mathcal{P}_p(B_\kappa)$ on B_κ spanned by a set of basis functions $\{\phi_{i,\kappa}\}$ for $i = 1, \ldots, N_{p_\kappa} = \dim(\mathcal{P}_p(B_\kappa))$. More precisely, we denote by $\{\mathcal{L}_n(x)\}_{n=0}^{\infty}$ the family of one-dimensional and L^2-orthonormal Legendre polynomials, defined over $L^2(-1, 1)$, i.e.,

$$\mathcal{L}_n(x) = \frac{L_n(x)}{\|L_n\|_{L^2(-1,1)}}, \quad \text{with } L_n(x) = \frac{1}{2^n n!}\frac{\mathrm{d}}{\mathrm{d}x}\big[(x^2-1)^n\big],$$

cf. [40,57]. We then define the basis functions for the polynomial space $\mathcal{P}_p(\hat{B})$ as follows: writing $I = (i_1, i_2, \ldots, i_d)$ to denote the multi-index used to identify each basis function $\{\hat{\phi}_I\}_{0\le|I|\le p}$, where $|I| = i_1 + \cdots + i_d$, we have that

$$\hat{\phi}_I(\hat{\mathbf{x}}) = \hat{\phi}_I(\hat{x}_1, \ldots, \hat{x}_d) = \mathcal{L}_{i_1}(\hat{x}_1)\mathcal{L}_{i_2}(\hat{x}_2)\cdots\mathcal{L}_{i_d}(\hat{x}_d).$$

Then, the basis functions for the polynomial space $\mathcal{P}_{p_\kappa}(\kappa)$ are defined by using the map $\mathbf{F}_\kappa$, namely:

$$\phi_{I,\kappa}(\mathbf{x}) = \hat{\phi}_I(\mathbf{F}_\kappa^{-1}(\mathbf{x})) \quad \forall \mathbf{x} \in \kappa \subset B_\kappa \quad \forall I : 0 \le |I| \le p_\kappa. \tag{23}$$

The set $\{\phi_{I,\kappa} : 0 \le |I| \le p_\kappa,\ \kappa \in \mathcal{T}_h\}$ forms a basis for the space V_h. On each element $\kappa \in \mathcal{T}_h$ we introduce a bijective relation between the set of multi-indices $\{I = (i_1, \ldots, i_d) : 0 \le |I| \le p_\kappa\}$ and the set $\{1, 2, \ldots, N_{p_\kappa}\}$.

4.2 Volume Integrals Over Polytopic Mesh Elements

In the following we describe the application of Algorithm 2 to compute the entries in the local mass and element-based stiffness matrices

$$\mathbf{M}^\kappa_{i,j} = \int_\Omega \phi_{i,\kappa}\phi_{j,\kappa}\mathrm{d}\mathbf{x}, \quad \mathbf{V}^\kappa_{i,j} = \int_\Omega \nabla\phi_{i,\kappa}\cdot\nabla\phi_{j,\kappa}\mathrm{d}\mathbf{x} \quad i, j = 1, \ldots, N_{p_\kappa}, \tag{24}$$

respectively, for all $\kappa \in \mathcal{T}_h$. For simplicity of presentation, we restrict ourselves to two-dimensions, though we emphasize that the three-dimensional case is analogous, cf. Sect. 5.2 below. Since the basis functions are supported only on one element, employing the transformation $\mathbf{F}_\kappa$, we have

$$\mathbf{M}^\kappa_{i,j} = \int_\kappa \phi_{i,\kappa}(x, y)\phi_{j,\kappa}(x, y)\mathrm{d}\mathbf{x} = \int_{\hat{\kappa}} \hat{\phi}_i(\hat{x}, \hat{y})\hat{\phi}_j(\hat{x}, \hat{y})|\mathbf{J}_\kappa|\mathrm{d}\hat{\mathbf{x}},\ i, j = 1, \ldots, N_{p_\kappa},$$

where in the last integral $\hat{\kappa} = \mathbf{F}_\kappa^{-1}(\kappa) \subset \hat{B}$, see Fig. 10. Here, the Jacobian of the transformation $\mathbf{F}_\kappa$ is given by $|\mathbf{J}_\kappa| = (\mathbf{J}_\kappa)_{1,1}(\mathbf{J}_\kappa)_{2,2}$, which is constant, due to the definition of the map. In order to employ the *homogeneous function* integration method described in the

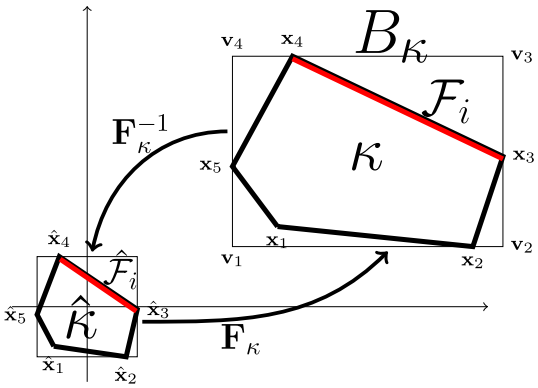

Fig. 10 Example of a polygonal element $\kappa \in \mathcal{T}_h$, the relative bounded box B_κ, the map $\mathbf{F}_\kappa$ and $\hat{\kappa} = \mathbf{F}_\kappa^{-1}(\kappa)$

previous section, we need to identify the coefficients of the homogeneous polynomial expansion for the function $\hat{\phi}_i(\hat{x}, \hat{y})\hat{\phi}_j(\hat{x}, \hat{y})$. We observe that $\hat{\phi}_i(\hat{x}, \hat{y}) = \mathcal{L}_{i_1}(\hat{x})\mathcal{L}_{i_2}(\hat{y})$, and each one–dimensional Legendre polynomial can be expanded as

$$\mathcal{L}_{i_1}(\hat{x}) = \sum_{m=0}^{i_1} C_{i_1,m}\, \hat{x}^m, \qquad \mathcal{L}_{i_2}(\hat{y}) = \sum_{n=0}^{i_2} C_{i_2,n}\, \hat{y}^n. \tag{25}$$

Therefore, we have

$$\begin{aligned}
\mathbf{M}_{i,j}^\kappa &= \int_{\hat{\kappa}} \left(\sum_{m=0}^{i_1} C_{i_1,m}\hat{x}^m\right)\left(\sum_{n=0}^{i_2} C_{i_2,n}\hat{y}^n\right)\left(\sum_{s=0}^{j_1} C_{j_1,s}\hat{x}^s\right)\left(\sum_{r=0}^{j_2} C_{j_2,r}\hat{y}^r\right)|\mathbf{J}_\kappa|\mathrm{d}\hat{\mathbf{x}} \\
&= \int_{\hat{\kappa}} \left(\sum_{k=0}^{i_1+j_1} \mathcal{C}_{i_1,i_2,k}\hat{x}^k\right)\left(\sum_{l=0}^{i_2+j_2} \mathcal{C}_{i_2,j_2,l}\hat{y}^l\right)|\mathbf{J}_\kappa|\mathrm{d}\hat{\mathbf{x}} \\
&= \sum_{k=0}^{i_1+j_1}\sum_{l=0}^{i_2+j_2} \mathcal{C}_{i_1,j_1,k}\,\mathcal{C}_{i_2,j_2,l}\,|\mathbf{J}_\kappa| \int_{\hat{\kappa}} \hat{x}^k\hat{y}^l\mathrm{d}\hat{\mathbf{x}}.
\end{aligned}$$

Here, we have written

$$\mathcal{C}_{i,j,k} = \sum_{n+m=k} \big(C_{i,n}\, C_{j,m}\big), \qquad \text{for } 0 \le i, j \le p_\kappa, \quad 0 \le k \le i + j. \tag{26}$$

Notice that the coefficients $\mathcal{C}_{i,j,k}$ can be evaluated, once and for all, independently of the polygonal element κ. We now consider the general element of the volume matrix $\mathbf{V}_{i,j}$, cf. (24). Proceeding as before, let I, J be the two multi-indices corresponding respectively to i and j, we have

$$\mathbf{V}_{i,j}^\kappa = \int_\kappa \nabla\phi_i \cdot \nabla\phi_j \,\mathrm{d}\mathbf{x} = \underbrace{\int_\kappa \frac{\partial\phi_{I,\kappa}}{\partial x}\frac{\partial\phi_{J,\kappa}}{\partial x}\,\mathrm{d}\mathbf{x}}_{①} + \underbrace{\int_\kappa \frac{\partial\phi_{I,\kappa}}{\partial y}\frac{\partial\phi_{J,\kappa}}{\partial y}\,\mathrm{d}\mathbf{x}}_{②}. \tag{27}$$

Proceeding as before, we apply a change of variables to the terms ① and ② with respect to the map $\mathbf{F}_\kappa$; thereby, we obtain

$$① = \int_{\hat{\kappa}} \frac{\partial \phi_{I,\kappa}}{\partial x}(\mathbf{F}_\kappa(\hat{\mathbf{x}}))\frac{\partial \phi_{J,\kappa}}{\partial x}(\mathbf{F}_\kappa(\hat{\mathbf{x}}))|\mathbf{J}_\kappa|\mathrm{d}\hat{\mathbf{x}},$$
$$② = \int_{\hat{\kappa}} \frac{\partial \phi_{I,\kappa}}{\partial y}(\mathbf{F}_\kappa(\hat{\mathbf{x}}))\frac{\partial \phi_{J,\kappa}}{\partial y}(\mathbf{F}_\kappa(\hat{\mathbf{x}}))|\mathbf{J}_\kappa|\mathrm{d}\hat{\mathbf{x}}.$$

From the definition of $\mathbf{F}_\kappa$, the inverse map is given by $\mathbf{F}_\kappa^{-1}(\mathbf{x}) = \mathbf{J}_\kappa^{-1}(\mathbf{x}-\mathbf{t}_\kappa)$. Then, using the definition (23) of the basis functions, we have the following characterization of the partial derivatives appearing in the terms ① and ②:

$$\frac{\partial}{\partial x}\phi_{I,\kappa}(\mathbf{x}) = \frac{\partial \hat{\phi}_I}{\partial \hat{x}}(\mathbf{F}_\kappa^{-1}(\mathbf{x}))\ (\mathbf{J}_\kappa^{-1})_{1,1}, \quad \frac{\partial}{\partial y}\phi_{I,\kappa}(\mathbf{x}) = \frac{\partial \hat{\phi}_I}{\partial \hat{y}}(\mathbf{F}_\kappa^{-1}(\mathbf{x}))\ (\mathbf{J}_\kappa^{-1})_{2,2}$$

where we have used that $(\mathbf{J}_\kappa^{-1})_{2,1} = (\mathbf{J}_\kappa^{-1})_{1,2} = 0$ since $\mathbf{J}_\kappa$ is diagonal. Then, ① can be written as:

$$① = \int_{\hat{\kappa}} \frac{\partial \hat{\phi}_I}{\partial \hat{x}}(\hat{\mathbf{x}})\frac{\partial \hat{\phi}_J}{\partial \hat{x}}(\hat{\mathbf{x}})\ (\mathbf{J}_\kappa^{-1})_{1,1}^2|\mathbf{J}_\kappa|\mathrm{d}\hat{\mathbf{x}}.$$

Since $(\mathbf{J}_\kappa^{-1})_{1,1}^2|\mathbf{J}_\kappa|$ is constant, the integrand function of term ① is a polynomial. Thereby, we have the following relation:

$$\left.\begin{aligned}\frac{\partial \hat{\phi}_I}{\partial \hat{x}}(\hat{\mathbf{x}}) &= \mathcal{L}'_{i_1}(\hat{x})\ \mathcal{L}_{i_2}(\hat{y}),\\ \frac{\partial \hat{\phi}_J}{\partial \hat{x}}(\hat{\mathbf{x}}) &= \mathcal{L}'_{j_1}(\hat{x})\ \mathcal{L}_{j_2}(\hat{y}),\end{aligned}\right\} \Rightarrow \frac{\partial \hat{\phi}_I}{\partial \hat{x}}(\hat{\mathbf{x}})\frac{\partial \hat{\phi}_J}{\partial \hat{x}}(\hat{\mathbf{x}}) = \mathcal{L}'_{i_1}(\hat{x})\ \mathcal{L}_{i_2}(\hat{y})\ \mathcal{L}'_{j_1}(\hat{x})\ \mathcal{L}_{j_2}(\hat{y}).$$

From the expansion (25) of the Legendre polynomials, we note that

$$\mathcal{L}'_0(\hat{x}) = 0, \quad \mathcal{L}'_i(\hat{x}) = \sum_{m=0}^{i-1}(m+1)C_{i,m+1}\,\hat{x}^m = \sum_{m=0}^{i-1} C'_{i,m}\,\hat{x}^m, \quad \text{for } i > 0; \tag{28}$$

where the indices $C'_{i,m} = (m+1)C_{i,m+1}$ are the coefficients for the expansion of $\mathcal{L}'_i(\cdot)$. We deduce that ① $= 0$ if $i_1 = 0$ or $j_1 = 0$, and

$$① = \sum_{k=0}^{i_1+j_1-2}\ \sum_{l=0}^{i_2+j_2} C'_{i_1,j_1,k}\ \mathcal{C}_{i_2,j_2,l}\ (\mathbf{J}_\kappa^{-1})_{1,1}^2|\mathbf{J}_\kappa|\int_{\hat{\kappa}}\hat{x}^k\hat{y}^l\mathrm{d}\hat{\mathbf{x}}, \quad i_1, j_1 > 0,$$

where $\mathcal{C}_{i_2,j_2,l}$ is defined in (26), and

$$C'_{i,j,k} = \sum_{n+m=k} C'_{i,n}\ C'_{j,m}, \quad 1 \le i, j \le p_\kappa, \text{ for } 0 \le k \le i+j-2,$$

with $C'_{i,n} = (n+1)C_{i,n+1}$, $C'_{j,m} = (m+1)C_{j,m+1}$, cf. (28), is the expansion of the derivatives of the Legendre polynomials which is computable independently of the element κ, $\kappa \in \mathcal{T}_h$. Analogously, we deduce the following expression for the second term of Eq. (27):

$$② = \sum_{k=0}^{i_1+j_1}\ \sum_{l=0}^{i_2+j_2-2} \mathcal{C}_{i_1,j_1,k}\ C'_{i_2,j_2,l}\ (\mathbf{J}_\kappa^{-1})_{2,2}^2|\mathbf{J}_\kappa|\int_{\hat{\kappa}}\hat{x}^k\hat{y}^l\mathrm{d}\hat{\mathbf{x}}.$$

4.3 Interface Integrals over Polytopic Mesh Elements

With regards the interface integrals appearing in Eq. (18), we describe the method by expanding the jump and average operators and computing each term separately, working, for simplicity, again in two-dimensions. Firstly, we discuss how to transform the integral over a physical face $F \subset \partial\kappa$ to the corresponding integral over the face $\hat{F} = \mathbf{F}_\kappa^{-1}(F) \subset \partial\hat{\kappa}$ on the reference rectangular element $\hat{\kappa}$. To this end, let $F \subset \partial\kappa$ be a face of the polygon κ, $\kappa \in \mathcal{T}_h$, and let $\mathbf{x}_1 = (x_1, y_1)$ and $\mathbf{x}_2 = (x_2, y_2)$ denote the vertices of the face, based on counter clock-wise ordering of the polygon vertices. The face $\hat{F} = \mathbf{F}_\kappa^{-1}(F)$ is identified by the two vertices $\hat{\mathbf{x}}_1 = \mathbf{F}_\kappa^{-1}(\mathbf{x})$ and $\hat{\mathbf{x}}_2 = \mathbf{F}_\kappa^{-1}(\mathbf{x}_2)$. For a general integrable function $g : \kappa \to \mathbb{R}$ we have

$$\int_F g(x, y)\mathrm{d}\sigma(x, y) = \int_{\hat{F}} g(\mathbf{F}_\kappa(\hat{x}, \hat{y}))\ \mathrm{d}\sigma(\mathbf{F}_\kappa(\hat{x}, \hat{y})),$$

where $\mathrm{d}\sigma(\mathbf{F}_\kappa(\hat{x}, \hat{y})) = \mathcal{J}_F\ \mathrm{d}\hat{\sigma}$ and $\mathcal{J}_F$ is defined as $\mathcal{J}_F = \|\mathbf{J}_\kappa^{-\top}\hat{\mathbf{n}}_{\hat{F}}\||\mathbf{J}_\kappa|$, where $\hat{\mathbf{n}}_{\hat{F}}$ is the unit outward normal vector to $\hat{F}$.

We next describe how to compute the interface integrals. From the definition of the jump and average operators, cf. (17), on each edge $F \in \mathcal{F}_h^I$ shared by the elements $\kappa^\pm$ we need to assemble

$$\begin{aligned}
\mathbf{S}_{i,j}^{+/+} &= \int_F \alpha_h\ \phi_{i,\kappa^+}\ \phi_{j,\kappa^+}\ \mathrm{d}\sigma, & \mathbf{I}_{i,j}^{+/+} &= \frac{1}{2}\int_F (\nabla\phi_{i,\kappa^+}\cdot\mathbf{n}^+)\ \phi_{j,\kappa^+}\ \mathrm{d}\sigma,\\
\mathbf{S}_{i,j}^{-/-} &= \int_F \alpha_h\ \phi_{i,\kappa^-}\ \phi_{j,\kappa^-}\ \mathrm{d}\sigma, & \mathbf{I}_{i,j}^{-/-} &= \frac{1}{2}\int_F (\nabla\phi_{i,\kappa^-}\cdot\mathbf{n}^-)\ \phi_{j,\kappa^-}\ \mathrm{d}\sigma,\\
\mathbf{S}_{i,j}^{+/-} &= -\int_F \alpha_h\ \phi_{i,\kappa^+}\ \phi_{j,\kappa^-}\ \mathrm{d}\sigma, & \mathbf{I}_{i,j}^{+/-} &= -\frac{1}{2}\int_F (\nabla\phi_{i,\kappa^+}\cdot\mathbf{n}^+)\ \phi_{j,\kappa^-}\ \mathrm{d}\sigma,\\
\mathbf{S}_{i,j}^{-/+} &= -\int_F \alpha_h\ \phi_{i,\kappa^-}\ \phi_{j,\kappa^+}\ \mathrm{d}\sigma, & \mathbf{I}_{i,j}^{-/+} &= -\frac{1}{2}\int_F (\nabla\phi_{i,\kappa^-}\cdot\mathbf{n}^-)\ \phi_{j,\kappa^+}\ \mathrm{d}\sigma,
\end{aligned}$$

for $i, j = 1, \ldots, N_{p_{\kappa^\pm}}$. Analogously, on the boundary face $F \in \mathcal{F}_h^B$ belonging to $\kappa^+ \in \mathcal{T}_h$ we only have to compute

$$\mathbf{S}_{i,j}^{+/+} = \int_F \alpha_h\ \phi_{i,\kappa^+}\ \phi_{j,\kappa^+}\ \mathrm{d}\sigma, \qquad \mathbf{I}_{i,j}^{+/+} = \int_F (\nabla\phi_{i,\kappa^+}\cdot\mathbf{n}^+)\ \phi_{j,\kappa^+}\ \mathrm{d}\sigma,$$

for $i, j = 1, \ldots, N_{p_{\kappa^+}}$. We next show how to efficiently compute a term of the form

$$\mathbf{S}_{i,j}^{+/+} = \int_F \alpha_h\ \phi_{I,\kappa^+}(x, y)\ \phi_{J,\kappa^+}(x, y)\mathrm{d}\sigma,$$

where I, J are the suitable multi-indices associated to $i, j = 1, \ldots, N_{p_{\kappa^+}}$, respectively. Proceeding as before, we have

$$\begin{aligned}
\mathbf{S}_{i,j}^{+/+} &= \int_F \alpha_h\phi_{I,\kappa^+}(x, y)\phi_{J,\kappa^+}(x, y)\mathrm{d}\sigma(x, y) = \int_{\hat{F}} \alpha_h\hat{\phi}_I(\hat{x}, \hat{y})\hat{\phi}_J(\hat{x}, \hat{y})\mathcal{J}_F\mathrm{d}\hat{\sigma}\\
&= \sum_{k=0}^{i_1+j_1}\sum_{l=0}^{i_2+j_2} \alpha_h\ \mathcal{C}_{i_1,j_1,k}\ \mathcal{C}_{i_2,j_2,l}\ \mathcal{J}_F \int_{\hat{F}} \hat{x}^k\hat{y}^l\mathrm{d}\hat{\sigma}.
\end{aligned}$$

Analogously, we have

$$\begin{aligned}\mathbf{S}_{i,j}^{+/-} &= -\int_F \alpha_h\, \phi_{I,\kappa^+}(x,y)\phi_{J,\kappa^-}(x,y)\mathrm{d}\sigma(x,y)\\ &= -\int_{\mathbf{F}_{\kappa^+}^{-1}(F)} \alpha_h \underbrace{\phi_{I,\kappa^+}(\mathbf{F}_{\kappa^+}(\hat{\mathbf{x}}))}_{\text{ⓐ}} \underbrace{\phi_{J,\kappa^-}(\mathbf{F}_{\kappa^+}(\hat{\mathbf{x}}))}_{\text{ⓑ}} \mathcal{J}_{F^+}\mathrm{d}\sigma. \end{aligned} \tag{29}$$

For the term ⓐ, we directly apply the definition of the basis function, and obtain

$$\text{ⓐ} = \phi_{I,\kappa^+}(\mathbf{F}_{\kappa^+}(\hat{\mathbf{x}})) = \hat{\phi}_I(\mathbf{F}_{\kappa^+}^{-1}(\mathbf{F}_{\kappa^+}(\hat{\mathbf{x}}))) = \hat{\phi}_I(\hat{\mathbf{x}}) = \sum_{k=0}^{i_1}\sum_{l=0}^{i_2} C_{i_1,k}\, C_{i_2,l}\, \hat{x}^k \hat{y}^l, \tag{30}$$

while for the term ⓑ we have

$$\text{ⓑ} = \phi_{J,\kappa^-}(\mathbf{F}_{\kappa^+}(\hat{\mathbf{x}})) = \hat{\phi}_J(\mathbf{F}_{\kappa^-}^{-1}(\mathbf{F}_{\kappa^+}(\hat{\mathbf{x}}))).$$

In order to obtain a homogeneous polynomial expansion for ⓑ we have to write explicitly the composite map $\tilde{\mathbf{F}}(\hat{\mathbf{x}}) = \mathbf{F}_{\kappa^-}^{-1}(\mathbf{F}_{\kappa^+}(\hat{\mathbf{x}}))$. That is

$$\tilde{\mathbf{F}}(\hat{\mathbf{x}}) = \mathbf{J}_{\kappa^-}^{-1}(\mathbf{J}_{\kappa^+}\hat{\mathbf{x}} + \mathbf{t}_{\kappa^+}) - \mathbf{J}_{\kappa^-}^{-1}\mathbf{t}_{\kappa^-} = \underbrace{\mathbf{J}_{\kappa^-}^{-1}\mathbf{J}_{\kappa^+}}_{\tilde{\mathbf{J}}}\hat{\mathbf{x}} + \underbrace{\mathbf{J}_{\kappa^-}^{-1}(\mathbf{t}_{\kappa^+} - \mathbf{t}_{\kappa^-})}_{\tilde{\mathbf{t}}},$$

where the matrix $\tilde{\mathbf{J}}$ is diagonal since $\mathbf{J}_{\kappa^-}^{-1}$ and $\mathbf{J}_{\kappa^+}$ are diagonal. We then have

$$\begin{aligned}\text{ⓑ} &= \hat{\phi}_J(\hat{\mathbf{F}}(\hat{\mathbf{x}})) = \hat{\phi}_J(\tilde{\mathbf{J}}\hat{\mathbf{x}} + \tilde{\mathbf{t}}) = \hat{\phi}_J(\tilde{\mathbf{J}}_{1,1}\hat{x} + \tilde{\mathbf{t}}_1, \tilde{\mathbf{J}}_{2,2}\hat{y} + \tilde{\mathbf{t}}_2)\\ &= \sum_{k=0}^{j_1}\sum_{l=0}^{j_2} C_{j_1,k}\, C_{j_2,l}\, (\tilde{\mathbf{J}}_{1,1}\hat{x} + \tilde{\mathbf{t}}_1)^k (\tilde{\mathbf{J}}_{2,2}\hat{y} + \tilde{\mathbf{t}}_2)^l. \end{aligned} \tag{31}$$

Combining (30) and (31), and denoting by $\hat{F}^+ = \mathbf{F}_{\kappa^+}^{-1}(F)$, cf. Fig. 11, from (29) we obtain

$$\mathbf{S}_{i,j}^{+/-} = -\sum_{k=0}^{i_1+j_1}\sum_{l=0}^{i_2+j_2} \tilde{\mathcal{X}}_{i_1,j_1,k}\, \tilde{\mathcal{Y}}_{i_2,j_2,l}\, \mathcal{J}_{F^+} \int_{\hat{F}^+} \hat{x}^k \hat{y}^l \mathrm{d}\hat{\sigma},$$

where $\tilde{\mathcal{X}}$ and $\tilde{\mathcal{Y}}$ are defined as

$$\left.\begin{aligned}\tilde{\mathcal{X}}_{i,j,k} &= \sum_{n+m=k} \left(C_{i,n}\, \tilde{X}_{j,m}\right)\\ \tilde{\mathcal{Y}}_{i,j,k} &= \sum_{n+m=k} \left(C_{i,n}\, \tilde{Y}_{j,m}\right)\end{aligned}\right\} \quad \text{for } 0 \le i \le p_{\kappa^+},\ 0 \le j \le p_{\kappa^-},\ 0 \le k \le i + j.$$

Here, as before, $C_{i,n}$ are the coefficients of the homogeneous function expansion of the Legendre polynomials in $(-1, 1)$, while $\tilde{X}_{j,m}$ and $\tilde{Y}_{j,m}$ are defined by

$$\left.\begin{aligned}\tilde{X}_{j,m} &= \sum_{r=m}^{j} C_{j,r} \binom{r}{m} (\tilde{\mathbf{J}}_{1,1})^m\, (\tilde{\mathbf{t}}_1)^{r-m}\\ \tilde{Y}_{j,m} &= \sum_{r=m}^{j} C_{j,r} \binom{r}{m} (\tilde{\mathbf{J}}_{2,2})^m\, (\tilde{\mathbf{t}}_2)^{r-m}\end{aligned}\right\} \quad \text{for } 0 \le m \le p_{\kappa^-},\ m \le j \le p_{\kappa^-};$$

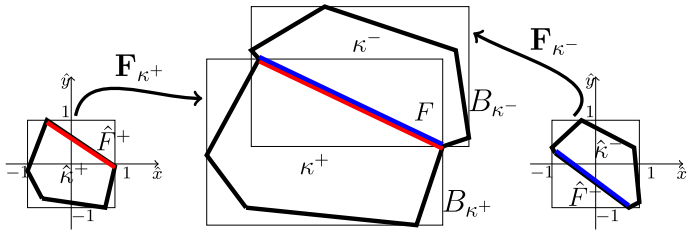

Fig. 11 Example of a polygonal elements $\kappa^{\pm} \in \mathcal{T}_h$, together with the bounded boxes $B_{\kappa^{\pm}}$, and the local maps $\mathbf{F}_{\kappa^{\pm}} : \hat{\kappa} \to \kappa^{\pm}$ for the common face $F \subset \kappa^{\pm}$

here, we have exploited the Newton-binomial expansion of the terms $(\tilde{\mathbf{J}}_{1,1}\hat{x} + \tilde{\mathbf{t}}_1)^k$ and $(\tilde{\mathbf{J}}_{2,2}\hat{y} + \tilde{\mathbf{t}}_2)^l$ appearing in equation (31).
Similar considerations allow us to compute

$$\begin{aligned}
\mathbf{I}^{+/+}_{i,j} &= \frac{1}{2}\left(\int_F \frac{\partial \phi_{I,\kappa^+}}{\partial x}(x,y)\phi_{J,\kappa^+}(x,y)n_x^+ + \int_F \frac{\partial \phi_{I,\kappa^+}}{\partial y}(x,y)\phi_{J,\kappa^+}(x,y)n_y^+\right)\\
&= \frac{1}{2}\left(\int_{\hat{F}} (\mathbf{J}^{-1}_{\kappa^+})_{1,1}\frac{\partial \hat{\phi}_I}{\partial \hat{x}}\hat{\phi}_J n_x^+ \mathcal{J}_F \mathrm{d}\hat{\sigma} + \int_{\hat{F}} (\mathbf{J}^{-1}_{\kappa^+})_{2,2}\frac{\partial \hat{\phi}_I}{\partial \hat{y}}\hat{\phi}_J n_y^+ \mathcal{J}_F \mathrm{d}\hat{\sigma}\right)\\
&= \frac{1}{2}\mathcal{J}_F\left((\mathbf{J}^{-1}_{\kappa^+})_{1,1} n_x^+ \sum_{k=0}^{i_1+j_1-1}\sum_{l=0}^{i_2+j_2} \mathcal{C}''_{i_1,j_1,k}\mathcal{C}_{i_2,j_2,l}\int_{\hat{F}} \hat{x}^k\hat{y}^l \mathrm{d}\hat{\sigma}\right.\\
&\quad \left. + (\mathbf{J}^{-1}_{\kappa^+})_{2,2} n_y^+ \sum_{k=0}^{i_1+j_1}\sum_{l=0}^{i_2+j_2-1} \mathcal{C}_{i_1,j_1,k}\,\mathcal{C}''_{i_2,j_2,l}\int_{\hat{F}} \hat{x}^k\hat{y}^l \mathrm{d}\hat{\sigma}\right),
\end{aligned}$$

where $\mathcal{C}''_{i,j,k}$ are defined as

$$\begin{cases} \mathcal{C}''_{0,j,k} = 0 & \forall j,\ \forall k,\\ \mathcal{C}''_{i,j,k} = \sum_{n+m=k} \mathcal{C}'_{i,n}\mathcal{C}_{j,m}, & 1 \le i \le p_{\kappa^+}, 0 \le j \le p_{\kappa^+}, 0 \le k \le i+j-1, \end{cases}$$

and where $\mathbf{n}^+ = [n_x^+, n_y^+]^\top$ is the unit outward normal vector to the physical face F from κ^+. Similarly,

$$\begin{aligned}
\mathbf{I}^{+/-}_{i,j} &= -\frac{1}{2}\int_F (\nabla \phi_{I,\kappa^+}\cdot \mathbf{n}^+)\phi_{J,\kappa^-}\mathrm{d}\sigma\\
&= -\frac{1}{2}\left(\int_F \frac{\partial \phi_{I,\kappa^+}}{\partial x}\phi_{J,\kappa^-}n_x^+ \mathrm{d}\sigma + \int_F \frac{\partial \phi_{I,\kappa^+}}{\partial y}\phi_{J,\kappa^-}n_y^+ \mathrm{d}\sigma\right)\\
&= -\frac{1}{2}\mathcal{J}_F\left((\mathbf{J}^{-1}_{\kappa})_{1,1}\sum_{k=0}^{i_1+j_1-1}\sum_{l=0}^{i_2+j_2} \tilde{\mathcal{X}}'_{i_1,j_1,k}\tilde{\mathcal{Y}}_{i_2,j_2,l}\int_{\hat{F}^+} \hat{x}^k\hat{y}^l \mathrm{d}\hat{\sigma}\right.\\
&\quad \left. + (\mathbf{J}^{-1}_{\kappa})_{2,2}\sum_{k=0}^{i_1+j_1}\sum_{l=0}^{i_2+j_2-1} \tilde{\mathcal{X}}_{i_1,j_1,k}\,\tilde{\mathcal{Y}}'_{i_2,j_2,l}\int_{\hat{F}^+} \hat{x}^k\hat{y}^l \mathrm{d}\hat{\sigma}\right),
\end{aligned}$$

where we have also introduced $\tilde{\mathcal{X}}'$ and $\tilde{\mathcal{Y}}'$ defined as

$$\left.\begin{aligned}\tilde{\mathcal{X}}'_{i,j,k} &= \sum_{n+m=k} \left(C'_{i,n}\,\tilde{X}_{j,m}\right),\\ \tilde{\mathcal{Y}}'_{i,j,k} &= \sum_{n+m=k} \left(C'_{i,n}\,\tilde{Y}_{j,m}\right),\end{aligned}\right\} \text{ for } 1 \le i \le p_{\kappa^+},\ 0 \le j \le p_{\kappa^-},\ 0 \le k \le i+j-1.$$

Remark 5 The coefficients $\tilde{X}$ and $\tilde{Y}$ depend on the maps $\mathbf{F}_{\kappa^+}$ and $\mathbf{F}_{\kappa^-}$, as well as $\tilde{\mathcal{X}}$, $\tilde{\mathcal{X}}'$, $\tilde{\mathcal{Y}}$ and $\tilde{\mathcal{Y}}'$; thereby, they must be computed for each element κ in the mesh $\mathcal{T}_h$.

Remark 6 With regards the computation of the forcing term

$$\mathbf{f}_i = \int_\Omega f(\mathbf{x})\phi_i(\mathbf{x})\mathrm{d}\mathbf{x},\ \forall i = 1,\dots,N_h, \tag{32}$$

we point out that the quadrature method proposed in this paper allows to exactly evaluate (32) when f is a constant or a polynomial function. If f is a general function, an explicit polynomial approximation of f is required.

5 Numerical Experiments

We present some two- and three-dimensional numerical experiments to test the practical performance of the proposed approach. Here, the results are compared with standard assembly algorithms based on employing efficient quadrature rules on a sub-tessellation.

5.1 Two-Dimensional Test Case

We test the performance of the algorithm outlined in Sect. 4 for the computation of the elemental mass and stiffness matrices resulting from the DG discretization (19) on Voronoi decompositions as shown in Fig. 12. In particular, we compare the CPU-time needed to assemble the local and global elemental matrices using Algorithm 2, cf. Sect. 4, with *Quadrature Integration* over polygonal domains, based on the sub-tessellation method on polygons and Gaussian line integration for the related interface terms. More precisely, given $\kappa \in \mathcal{T}_h$, the sub-tessellation scheme on κ is performed by constructing a non-overlapping sub-tessellation $\kappa_{\mathscr{S}} = \{\tau_\kappa\}$ consisting of standard triangular elements; in particular, as, for our tests, we consider Voronoi numerical grids, we exploit the convexity of κ and define $\kappa_{\mathscr{S}}$ by connecting the centre of mass of κ with its vertices. As an example, if we consider computing the elemental mass matrix $\mathbf{M}^\kappa_{i,j}$, we have that

$$\mathbf{M}^\kappa_{i,j} = \int_\kappa \phi_i\phi_j\mathrm{d}\mathbf{x} \approx \sum_{\tau_\kappa\in\kappa_{\mathscr{S}}}\sum_{r=1}^{q_{\tau_\kappa}} \phi_i(\mathbf{F}_{\tau_\kappa}(\boldsymbol{\xi}_r))\phi_j(\mathbf{F}_{\tau_\kappa}(\boldsymbol{\xi}_r))|\mathbf{J}_{\tau_\kappa}|\omega_r,$$

where $\mathbf{F}_{\tau_\kappa} : \hat{\tau} \to \tau_\kappa$ is the mapping from the reference simplex $\hat{\tau}$ to τ_κ, with Jacobian $|\mathbf{J}_{\tau_\kappa}|$, and $\{(\boldsymbol{\xi}_r, \omega_r)\}_{r=1}^{q_\kappa}$ denotes the quadrature rule defined on $\hat{\tau}$. The construction of quadrature rules on $\hat{\tau}$ may be computed based on employing the Duffy transformation, whereby the reference tensor-product element $(-1,1)^2$ is mapped to the reference simplex. As the algorithm outlined in Sect. 4 does not require the definition of quadrature nodes and weights, in the following we will refer to it as the *Quadrature Free Method*. Consider the problem (19) introduced in Sect. 3 with $d = 2$ and $\Omega = (0,1)^2$, where we select the set of basis functions

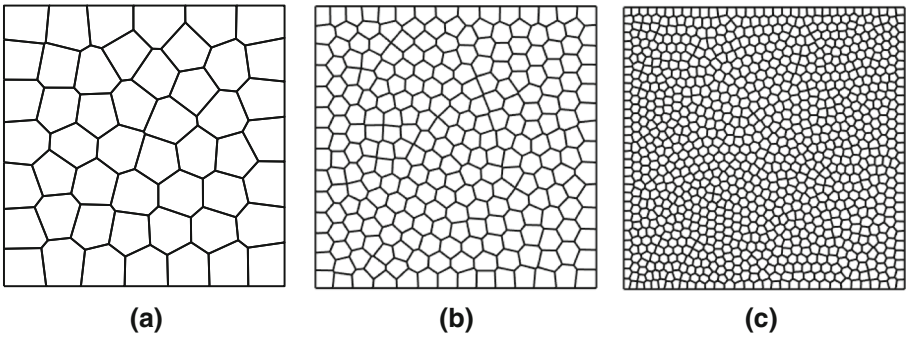

Fig. 12 Example of Voronoi mesh on $\Omega = (0, 1)^2$. **a** 50 elements. **b** 250 elements. **c** 1000 elements

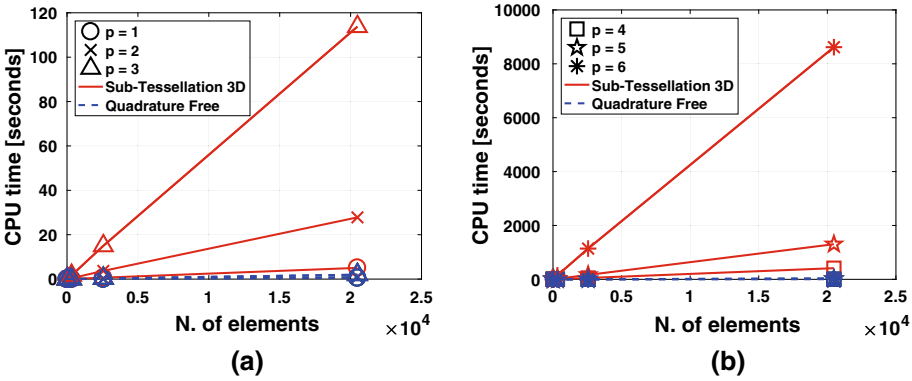

Fig. 13 Comparison of the CPU time needed to assemble the global matrices $\mathbf{M}$ and $\mathbf{V}$ for a two-dimensional problem by using the proposed quadrature free method and the classical sub-tessellation scheme. For each algorithm, each line is obtained by fixing the polynomial approximation degree $p \in \{1, 2, 3\}$ (left) and $p \in \{4, 5, 6\}$ (right), and measuring the CPU time by varying the number of elements in the underlying mesh. **a** Comparison for $p \in \{1, 2, 3\}$. **b** Comparison for $p \in \{4, 5, 6\}$

$\{\phi_i\}_{i=1}^{N_h}$ for V_h as described in Sect. 4. In order to quantify the performance of the proposed approach, we consider a series of numerical tests obtained by varying the polynomial degree $p_\kappa = p$ for all $\kappa \in \mathcal{T}_h$, between 1 and 6 and by employing a series of uniform polygonal meshes of different granularity, cf. Fig. 12. The numerical grids are constructed based on employing `PolyMesher`, cf. [65]. Here, we are interested in the CPU time needed to assemble the matrices (21) and (22).

In the first test case, we consider the CPU time needed to assemble the matrices $\mathbf{M}$ and $\mathbf{V}$. As pointed out in Sect. 4, these matrices are block diagonal and each block consists of an integral over each polygonal element $\kappa \in \mathcal{T}_h$. In Fig. 13 we present the comparison between the CPU times needed to assemble the global matrices $\mathbf{M}$ and $\mathbf{V}$ based on employing the quadrature free method and quadrature integration (based on sub-tessellation) when varying the number of elements $N_e \in \{64, 256, 1024, 4094, 16384, 65536\}$ and the polynomial degree $p \in \{1, 2, 3\}$ (left), and $p \in \{4, 5, 6\}$ (right). Clearly, our approach outperforms the classical sub-tessellation method leading to substantial gains in efficiency. For a more detailed comparison, we have presented in Fig. 15a the logarithmic-scaled graphs of each computation: from the results of Fig. 15a we observe that the CPU time grows like $\mathcal{O}(N_e)$, as N_e increases, as expected.

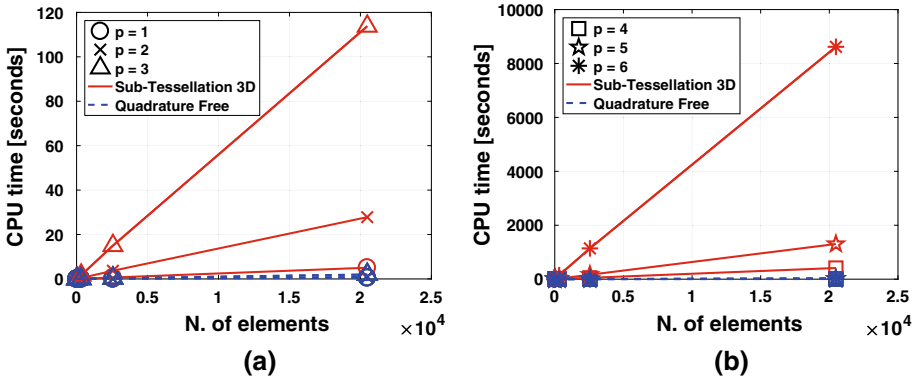

Fig. 14 Comparison of the CPU time needed to assemble the global matrices **S** and **I** for a two-dimensional problem by using the proposed quadrature free method and the classical Gauss line integration scheme. For each approach, each line is obtained by fixing the polynomial approximation degree $p \in \{1, 2, 3\}$ (left) and $p \in \{4, 5, 6\}$ (right), and measuring the CPU time by varying the number of elements in the underlying mesh. **a** Comparison for $p \in \{1, 2, 3\}$. **b** Comparison for $p \in \{4, 5, 6\}$

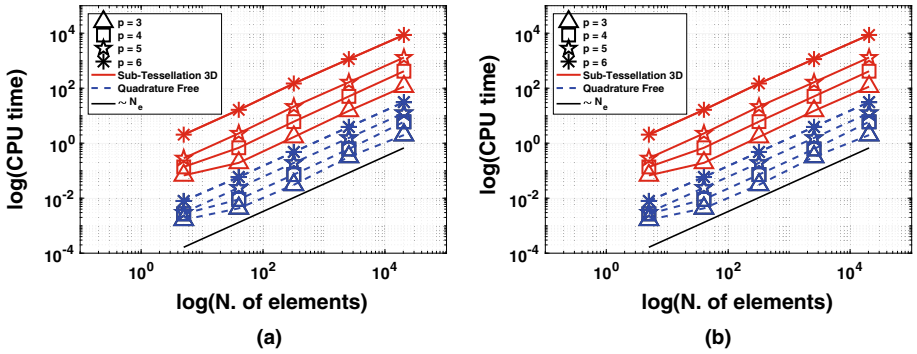

Fig. 15 Comparison between the CPU time needed by the two method to assemble the global matrices **M** and **V** (left) and **S** and **I** (right) for a three-dimensional problem, versus the number of elements and for different choices of $p = 3, \ldots, 6$ (log–log scale). **a** CPU time comparison needed to assemble **M** and **V** in log–log scale. **b** CPU time comparison needed to assemble **S** and **I** in log-log scale

We have repeated the same set of numerical experiments measuring the CPU times needed to assemble the face terms appearing in the matrices **S** and **I**; these results are reported in Fig. 14. Here, the domains of integration of the integrals involved are the edges of the polygonal elements, which are simply line segments in the plane $\mathbf{R}^2$. Here, we compare the quadrature free method described in Sect. 4.3 with classical Gaussian line integration, where the integrating function is pointwise evaluated on the physical numerical nodes lying on each face. The graphs in Fig. 14a, b show the comparison between the CPU time taken for the two different approaches. Here, we again observe that significant computational savings are made when the proposed quadrature free method is employed, though the increase in efficiency is less than that attained for the computation of the volume integrals. In Fig. 15b we plot the logarithmic-scaled CPU time with respect to the number of mesh elements; again the CPU time grows as $\mathcal{O}(N_e)$ as N_e increases.

Referring to Figs. 13 and 14, we observe that the cost of assembly of the matrices **M** and **V**, which involve volume integrals over each element κ in the computational mesh $\mathcal{T}_h$,

is more expensive than the time it takes to assemble the face-based matrices **S** and **I**, when the classical Gaussian line integration method is employed. This is, of course, due to the greater number of function evaluations required to compute **M** and **V** on the underlying sub-tessellation; note that in two-dimensions, a sub-tesellation of the faces is not necessary, since they simply consist of line segments. However, the opposite behaviour is observed when the quadrature free method is employed; in this case, the volume integrals can be very efficiently computed since the coefficients $\mathcal{C}_{i,j,k}$ and $\mathcal{C}'_{i,j,k}$ only need to be computed once, cf. Sect. 4.2. On the other hand, computing the face integrals present in **S** and **I** requires the evaluation of the coefficients $\tilde{X}_{j,m}$, $\tilde{\mathcal{X}}_{i,j,k}$, $\tilde{\mathcal{X}}'_{i,j,k}$, $\tilde{Y}_{j,m}$, $\tilde{\mathcal{Y}}_{i,j,k}$, and $\tilde{\mathcal{Y}}'_{i,j,k}$, cf. Sect. 4.3, which must be computed for each face $F \in \mathcal{F}_h$.

5.2 Three-Dimensional Test Case

We now consider the diffusion–reaction problem (19) with $d = 3$ and $\Omega = (0,1)^3$. The polyhedral grids employed for this test case are defined by agglomeration: starting from a fine partition $\mathcal{T}_{fine}$ of Ω consisting of N_{fine} disjoint tetrahedrons $\{\kappa_f^i\}_{i=1}^{N_{fine}}$, such that $\overline{\Omega} = \cup_{i=1}^{N_{fine}} \overline{\kappa}_f^i$, a coarse mesh $\mathcal{T}_h$ of Ω consisting of disjoint polyhedral elements κ can be defined such that

$$\kappa = \cup_{\kappa'_f \in \mathcal{S}_\kappa} \kappa'_f \quad \forall \kappa \in \mathcal{T}_h, \tag{33}$$

where $\mathcal{S}_\kappa \subset \mathcal{T}_h^{fine}$ denotes the set of fine elements which forms κ. Here, the agglomeration of fine tetrahedral elements is performed based on employing the *METIS* library for graph partitioning, cf., for example, [45,46]. With this definition each polyhedral element is typically non-convex. For simplicity, we have considered only the case of simply connected elements. In this particular case, the faces of the mesh $\mathcal{T}_h$ are the triangular intersections of two-dimensional facets of neighbouring elements. Figure 16 shows three examples of the polyhedral elements resulting from agglomeration.

We perform a similar set of experiments as the ones outlined in Sect. 5.2 for the two-dimensional case. Again, we compare the CPU time required by the proposed quadrature free method with the quadrature integration/sub-tessellation approach to assemble the stiff-

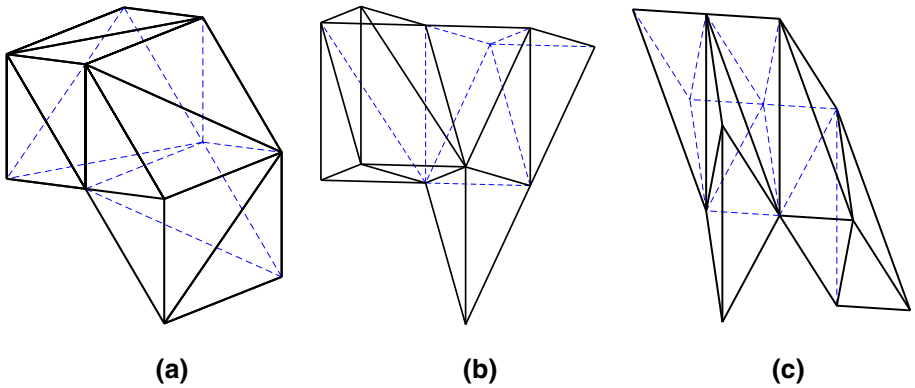

Fig. 16 Example of polyhedral elements $\kappa \in \mathcal{T}_h$ obtained by agglomeration of tetrahedrons. κ_1 has 18 triangular *faces*, κ_2 has 20 triangular *faces* and κ_3 has 22 triangular *faces*. **a** Element κ_1. **b** Element κ_2. **c** Element κ_3

ness and mass matrices resulting from the DG discretization of problem (19). Numerical integration over a polyhedral domain is required to assemble the matrices **M** and **V**, cf. (21), whereas for the computation of **S** and **I**, cf. (22), a cubature rule over polygonal faces (here triangular shaped) is needed. In general, for three-dimensional problems the quadrature integration approach consists in the application of the sub-tessellation method both for volume and face integrals, although in this particular case no sub-tessellation is required for the face integrals, since they simply consist of triangular domains. Moreover, as in this case a sub-tessellation into tetrahedral domains is already given by the definition of the polyhedral mesh, the quadrature integration for volume integrals on a general agglomerated polyhedral element $\kappa = \cup_{\kappa'_f \in \mathscr{S}_\kappa} \kappa'_f$ is realized by applying an exact quadrature rule on each tetrahedron $\kappa'_f \in \mathscr{S}_\kappa$. The comparison of the CPU times for the two methods outlined here are

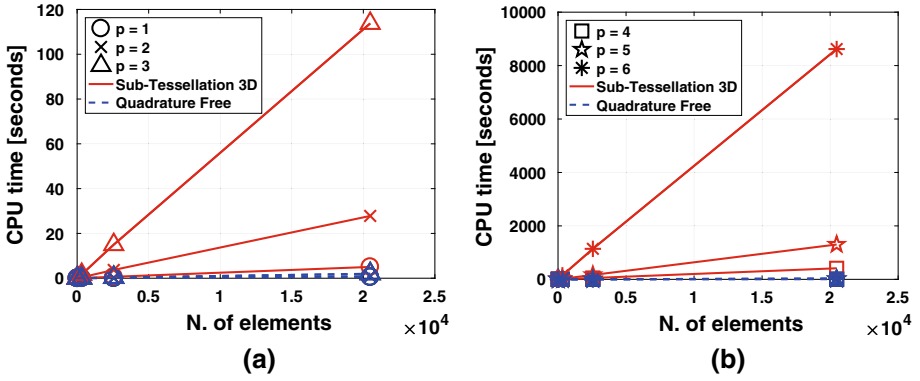

Fig. 17 Comparison of the CPU time needed to assemble the global matrices **M** and **V** for a three-dimensional problem by using the proposed quadrature free method and the classical sub-tessellation method. For each approach, each line is obtained by fixing the polynomial approximation degree $p \in \{1, 2, 3\}$ (left) and $p \in \{4, 5, 6\}$ (right), and measuring the CPU time by varying the number of elements of the underlying mesh. **a** Comparison for $p \in \{1, 2, 3\}$. **b** Comparison for $p \in \{4, 5, 6\}$

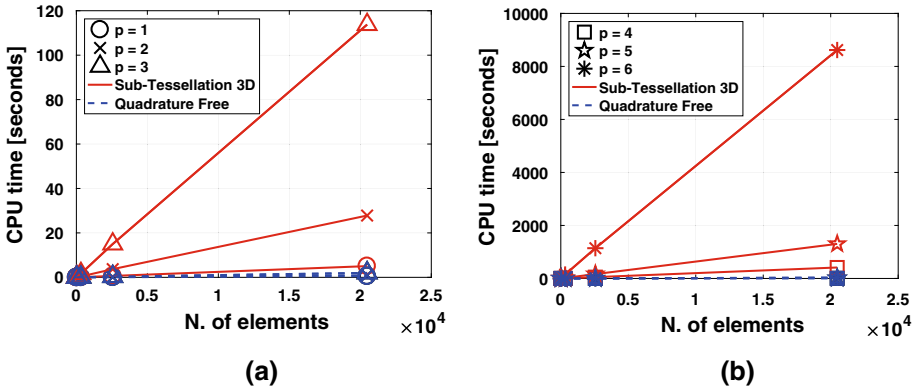

Fig. 18 Comparison of the CPU time needed to assemble the global matrices **S** and **I** for a three–dimensional problem by using the proposed quadrature free method and the classical sub-tessellation method. For each approach, each line is obtained by fixing the polynomial approximation degree $p \in \{1, 2, 3\}$ (left) and $p \in \{4, 5, 6\}$ (right), and measuring the CPU time by varying the number of elements of the underlying mesh. **a** Comparison for $p \in \{1, 2, 3\}$. **b** Comparison for $p \in \{4, 5, 6\}$

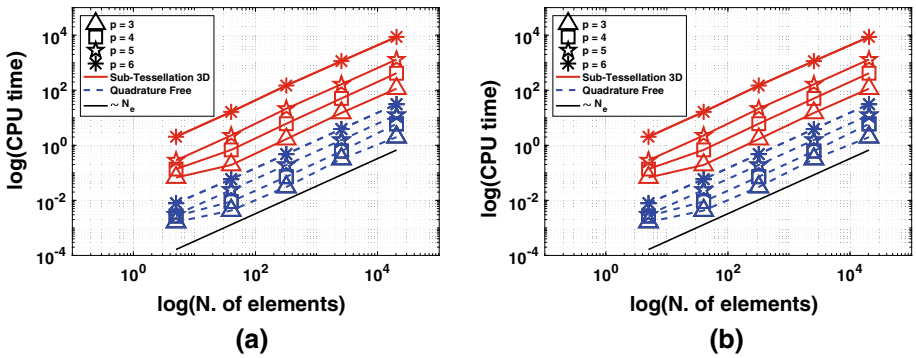

Fig. 19 Comparison between the CPU time needed by the two method to assemble the global matrices **M** and **V** (left) and **S** and **I** (right) for a three-dimensional problem, versus the number of elements and for different choices of $p = 3, \ldots, 6$ (log–log scale). **a** CPU time comparison needed to assemble **M** and **V** in log–log scale. **b** CPU time comparison needed to assemble **S** and **I** in log–log scale

presented for a set of agglomerated polyhedral grids where we vary the number of elements $N_e \in \{5, 40, 320, 2560, 20480\}$, and the polynomial degree $p \in \{1, 2, 3, 4, 5, 6\}$. For each agglomerated polyhedral grid $\mathcal{T}_h$ we have chosen the corresponding fine tetrahedral grid $\mathcal{T}_{fine}$ such that the cardinality of the set $\mathscr{S}_\kappa$ appearing in (33) is $|\mathscr{S}_\kappa| \sim 10 \ \forall \kappa \in \mathcal{T}_h$. The results are shown in Fig. 17 for the computation of the matrices **M** and **V**, and in Fig. 18 for the computation of matrices **S** and **I**. Here, we observe analogous behaviour to the two-dimensional case: the quadrature free method substantially outperforms quadrature integration both for the computation of the volume and face integrals. We also have reported in Fig. 19 the logarithmic-scaled graphs of each computation, showing that, as expected, the gain in terms of CPU time attained by exploiting the proposed method is more evident here, with respect to the two-dimensional case, also for the face integrals.

6 Conclusions

In this article we have proposed a new approach for the numerical evaluation of the integrals required to assemble the mass and stiffness matrices arising from the DG discretization of diffusion–reaction problems, where the underlying mesh is composed of polygonal/polyhedral elements. Starting from the idea proposed in [27] for the integration of homogeneous functions, we have developed a cubature method which does not require the definition of a set of nodes and weights on the domain of integration, and allows for the exact integration of polynomial functions based on evaluating the integrand and its derivatives only at the vertices of the polytopal integration domain. This approach shows a remarkable gain in terms of CPU time with respect to classical quadrature rules, maintaining the same degree of accuracy. On the one hand, the number of computations is optimized, with respect to the polynomial degree of the integrand, and moreover less memory storage is required as no sub-tessellation and quadrature nodes and weights are required. With regards the three-dimensional tests presented in Sect. 5.2, we note that more substantial gains in terms of CPU time, with respect to classical approaches, can be obtained if the underlying grid is composed of pure (not agglomerated) polyhedral elements: firstly, this is because a sub-partition should be defined on the fly for each element, and secondly, as faces are not only triangles but pos-

sibly polygons of arbitrary shape, a sub-tessellation is needed also for surface integrals. The proposed technique is completely general and can be extended to several numerical methods based on discrete spaces defined on polygonal/polyhedral meshes, such as Virtual Element Methods, Mimetic Finite Differences, Hybrid High-Order Methods, Hybridizable Discontinuous Galerkin Schemes, and Polygonal Finite Element Methods, for example. We stress that for moderate polynomial degrees, the proposed integration technique, which involves exact integration of bivariate and trivariate functions in two- and three-dimensions, respectively, has been observed to be numerically stable.

References

1. Antonietti, P.F., Brezzi, F., Marini, L.D.: Bubble stabilization of discontinuous Galerkin methods. Comput. Methods Appl. Mech. Eng. **198**(21–26), 1651–1659 (2009)
2. Antonietti, P.F., Cangiani, A., Collis, J., Dong, Z., Georgoulis, E.H., Giani, S., Houston, P.: Review of discontinuous Galerkin Finite Element Methods for partial differential equations on complicated domains, *Lecture Notes in Computational Science and Engineering*, vol. 114, 1st edn., chap. 8, pp. 281–310. Springer (2016)
3. Antonietti, P.F., Facciolà, C., Russo, A., Verani, M.: Discontinuous Galerkin approximation of flows in fractured porous media. Technical Report 22/2016, MOX Report (2016)
4. Antonietti, P.F., Formaggia, L., Scotti, A., Verani, M., Verzott, N.: Mimetic Finite Difference approximation of flows in fractured porous media. ESAIM Math. Model. Numer. Anal. **50**(3), 809–832 (2016)
5. Antonietti, P.F., Giani, S., Houston, P.: *hp*-version Composite Discontinuous Galerkin methods for elliptic problems on complicated domains. SIAM J. Sci. Comput. **35**(3), A1417–A1439 (2013)
6. Antonietti, P.F., Giani, S., Houston, P.: Domain decomposition preconditioners for discontinuous Galerkin methods for elliptic problems on complicated domains. J. Sci. Comput. **60**(1), 203–227 (2014)
7. Antonietti, P.F., Houston, P., Pennesi, G.: Fast numerical integration on polytopic meshes with applications to discontinuous Galerkin finite element methods. MOX Report No. 03/2018 (2018)
8. Antonietti, P.F., Manzini, G., Verani, M.: The fully nonconforming Virtual Element Method for biharmonic problems. Math. Mod. Methods Appl. Sci. (Accepted: 16 October 2017)
9. Antonietti, P.F., Pennesi, G.: V-cycle multigrid algorithms for discontinuous Galerkin methods on non-nested polytopic meshes. MOX Report No. 49/2017 (submitted 2017)
10. Antonietti, P.F., da Veiga, L.B., Mora, D., Verani, M.: A stream Virtual Element formulation of the Stokes problem on polygonal meshes. SIAM J. Numer. Anal. **52**(1), 386–404 (2014)
11. Antonietti, P.F., da Veiga, L.B., Scacchi, S., Verani, M.: A C1 Virtual Element Method for the Cahn–Hilliard equation with polygonal meshes. SIAM J. Numer. Anal. **54**(1), 34–56 (2016)
12. Arnold, N.D., Brezzi, F., Cockburn, B., Marini, L.D.: Unified analysis of discontinuous Galerkin methods for elliptic problems. SIAM J. Numer. Anal. **39**(5), 1749–1779 (2001/2002)
13. Ayuso de Dios, B., Lipnikov, K., Manzini, G.: The nonconforming Virtual Element Method. ESAIM Math. Model. Numer. Anal. **50**(3), 879–904 (2016)
14. Bassi, F., Botti, L., Colombo, A.: Agglomeration-based physical frame dG discretizations: an attempt to be mesh free. Math. Mod. Methods Appl. Sci. **24**(8), 1495–1539 (2014)
15. Beirão da Veiga, L., Brezzi, F., Cangiani, A., Manzini, G., Marini, L.D., Russo, A.: Basic principles of Virtual Element Methods. Math. Models Methods Appl. Sci. **23**(1), 199–214 (2013)
16. Beirão da Veiga, L., Brezzi, F., Marini, L.D., Russo, A.: Mixed Virtual Element Methods for general second order elliptic problems on polygonal meshes. ESAIM Math. Model. Numer. Anal. **50**(3), 727–747 (2016)
17. Beirão da Veiga, L., Brezzi, F., Marini, L.D., Russo, A.: Virtual Element Method for general second-order elliptic problems on polygonal meshes. Math. Models Methods Appl. Sci. **26**(4), 729–750 (2016)
18. Beirão da Veiga, L., Lipnikov, K., Manzini, G.: The Mimetic Finite Difference Method for Elliptic Problems, vol. 11. Springer, Cham (2014)

19. Brezzi, F., Lipnikov, K., Shashkov, M.: Convergence of the Mimetic Finite Difference method for diffusion problems on polyhedral meshes. SIAM J. Numer. Anal. **43**(5), 1872–1896 (electronic) (2005)
20. Brezzi, F., Lipnikov, K., Shashkov, M.: Convergence of Mimetic Finite Difference method for diffusion problems on polyhedral meshes with curved faces. Math. Mod. Methods Appl. Sci. **16**(2), 275–297 (2006)
21. Brezzi, F., Lipnikov, K., Simoncini, V.: A family of Mimetic Finite Difference methods on polygonal and polyhedral meshes. Math. Mod. Methods Appl. S. **15**(10), 1533–1551 (2005)
22. Cangiani, A., Dong, Z., Georgoulis, E.H.: *hp*-Version space–time discontinuous Galerkin methods for parabolic problems on prismatic meshes. SIAM J. Sci. Comput. **39**(4), A1251–A1279 (2017)
23. Cangiani, A., Dong, Z., Georgoulis, E.H., Houston, P.: *hp*-Version discontinuous Galerkin methods for advection-diffusion-reaction problems on polytopic meshes. ESAIM Math. Model. Numer. Anal. **50**, 699–725 (2016)
24. Cangiani, A., Dong, Z., Georgoulis, E.H., Houston, P.: *hp*-Version discontinuous Galerkin methods on polygonal and polyhedral meshes. Springer International Publishing, SpringerBriefs in Mathematics, Berlin (2017)
25. Cangiani, A., Georgoulis, E.H., Houston, P.: *hp*-Version discontinuous Galerkin methods on polygonal and polyhedral meshes. Math. Models Methods Appl. Sci. **24**(10), 2009–2041 (2014)
26. Cangiani, A., Manzini, G., Sutton, O.J.: Conforming and nonconforming Virtual Element Methods for elliptic problems. IMA J. Numer. Anal. **37**(3), 1317–1354 (2017)
27. Chin, E.B., Lasserre, J.B., Sukumar, N.: Numerical integration of homogeneous functions on convex and nonconvex polygons and polyhedra. Comput. Mech. **56**(6), 967–981 (2015)
28. Chin, E.B., Lasserre, J.B., Sukumar, N.: Modeling crack discontinuities without element-partitioning in the extended finite element method. Int. J. Numer. Methods Eng. **110**(11), 1021–1048 (2017)
29. Cockburn, B., Dond, B., Guzmán, J.: A superconvergent LDG-hybridizable Galerkin method for second-order elliptic problems. Math. Comput. **77**(264), 1887–1916 (2008)
30. Cockburn, B., Gopalakrishnan, J., Lazarov, R.: Unified hybridization of discontinuous Galerkin, mixed, and continuous Galerkin methods for second order elliptic problems. SIAM J. Numer. Anal. **47**(2), 1319–1365 (2009)
31. Cockburn, B., Gopalakrishnan, J., Sayas, F.J.: A projection-based error analysis of HDG methods. Math. Comput. **79**(271), 1351–1367 (2010)
32. Cockburn, B., Guzmán, J., Wang, H.: Superconvergent discontinuous Galerkin methods for second-order elliptic problems. Math. Comput. **78**(265), 1–24 (2009)
33. Di Pietro, D.A., Ern, A.: A hybrid high-order locking-free method for linear elasticity on general meshes. Comput. Methods Appl. Mech. Eng. **283**(1), 1–21 (2015)
34. Di Pietro, D.A., Ern, A.: Hybrid High-Order methods for variable-diffusion problems on general meshes. C. R. Math. Acad. Sci. Soc. R. Can. **353**(1), 31–34 (2015)
35. Di Pietro, D.A., Ern, A., Lemaire, S.: An arbitrary-order and compact-stencil discretization of diffusion on general meshes based on local reconstruction operators. Comput. Method Appl. Math. **14**(4), 461–472 (2014)
36. Droniou, J., Eymard, R., Herbin, R.: Gradient schemes: generic tools for the numerical analysis of diffusion equations. ESAIM Math. Model. Numer. Anal. **50**(3), 749–781 (2016)
37. Fries, T.P., Belytschko, T.: The extended/generalized finite element method: an overview of the method and its applications. Int. J. Numer. Methods Eng. **84**(3), 253–304 (2010)
38. Gerstenberger, A., Wall, A.W.: An Extended Finite Element Method/Lagrange multiplier based approach for fluid-structure interaction. Comput. Methods Appl. Mech. Eng. **197**(19–20), 1699–1714 (2008)
39. Giani, S., Houston, P.: Domain decomposition preconditioners for discontinuous Galerkin discretizations of compressible fluid flows. Numer. Math. Theory Methods Appl. **7**(2), 123–148 (2014)
40. Griffiths, D.J.: Introduction to Quantum Mechanics. Pearson Education, London (2005)
41. Hackbusch, W., Sauter, S.A.: Composite Finite Elements for problems containing small geometric details. Part II: Implementation and numerical results. Comput. Visual Sci. **1**(4), 15–25 (1997)
42. Hackbusch, W., Sauter, S.A.: Composite Finite Elements for the approximation of PDEs on domains with complicated micro-structures. Numer. Math. **75**(4), 447–472 (1997)
43. Holdych, D.J., Noble, D.R., Secor, R.B.: Quadrature rules for triangular and tetrahedral elements with generalized functions. Int. J. Numer. Methods Eng. **73**(9), 1310–1327 (2015)
44. Hyman, J., Shashkov, M., Steinberg, S.: The numerical solution of diffusion problems in strongly heterogeneous non-isotropic materials. J. Comput. Phys. **132**(1), 130–148 (1997)
45. Karypis, G., Kumar, V.: A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM J. Sci. Comput. **20**(1), 359–392 (1998)
46. Karypis, G., Kumar, V.: Metis: Unstructured graph partitioning and sparse matrix ordering system, version 4.0. http://www.cs.umn.edu/~metis (2009)
47. Lasserre, J.B.: Integration on a convex polytope. Proc. Am. Math. Soc. **126**(8), 2433–2441 (1998)

48. Li, C.J., Lambertu, P., Dagnino, C.: Numerical integration over polygons using an eight-node quadrilateral spline finite element. J. Comput. Appl. Math. **233**(2), 279–292 (2009)
49. Lyness, J.N., Monegato, G.: Quadrature rules for regions having regular hexagonal symmetry. SIAM J. Numer. Anal. **14**(2), 283–295 (1977)
50. Ma, J., Rokhlin, V., Wandzura, S.: Generalized Gaussian quadrature of systems of arbitrary functions. SIAM J. Numer. Anal. **33**(3), 971–996 (1996)
51. Moës, N., Dolbow, J., Belytschko, T.: A finite element method for crack growth without remeshing. Int. J. Numer. Methods Eng. **46**(1), 131–150 (1999)
52. Mousavi, S.E., Sukumar, N.: Numerical integration of polynomials and discontinuous functions on irregular convex polygons and polyhedrons. Comput. Mech. **47**(5), 535–554 (2011)
53. Mousavi, S.E., Xiao, H., Sukumar, N.: Generalized Gaussian quadrature rules on arbitrary polygons. Internat. J. Numer. Methods Eng. **82**(1), 99–113 (2010)
54. Natarajan, S., Bordas, S., Mahapatra, D.R.: Numerical integration over arbitrary polygonal domains based on Schwarz–Christoffel conformal mapping. Int. J. Numer. Methods Eng. **80**(1), 103–134 (2009)
55. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes 3rd Edition: The Art of Scientific Computing, 3rd edn. Cambridge University Press, New York (2007)
56. Qian, H.: Counting the floating point operations (flops). https://it.mathworks.com/matlabcentral/fileexchange/50608-counting-the-floating-point-operations--flops- (2015)
57. Quarteroni, A., Sacco, R., Saleri, F.: Numerical Mathematics. Springer, Berlin (2007)
58. Simon, C.P., Blume, L.E.: Mathematics for Economists. W. W. Norton and Company, New York (1996)
59. Sommariva, A., Vianello, M.: Product Gauss cubature over polygons based on Green's integration formula. BIT **47**(2), 441–453 (2007)
60. Stroud, A.H., Secrest, D.: Gaussiam quadrature formulas. ZAMM Z. Angew. Math. Mech. **47**(2), 138–139 (1967)
61. Sudhakar, Y., Wall, W.A.: Quadrature schemes for arbitrary convex/concave volumes and integration of weak form in enriched partition of unity methods. Comput. Methods Appl. Mech. Eng. **258**(1), 39–54 (2013)
62. Sukumar, N., Moës, N., Belytschko, T.: Extended Finite Element Method for three-dimensional crack modelling. Int. J. Numer. Methods Eng. **48**(11), 1549–1570 (2000)
63. Sukumar, N., Tabarraei, A.: Conforming polygonal finite elements. Int. J. Numer. Methods Eng. **61**(12), 2045–2066 (2004)
64. Tabarraei, A., Sukumar, N.: Extended Finite Element Method on polygonal and quadtree meshes. Comput. Methods Appl. Mech. Eng. **197**(5), 425–438 (2008)
65. Talischi, C., Paulino, G.H., Pereira, A., Menezes, I.F.M.: Polymesher: a general-purpose mesh generator for polygonal elements written in matlab. Struct. Multidiscip. Optim. **45**(3), 309–328 (2012)
66. Taylor, M.E.: Partial Differential Equations: Basic Theory. Springer, New York (1996)
67. Ventura, G.: On the elimination of quadrature subcells for discontinuous functions in the extended finite-element method. Int. J. Numer. Methods Eng. **66**(5), 761–795 (2006)
68. Ventura, G., Benvenuti, E.: Equivalent polynomials for quadrature in Heaviside function enriched elements. Int. J. Numer. Methods Eng. **102**(3–4), 688–710 (2015)
69. Xiao, H., Gimbutas, Z.: A numerical algorithm for the construction of efficient quadrature rules in two and higher dimensions. Comput. Math. Appl. **59**(2), 663–676 (2010)
70. Yarvin, N., Rokhlin, V.: Generalized Gaussian quadratures and singular value decompositions of integral operators. SIAM J. Sci. Comput. **20**(2), 669–718 (1998)
71. Yogitha, A.M., Shivaram, K.T.: Numerical integration of arbitrary functions over a convex and non convex polygonal domain by eight noded linear quadrilateral finite element method. Aust. J. Basic Appl. Sci. **10**(16), 104–110 (2016)