Chapter 6
Implementation Aspects

In this chapter we briefly outline some of the implementation aspects of DGFEMs posed on general computational meshes consisting of polytopic elements. To this end, we focus on three key topics: mesh generation, construction of the elemental polynomial basis, and efficient numerical integration over polytopic elements. Finally, we end this chapter by presenting some numerical examples to highlight the sharpness of the a priori error bounds derived in Chap. 5 for both a steady advection-diffusion-reaction problem and a (degenerate) evolution problem.

6.1 Mesh Generation

General meshes consisting of polytopic elements can be constructed in a number of different ways; in particular, a Voronoi tessellation of the underlying geometry may be generated, cf. [88, 165], for example. Alternatively, a flexible approach for meshing complicated geometries, is to exploit some form of agglomeration algorithm, whereby the underlying polytopic elements are formed by taking the union of a set of elements from a given geometry-conforming fine mesh $\mathscr{T}_h^{\text{fine}}$. We point out that $\mathscr{T}_h^{\text{fine}}$ is typically constructed by employing standard shaped elements, i.e., simplices or tensor-product elements; in this setting, the resulting underlying FEM is often referred to as a *composite* FEM, cf. [8, 12, 106, 107], or an *agglomerated* FEM, cf. [32, 34]. The construction of agglomerated meshes may be undertaken using two key approaches: firstly, in the series of articles [8, 12, 106, 107] a hierarchy of overlapping (so-called) reference and logical meshes, consisting of standard-shaped elements, are constructed based on successive adaptive refinement of elements which intersect the boundary $\partial\Omega$ of the computational domain Ω. Once a sufficiently fine logical mesh has been constructed, possibly by moving nodes onto $\partial\Omega$, a sequence of coarse geometry-conforming physical meshes, consisting

A. Cangiani et al., *hp-Version Discontinuous Galerkin Methods on Polygonal and Polyhedral Meshes*, SpringerBriefs in Mathematics,
DOI 10.1007/978-3-319-67673-9_6

of general polytopic elements, may be constructed via agglomerating elements which share the same parent within the underlying refinement tree. Secondly, the fine mesh $\mathscr{T}_h^{\text{fine}}$ may be constructed using a standard mesh generator, for example, Triangle [157] or Tetgen [158], and then subsequently agglomerated into polytopes using graph partitioning algorithms; for this purpose, in the numerical investigations presented in this volume, we employ METIS [130], though we stress that many other such packages also exist.

6.2 Construction of Basis Functions on Polytopes

In the case when the computational mesh $\mathscr{T}_h$ consists of standard element shapes, the construction of the underlying finite element space $V^{\mathbf{p}}(\mathscr{T}_h)$ is typically undertaken by mapping each element κ in $\mathscr{T}_h$ to a given reference or canonical element, denoted by κ_{R}. Thereby, on κ_{R} local spaces of polynomials may be constructed in a simple manner, subject to the enforcement of any inter-element continuity constraints, for example, in the case when C^0-conforming elements are employed; for further details, we refer to, for example, [78, 129, 160], and the references cited therein. While this approach is used quite universally within most FEM software packages, a disadvantage is that the calculation of high-order derivatives of the computed numerical solution is rather cumbersome when non-affine element mappings are employed. Moreover, in this setting, the order of approximation of the underlying FEM may be adversely affected by mesh distortion, unless a sufficiently rich local space is employed, cf. [17, 19].

The flexibility of the DGFEM allows for the elemental basis to be constructed within the physical frame, without the need to map to a given reference/canonical element, cf. [33], for example; indeed, this is an essential feature of DGFEMs which allows them to admit general polytopic elements in a simple fashion. In [33] basis functions are constructed on general meshes consisting of agglomerated elements, based on employing a Gram-Schmidt orthogonalization process applied to a given set of polynomial functions defined on each κ, $\kappa \in \mathscr{T}_h$. An alternative approach proposed in [54] is based on first defining polynomial spaces over a suitably chosen bounding box of each element κ, $\kappa \in \mathscr{T}_h$; then the element basis is simply constructed by restricting this space to κ. More precisely, given an element $\kappa \in \mathscr{T}_h$, we write B_κ to denote its corresponding bounding box; selecting, for example, B_κ to be the Cartesian bounding box, i.e., the sides of B_κ are aligned with the Cartesian axes, then B_κ can be easily constructed, such that $\bar{\kappa} \subseteq \bar{B}_\kappa$, cf. Figure 6.1 for the case of a polygonal element κ in $\mathbb{R}^2$. On this Cartesian bounding box B_κ we may define a standard polynomial space $\mathscr{P}_{p_\kappa}(B_\kappa)$ spanned by a set of basis functions $\{\phi_{i,\kappa}\}$, $i = 1, \ldots, \dim(\mathscr{P}_{p_\kappa}(B_\kappa))$; note that tensor-product polynomial spaces $\mathscr{Q}_{p_\kappa}(B_\kappa)$ may also be employed, though in the absence of non-affine element mappings, the approximation order of both spaces will be identical.

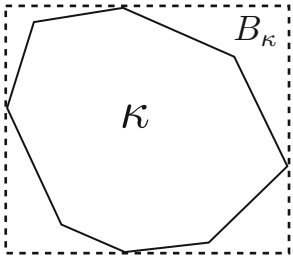


Fig. 6.1 Bounding box B_κ of an element $\kappa \in \mathscr{T}_h$

Writing $B_\kappa := \mathscr{I}_1 \times \mathscr{I}_2 \times \cdots \times \mathscr{I}_d$, where $\mathscr{I}_j := (x_1^j, x_2^j), j = 1, \ldots, d$, and selecting $\kappa_{\mathrm{R}} := (-1, 1)^d$ to be the reference hypercube, the bounding box B_κ may be affinely mapped to κ_{R}, via the mapping

$$\mathbf{x} = J_\kappa \hat{\mathbf{x}} + \mathbf{c}, \tag{6.1}$$

where $J_\kappa := \mathrm{diag}(h_1, \ldots, h_d)$, $\mathbf{c} := (m_1, \ldots, m_d)^\top$, and $\hat{\mathbf{x}}$ is a general point in κ_{R}. Furthermore, $h_j, j = 1, \ldots, d$, is half of the length of the jth-side of B_κ, respectively, i.e., $h_j := (x_2^j - x_1^j)/2, j = 1, \ldots, d$, and $m_j := (x_1^j + x_2^j)/2, j = 1, \ldots, d$, is the midpoint of $\mathscr{I}_j$.

For convenience, on κ_{R} we employ tensor-product Legendre polynomials; to this end, writing $\{\hat{L}_i(\hat{x})\}_{i=0}^{\infty}$ to denote the family of $L^2(-1, 1)$-orthogonal Legendre polynomials, cf. [156], for example, the space of polynomials $\mathscr{P}_{p_\kappa}(\kappa_{\mathrm{R}})$ of total degree p_k over κ_{R} is given by

$$\mathscr{P}_{p_\kappa}(\kappa_{\mathrm{R}}) := \mathrm{span}\{\hat{\phi}_{i,\kappa}\}_{i=1}^{\dim(\mathscr{P}_{p_\kappa}(\kappa_{\mathrm{R}}))},$$

where

$$\hat{\phi}_{i,\kappa}(\hat{\mathbf{x}}) = \hat{L}_{i_1}(\hat{x}_1)\hat{L}_{i_2}(\hat{x}_2)\cdots\hat{L}_{i_d}(\hat{x}_d), \quad i_1 + i_2 + \ldots + i_d \leq p_\kappa, \;\; i_k \geq 0, \; k = 1, \ldots, d,$$

and $\hat{\mathbf{x}} = (\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_d)$. Writing $L_i(x) = \hat{L}_i((x - m_j)/h_j)$, under the transformation (6.1), the space of polynomials $\mathscr{P}_{p_\kappa}(B_\kappa)$ of total degree p_k over B_κ is given by

$$\mathscr{P}_{p_\kappa}(B_\kappa) = \mathrm{span}\{\phi_{i,\kappa}\}_{i=1}^{\dim(\mathscr{P}_{p_\kappa}(B_\kappa))},$$

where

$$\phi_{i,\kappa}(\mathbf{x}) = L_{i_1}(x_1)L_{i_2}(x_2)\cdots L_{i_d}(x_d), \quad i_1 + i_2 + \ldots + i_d \leq p_\kappa, \;\; i_k \geq 0, \; k = 1, \ldots, d,$$

and $\mathbf{x} = (x_1, x_2, \ldots, x_d)$. Thereby, the polynomial basis over the general polytopic element κ may be defined by simply restricting the support of $\{\phi_{i,\kappa}\}$, $i =$

$1, \ldots, \dim(\mathscr{P}_{p_\kappa}(B_\kappa))$ to κ; i.e., the polynomial basis defined over κ is given by $\{\phi_{i,\kappa}|_\kappa\}$, $i = 1, \ldots, \dim(\mathscr{P}_{p_\kappa}(B_\kappa))$.

Remark 59 We stress that the choice of B_κ is arbitrary; indeed, alternative bounding boxes, other than the Cartesian aligned one presented above, may be employed, provided that the construction of the underlying polynomial basis may be undertaken in a simple fashion. For example, for anisotropic polytopic elements, it may be more desirable to select a 'rotated' bounded box which is aligned with the principle axes of the element.

6.3 Quadrature Rules

The design of efficient and accurate quadrature rules for general polytopes is a challenging task; while several approaches have been proposed within the literature, cf. below, this still remains an open and active area of research. Below we review three prominent approaches which have been proposed; for further details, we refer to [14].

6.3.1 Sub-Tessellation

The simplest, and perhaps most natural approach is to simply construct a sub-tessellation of each polytopic element into standard element shapes, upon which standard quadrature rules may be employed, cf. [54, 55, 125], for example. More precisely, given $\kappa \in \mathscr{T}_h$, we first construct a non-overlapping sub-tessellation $\kappa_\mathscr{S} := \{\tau_\kappa\}$ consisting of standard element shapes; here, a general hybrid sub-tessellation consisting of quadrilateral and triangular elements in $\mathbb{R}^2$, or tetrahedral, hexahedral, prismatic, and pyramidal elements in $\mathbb{R}^3$, may be constructed. On agglomerated meshes, the sub-tessellation will already be available; however, for reasons of efficiency, one may still wish to construct an alternative sub-tessellation which comprises of a minimal number of elements. As an example, if we consider computing the DGFEM mass matrix, restricted to κ, $\kappa \in \mathscr{T}_h$, then we have that

$$\int_\kappa wv \,\mathrm{d}\mathbf{x} = \sum_{\tau_\kappa \in \kappa_\mathscr{S}} \int_{\tau_\kappa} wv \,\mathrm{d}\mathbf{x} \approx \sum_{\tau_\kappa \in \kappa_\mathscr{S}} \sum_{i=1}^{q_{\tau_\kappa}} w(F_\kappa(\boldsymbol{\xi}_i)) v(F_\kappa(\boldsymbol{\xi}_i)) \det(J_{F_\kappa}(\boldsymbol{\xi}_i)) w_i,$$

where $F_\kappa : \kappa_\mathrm{R} \to \tau_\kappa$ is the mapping from the reference element κ_R to τ_κ, with Jacobi matrix J_{F_κ}, and $(\boldsymbol{\xi}_i, w_i)_{i=1}^{q_{\tau_\kappa}}$ denotes the quadrature rule defined on κ_R. Quadrature rules on standard element shapes which form the sub-tessellation may be constructed based on employing tensor-product Gauss quadratures on the reference square or cube in $\mathbb{R}^2$ or $\mathbb{R}^3$, respectively; for non-tensor-product elements, such as simplices, for example, the resulting quadrature may be computed based on

employing the Duffy transformation, whereby the reference tensor-product element is mapped to the reference simplex, cf. [86, 129, 171]. For alternative quadratures on non-tensor-product elements, we refer to, for example, [78, 87, 160], and the references cited therein.

An alternative approach based on employing Stokes' theorem in $\mathbb{R}^2$ over polygons has been studied in [161]. While this idea does not directly require a sub-tessellation of the underlying element domain κ, $\kappa \in \mathscr{T}_h$, a judicious choice of parameters within the resulting formula, which ensures that all of the quadrature points lie within κ, essentially leads to a compound quadrature scheme defined on an appropriate sub-tessellation of κ.

We point out that while quadrature schemes based on employing a sub-tessellation of each polytopic element are straightforward to implement, they tend to be computationally expensive, in the sense that, depending on the cardinality of the sub-tessellation, the number of required function evaluations may be very large. This is particularly the case when the sub-tessellation employed is simply the background fine mesh $\mathscr{T}_h^{\text{fine}}$ used to construct a coarse agglomerated grid. Thereby, it is desirable to attempt to minimise the resulting number of points; one such approach is outlined in the next section.

6.3.2 Moment Quadratures

Quadrature rules such as those based on sub-tessellation outlined above can be optimized based on employing a node elimination scheme, together with the least squares Newton method, cf. [175]; for related work, we refer to [141]. In this way, (near)-minimal quadrature schemes can be constructed. To illustrate this approach, following [175], let $\kappa \in \mathscr{T}_h$ be a polytopic element. Given a user-defined set of functions $\mathscr{V}_\kappa = \{\phi_1, \phi_2, \ldots, \phi_n\}$, $n \geq 1$, defined over κ, and a quadrature rule $(\mathbf{x}_j, w_j)_{j=1}^{q_\kappa}$ on κ, $q_\kappa \geq n$, which is *assumed* to integrate all functions in $\mathscr{V}_\kappa$ exactly, we arrive at the following system of equations:

$$A\mathbf{w} = \mathbf{I}, \tag{6.2}$$

where A is an $n \times q_\kappa$ matrix with entries $A_{ij} := \phi_i(\mathbf{x}_j)$, $i = 1, \ldots, n$, $j = 1, \ldots, q_\kappa$, $\mathbf{w} := (w_1, \ldots, w_{q_\kappa})^\top$ is the vector of quadrature weights, and $\mathbf{I}$ is a vector of dimension n, with entries $\mathbf{I}_i := \int_\kappa \phi_i \, \mathrm{d}\mathbf{x}$, $i = 1, \ldots, n$. We note that as in [175] a weight function ω may also be included within the integral; for simplicity, here we set $\omega \equiv 1$.

The initial quadrature rule $(\mathbf{x}_i, w_i)_{i=1}^{q_\kappa}$ may be selected in a number of different ways; for example, the sub-tessellation approach outlined above may be exploited. The essential idea to optimise the initial quadrature is to continuously eliminate points until the solution of (6.2) can no longer be determined. More precisely, for each quadrature point and weight, the corresponding *significance index* s_j,

$j = 1, \ldots, q_\kappa$, is computed; in [175] the following two expressions are proposed:

$$s_j := w_j \sum_{i=1}^{n} \phi_i^2(\mathbf{x}_j), \quad \text{or} \quad s_j := \sum_{i=1}^{n} \phi_i^2(\mathbf{x}_j),$$

$j = 1, \ldots, q_\kappa$. Then, the quadrature point and weight $(\mathbf{x}_k, w_k)$, for some k, $1 \leq k \leq q_\kappa$, which has the smallest significance factor is removed from the quadrature rule, i.e., $s_k := \min_{j=1}^{q_\kappa} s_j$. The least-squares version of Newton's method is then applied to (6.2) to compute a new quadrature $(\mathbf{x}_j, w_j)_{j=1}^{q_\kappa - 1}$ with $(q_\kappa - 1)$ points and weights. This process is continuously repeated until the Newton algorithm fails to converge; at the end of this process, an *optimized* quadrature rule which can precisely integrate all of the functions present in the space $\mathscr{V}_\kappa$ will be computed. An alternative approach proposed in [140] is to simply fix a set of quadrature points, which may even be randomly located points, and solve (6.2) to determine the corresponding weights.

While this approach is very appealing from a computational point of view, this process must first be applied to all elements κ in the computational mesh $\mathscr{T}_h$, and the resulting quadrature scheme stored elementwise, before assembly of the underlying matrix system can be undertaken. We also stress that while the quadrature constructed using the above algorithm is exact for the set of functions in $\mathscr{V}_\kappa$, their accuracy in terms of integrating general functions is unclear; moreover, for general polytopes, the resulting quadrature weights may be negative, cf. [140].

6.3.3 Integration of Homogeneous Functions

Lasserre's method for integrating homogeneous functions over convex polytopes was first introduced in [134]; this technique was subsequently extended to general non-convex polytopes in the recent article [63]. The essential idea here is to exploit Stokes' theorem, together with Euler's homogeneous function theorem. More precisely, given a polytopic element $\kappa \in \mathscr{T}_h$, and a sufficiently regular function f, defined over κ, we wish to evaluate

$$\int_\kappa f \, \mathrm{d}\mathbf{x}.$$

Assuming that f is a positively homogeneous function of degree q, i.e.,

$$f(\lambda \mathbf{x}) = \lambda^q f(\mathbf{x}),$$

for $\lambda > 0$, then assuming f is continuously differentiable, Euler's homogeneous function theorem states that

$$q f(\mathbf{x}) = \mathbf{x} \cdot \nabla f(\mathbf{x}). \tag{6.3}$$

Moreover, given any vector-valued function $\mathbf{g}$, again assumed to be sufficiently smooth, Stokes' theorem states that

$$\int_\kappa (\nabla \cdot \mathbf{g}) f \, \mathrm{d}\mathbf{x} = \int_{\partial\kappa} (\mathbf{g} \cdot \mathbf{n}_\kappa) f \, \mathrm{d}s - \int_\kappa \mathbf{g} \cdot \nabla f \, \mathrm{d}\mathbf{x}, \tag{6.4}$$

where $\mathbf{n}_\kappa$ denotes the unit outward normal vector to the boundary $\partial\kappa$ of κ. Thereby, selecting $\mathbf{g} = \mathbf{x}$ in (6.4) and employing (6.3), we deduce that

$$\int_\kappa f \, \mathrm{d}\mathbf{x} = \frac{1}{d+q} \int_{\partial\kappa} (\mathbf{x} \cdot \mathbf{n}_\kappa) f \, \mathrm{d}s; \tag{6.5}$$

i.e., the integral over κ is reduced to an integration over the boundary $\partial\kappa$. Writing $\partial\kappa = \bigcup_{i=1}^{n_F} F_i$, where F_i, $i = 1, \ldots, n_F$, denote the planar $(d-1)$-dimensional facets which form the boundary of κ, Eq. (6.5) may be rewritten in the following equivalent form

$$\int_\kappa f \, \mathrm{d}\mathbf{x} = \frac{1}{d+q} \sum_{i=1}^{n_F} \int_{F_i} (\mathbf{x} \cdot \mathbf{n}_{F_i}) f \, \mathrm{d}s, \tag{6.6}$$

where $\mathbf{n}_{F_i}$ denotes the restriction of the unit outward normal vector $\mathbf{n}_\kappa$ to the facet F_i, $i = 1, \ldots, n_F$.

We note that this process can be repeated in order to yield a formula which involves integration on lower dimensional facets. For example, given F_i, for some (fixed) i, $1 \le i \le n_F$, we write

$$\partial F_i = \{F_{ij} = F_i \cap F_j : F_i \cap F_j \neq \emptyset, \ i \neq j\}$$

to denote the set of $(d-2)$-dimensional facets of κ, i.e., F_{ij} is an edge of a polyhedron in $\mathbb{R}^3$ which lies on the boundary of the face F_i. Furthermore, we define $\mathbf{n}_{F_{ij}}$ to be the unit normal vector to F_{ij} which lies in the plane F_i. Given an arbitrary point $\mathbf{x}_i \in F_i$ and a $(d-1)$-dimensional orthonormal basis $\{\mathbf{e}_j^i\}_{j=1}^{d-1}$ on the facet F_i, i.e., any $\mathbf{x} \in F_i$ may be written in the form

$$\mathbf{x} = \mathbf{x}_i + \sum_{k=1}^{d-1} \alpha_k \mathbf{e}_k^i,$$

for some scalars α_k, $k = 1, \ldots, d-1$. Then, upon application of (6.4) to a given facet F_i, $1 \le i \le n_F$, with $\mathbf{g} = \mathbf{x} - \mathbf{x}_i$, we deduce that

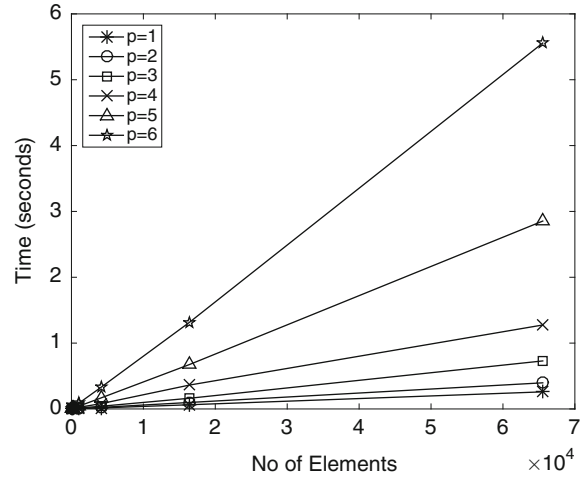
$$\int_{F_i} f \, \mathrm{d}s = \frac{1}{d+q-1} \left(\sum_{F_{ij} \subset \partial F_i} \int_{F_{ij}} ((\mathbf{x} - \mathbf{x}_j) \cdot \mathbf{n}_{F_{ij}}) f \, \mathrm{d}s + \int_{F_i} (\mathbf{x}_i \cdot \nabla f) \, \mathrm{d}s \right), \tag{6.7}$$

cf. [63, 134]. In $\mathbb{R}^2$, we note that the first term on the right-hand side of (6.7) simply involves evaluations of the integrand at the points which form the underlying face (edge) F_i, $1 \leq i \leq n_F$. Thereby, for the integration of polynomial functions, repeated application of (6.7) yields an (exact) integration rule which only requires evaluation of f and its partial derivatives at the vertices of κ, $\kappa \in \mathcal{T}_h$. In $\mathbb{R}^3$, proceeding as above, we first rewrite the integral over F_{ij} to be an integral over its $(d-2)$-dimensional facets (points), and an integral involving the derivative of f over F_{ij}; thereby, again for polynomial functions, recursive application of this formula then only requires the evaluation of f and its partial derivatives at the vertices of κ in order to precisely evaluate the integral of f over κ, $\kappa \in \mathcal{T}_h$. For further details and implementation of this approach in the context of DGFEMs, we refer to [14]. To give an example of the potential performance improvement of employing this approach in comparison to the use of quadrature schemes defined on a sub-tessellation of each polytopic element κ in the mesh $\mathcal{T}_h$, in Fig. 6.2 we compare the time taken to assemble the local element stiffness matrix for the Poisson equation in two-dimensions, cf. the DGFEM bilinear form given in Sect. 4.1, cf. also Sect. 2.3. Here, the polygonal meshes are constructed using PolyMesher [165], cf. above. For clarity, in Fig. 6.2a we plot the time taken to assemble the element stiffness matrix via exact integration of homogeneous functions on a series of uniform polygonal meshes for polynomial degrees p between 1 and 6. Figure 6.2b then presents a comparison of these results with the corresponding timings based on employing quadrature on a sub-tessellation. Here, the sub-tessellation is constructed by inserting one internal point within each element κ, $\kappa \in \mathcal{T}_h$, located at the element centroid, and connecting this point to each pair of nodes defining the faces of κ; thereby, the number of elements within the sub-tessellation is equal to the number of faces that each element $\kappa \in \mathcal{T}_h$ possesses.

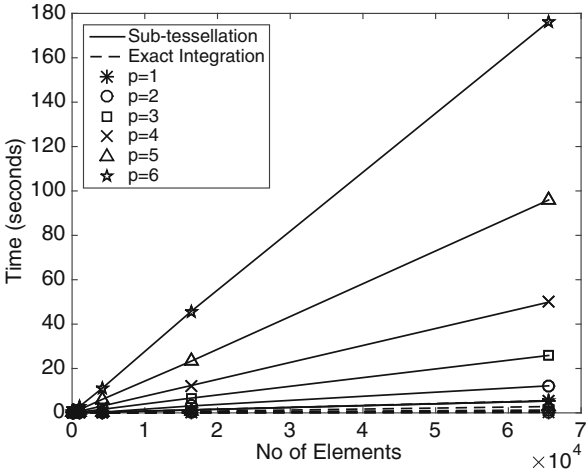
As a final remark, we note that this approach may also be extended to integrate functions which may be represented as the sum of homogeneous functions; in particular, the integral may be computed without explicit knowledge of the individual terms which form this sum. However, this leads to the introduction of evaluation points within the interior of κ, $\kappa \in \mathcal{T}_h$, and even points which may lie outside κ; in the simplest case, for example, for star-shaped polytopes, this can be viewed as undertaking a subdivision, relative to a point lying in the ball for which κ is star-shaped, cf. [63], for further details.

6.4 Numerical Experiments

In this section we present two computational examples to numerically to highlight the practical performance of the DGFEMs studied in Chap. 5 on general polytopic meshes; see also [54–56] for further numerical experiments.



(a)



(b)

Fig. 6.2 Time required to assemble the element stiffness matrix for the Poisson equation in two-dimensions: (**a**) Exploiting exact integration of homogeneous functions; (**b**) Comparison between quadrature employed on a sub-tessellation and exact integration

6.4.1 Example 1: Advection-Diffusion-Reaction Problem

We consider the discretization of the advection-diffusion-reaction problem: find u such that

$$-\nabla \cdot (a\nabla u) + \nabla \cdot (\mathbf{b}u) + cu = f \quad \text{in } (0,1)^2, \tag{6.8}$$

where $a = \delta I_2$, $\delta = \epsilon\, \mathrm{e}^{-20r}$, $r^2 = x^2 + y^2$, $\epsilon > 0$, $\mathbf{b} = (2 - y^2, 2 - x)^\top$, and $c = (1+x)(1+y)^2$; f is then selected so that the analytical solution to (6.8), subject to appropriate inhomogeneous boundary conditions, is given by

$$u(x, y) = 1 + \sin(\pi(1 + x)(1 + y)^2/8),$$

cf. [123].

We study the asymptotic behaviour of the DGFEM (5.8) on a sequence of successively finer square and polygonal meshes for different values of the polynomial degree p in both the diffusion-dominated and advection-dominated regimes. In each case we consider two types of polygonal meshes: grids generated using PolyMesher [165], cf. Fig. 6.3a, as well as grids stemming from the agglomeration of a given (fixed) fine mesh $\mathscr{T}_h^{\text{fine}}$. In the latter case, we employ a fine mesh consisting of 262,144 elements; the corresponding coarse agglomerated mesh $\mathscr{T}_h$ is then constructed by exploiting the graph partitioning package METIS [130]. We note that for METIS to partition the fine mesh $\mathscr{T}_h^{\text{fine}}$, the logical structure of $\mathscr{T}_h^{\text{fine}}$ is first stored in the form of a graph, where each node represents an element domain of $\mathscr{T}_h^{\text{fine}}$, and each link between two nodes represents a face shared by the two elements represented by the graph nodes. The resulting partition of $\mathscr{T}_h^{\text{fine}}$ constructed by METIS is produced with the objective of minimizing the number of neighbours among each of the resulting partitions, or more precisely, the resulting polygonal elements. In Fig. 6.3b we show the resulting polygonal mesh generated by METIS with 256 elements.

We first study the diffusion-dominated case; to this end, we set $\epsilon = 1$. In Fig. 6.4 we investigate the convergence of the DGFEM on sequences of finer square and polygonal meshes with polynomial degrees p between 1 and 4. In each case we

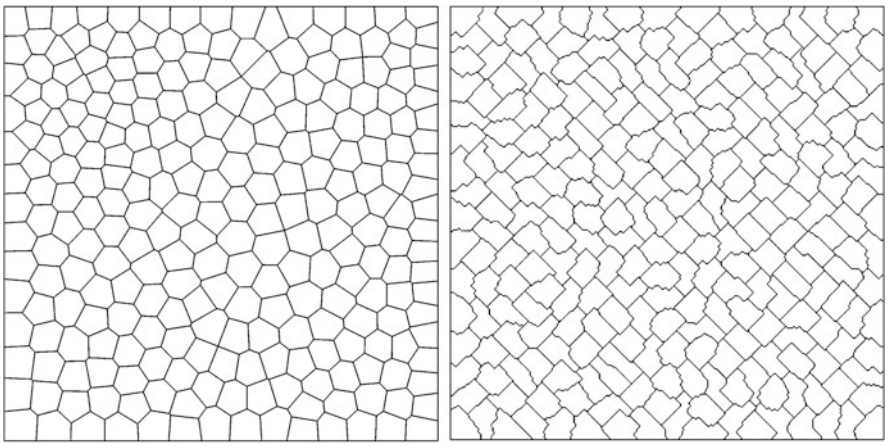


Fig. 6.3 Example 1. Polygonal meshes consisting of 256 elements generated based on employing: (**a**) Voronoi tessellation generated by PolyMesher [165]; (**b**) Agglomeration using METIS [130]

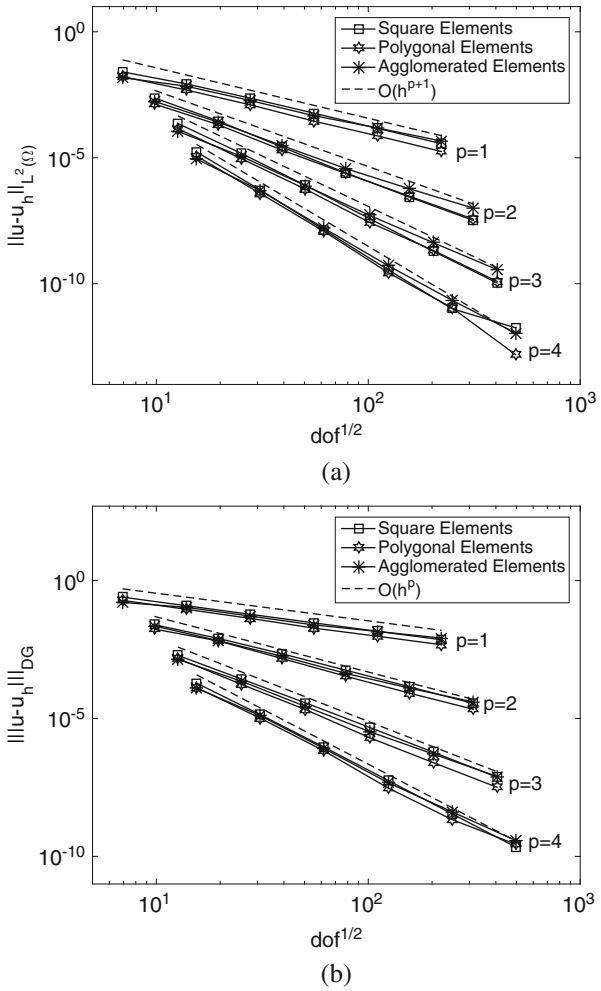


Fig. 6.4 Example 1. Convergence of the DGFEM with h-refinement for $\epsilon = 1$: (**a**) $\|u - u_h\|_{L^2(\Omega)}$; (**b**) $|||u - u_h|||_{\mathrm{DG}}$

plot the error, measured in terms of both the $L^2(\Omega)$ and DGFEM norm, against the square root of the number of degrees of freedom in the underlying finite element space $V^{\mathbf{p}}(\mathscr{T}_h)$. Here, we clearly observe that $\|u-u_h\|_{L^2(\Omega)}$ and $|||u - u_h|||_{\mathrm{DG}}$ converge to zero at the optimal rates $\mathscr{O}(h^{p+1})$ and $\mathscr{O}(h^p)$, respectively, as the mesh size h tends to zero for each fixed p. The latter set of numerical results confirm the optimality of Theorem 46 in the diffusion-dominated setting, cf. Remark 47. Moreover, from Fig. 6.4 we observe that the accuracy of the DGFEM is very similar on all three sets of meshes employed here, given the same number of degrees of freedom, though in general we observe a slight improvement in $|||u - u_h|||_{\mathrm{DG}}$ when the polygonal elements generated by Polymesher are employed, in comparison to the

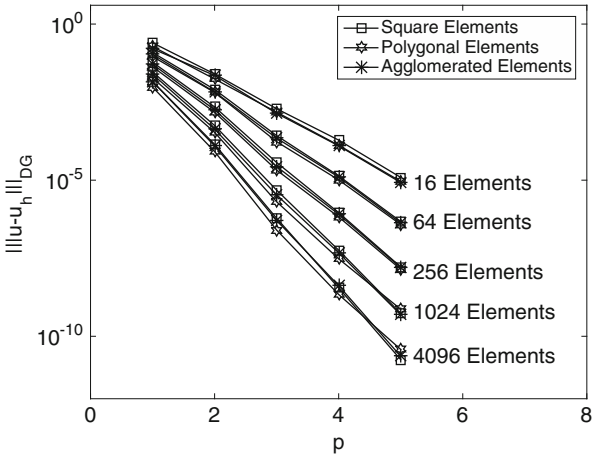


Fig. 6.5 Example 1. Convergence of the DGFEM with p-refinement for $\epsilon = 1$

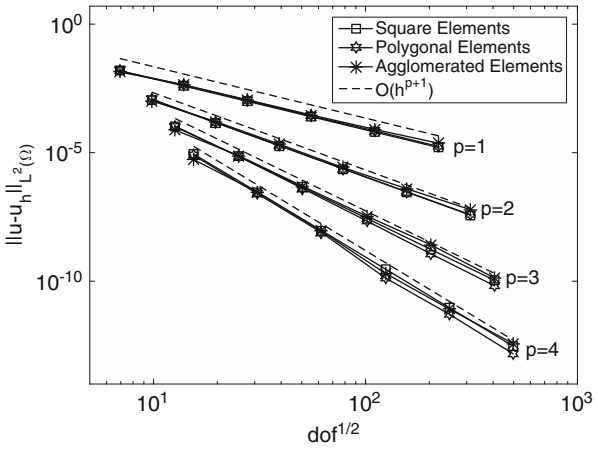
corresponding quantity computed based on exploiting either square or agglomerated meshes. In Fig. 6.5 we investigate the convergence behaviour of the DGFEM under p-refinement, for a given fixed mesh. For each mesh, we plot $|||u - u_h|||_{\mathrm{DG}}$ against the polynomial degree p on a linear-log scale; in each case we clearly observe exponential convergence of the DGFEM.

Secondly, we consider the convergence of the DGFEM in the advection-dominated regime, whereby, we now set $\epsilon = 10^{-6}$. In Fig. 6.6 we plot $\|u - u_h\|_{L^2(\Omega)}$ and $|||u - u_h|||_{\mathrm{DG}}$ against the square root of the number of degrees of freedom in the underlying finite element space $V^{\mathbf{p}}(\mathscr{T}_h)$ on a sequence of uniform square and polygonal meshes for fixed p. In this case, we observe that while the $L^2(\Omega)$ norm of the error converges to zero at the optimal rate $\mathscr{O}(h^{p+1})$, the DGFEM norm behaves like $\mathscr{O}(h^{p+1/2})$, as h tends to zero, for each fixed p; in the latter case, this is indeed the optimal rate predicted by Theorem 46, cf. Remark 48. As above, we again observe a slight improvement in the computed error when the Voronoi meshes generated by PolyMesher are employed, as opposed to square or agglomerated meshes. Figure 6.7 plots $|||u - u_h|||_{\mathrm{DG}}$ against the polynomial degree p for a given set of fixed uniform meshes; as above, we observe exponential convergence of the DGFEM under p-refinement.

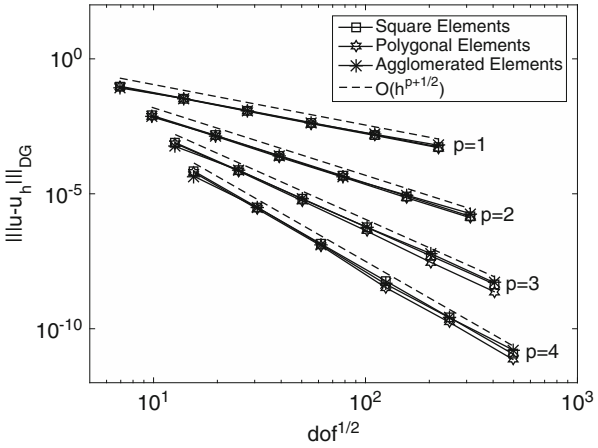
For further numerical experiments, and in particular, for comparisons between both standard (conforming) Galerkin FEMs and DGFEMs employing local tensor-product polynomial bases ($\mathscr{Q}_p$-basis), we refer to the articles [54, 55].

6.4.2 *Example 2: Degenerate Evolution Equation*

Computational experiments for the space-time DGFEM (5.54), cf., also, (5.56), have been presented in the recent article [56] for the case when $\mathbf{a}$ is positive definite,



(a)



(b)

Fig. 6.6 Example 1. Convergence of the DGFEM with h-refinement for $\epsilon = 10^{-6}$: (**a**) $\|u - u_h\|_{L^2(\Omega)}$; (**b**) $|||u - u_h|||_{\mathrm{DG}}$

i.e., when (5.52) holds. Thereby, to test the necessity of this hypothesis in terms of generalizing the a priori error bounds derived in Sect. 5.3 to degenerate parabolic PDEs, we consider the following example: find $u = u(t, x, y)$ such that

$$\partial_t u - x^2 \partial^2_{xx} u - x \partial_y u = f \quad \text{in } \Omega := J \times D, \tag{6.9}$$

where $J = (0, 1)$ and $D = (0, 1)^2$. We note that (6.9) can be written in the general form (2.1), cf., also, (5.1); indeed, setting $\mathbf{x} := (t, x, y)^\top$ and $\nabla := (\partial_t, \partial_x, \partial_y)^\top$, we select $a_{22} = x^2$, $a_{ij} = 0$ for $i, j = 1, 2, 3$, $i, j \neq 2$, $\mathbf{b} = (1, 2x, -x)^\top$, and $c = -2$. Furthermore, we stress that the PDE (6.9) is not hypoelliptic on the plane $\{x = 0\}$

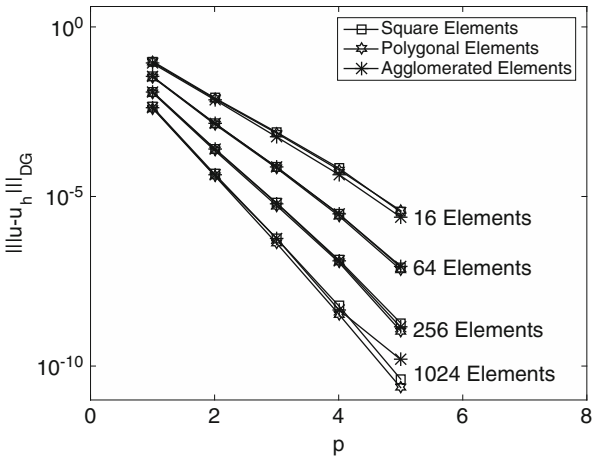


Fig. 6.7 Example 1. Convergence of the DGFEM with p-refinement for $\epsilon = 10^{-6}$

which is contained in $J \times \partial D$ and may, therefore, lead to singularities in the analytical solution u in the vicinity of the boundary, depending on the choice of the boundary conditions imposed and/or the selection of f.

In this section, the forcing function f is selected so that, upon supplementing (6.9) with appropriate inhomogeneous Dirichlet boundary conditions and initial condition, the analytical solution is given by

$$u(t,x,y) = \mathrm{e}^{-5((x-1/2)^2+(y-1/2)^2)}\ \sin(x-t+2y).$$

In Fig. 6.8 we investigate the convergence of the space-time DGFEM on sequences of finer (space-time) hexahedral (rectangular in space) and prismatic (polygonal prism base in space) meshes, for polynomial degrees p between 1 and 4. In each case we plot the error measured in terms of both the $L^2((0,1);L^2(D))$ and $L^2((0,1);H^1(D,\mathscr{T}_h))$ norms with respect to the third root of the number of degrees of freedom in the underlying space-time finite element space $V^{\mathbf{p}}(\mathscr{U};\mathscr{T}_h)$. Here, we employ spatial meshes consisting of 16, 64, 256, 1024, and 4096 elements, with 8, 16, 32, 64, and 128 time-steps, respectively. Firstly, from Fig. 6.8b we observe that $\|u-u_h\|_{L^2((0,1);H^1(D,\mathscr{T}_h))}$ converges to zero at the optimal rate $\mathscr{O}(h^p)$ as the space-time mesh size h tends to zero for each fixed p; this is in accordance with the rate predicted by Corollary 56, though we stress that in this generalized setting, the conditions of Theorem 55 (and, therefore, of Corollary 56) are not fulfilled. Secondly, from Fig. 6.8a we observe some deterioration in the rate of convergence of the space-time DGFEM with respect to the $L^2((0,1);L^2(D))$ norm, cf. the discussion at the end of Sect. 5.3; indeed these numerics seem to indicate a loss of (roughly) between half and one order in h, as the mesh is refined for each fixed p.

Finally, we investigate the computational efficiency of employing the space-time DGFEM (5.54) with different space-time elemental polynomial bases on

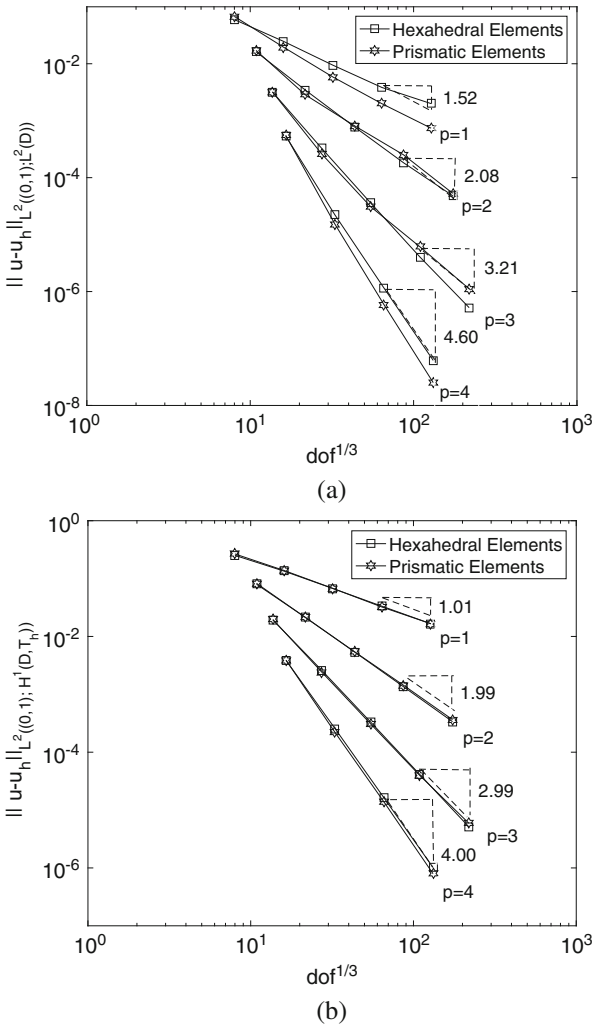
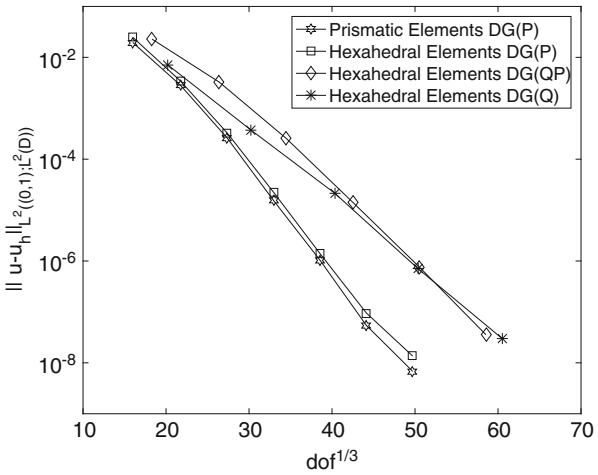


Fig. 6.8 Example 2. Convergence of the space-time DGFEM with h-refinement on cubic (square elements in space) and prismatic (polygonal elements in space) meshes: (**a**) $\|u - u_h\|_{L^2((0,1);L^2(D))}$; (**b**) $\|u - u_h\|_{L^2((0,1);H^1(D,\mathscr{T}_h))}$

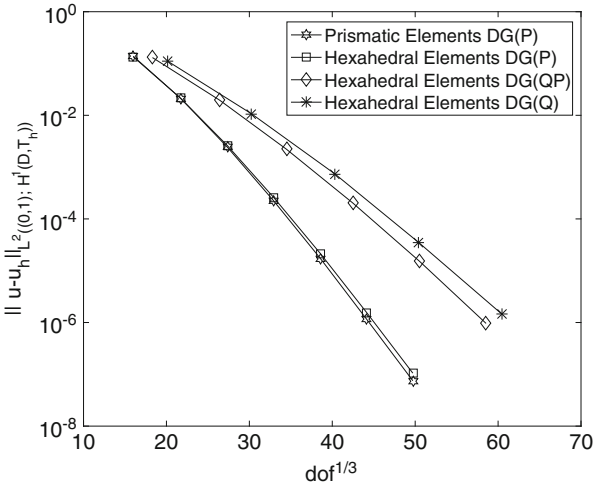
both hexahedral and prismatic meshes. In particular, writing $\kappa_n := I_n \times \kappa$ to denote a given space-time element, where $\kappa \in \mathscr{T}_h$ is a spatial element and I_n, $n = 1, \ldots, N_t$, is a given time interval, we consider the following three choices for the (elementwise) polynomial space, denoted by $\mathscr{R}_p(\kappa_n)$: (1) $\mathscr{R}_p(\kappa_n) := \mathscr{P}_p(\kappa_n)$, i.e., polynomials of total degree p are employed on each element, cf. Sect. 5.3; (2) $\mathscr{R}_p(\kappa_n) := \mathscr{P}_p(I_n) \times \mathscr{P}_p(\kappa)$, i.e., polynomials of total degree p are employed in time and space, and these are tensorized to form a space-time basis defined on κ_n; (3) $\mathscr{R}_p(\kappa_n) := \mathscr{P}_p(I_n) \times \mathscr{Q}_p(\kappa)$, i.e., polynomials of total degree p are employed

in time, while tensor-product polynomials of degree p in each coordinate direction are exploited in space; these spaces are then tensorized to form a space-time basis defined on κ_n. These schemes will be denoted, respectively, by DG(P), DG(QP), and DG(Q).

In Fig. 6.9 we investigate the convergence behaviour of these three schemes under p-refinement for fixed h; for brevity, we only consider space-time prismatic meshes for the first scheme, i.e., for DG(P). Here, we have employed 64 spatial



(a)



(b)

Fig. 6.9 Example 2. Comparison of the space-time DGFEMs: DG(P), DG(QP), and DG(Q) under p-refinement with 64 spatial elements and 16 time-steps: (**a**) $\|u - u_h\|_{L^2((0,1);L^2(D))}$; (**b**) $\|u - u_h\|_{L^2((0,1);H^1(D,\mathscr{T}_h))}$

elements and 16 time-steps. Firstly, we observe exponential convergence for all three choices of the elemental polynomial bases, in the sense that, on the linear-log scale, the convergence plots become straight lines as p is increased. Moreover, we observe that, under p-refinement, DG(P) is more efficient than both DG(QP) and DG(Q), in the sense that the error, computed in terms of both the $L^2((0,1);L^2(D))$ and $L^2((0,1);H^1(D,\mathscr{T}_h))$ norms is smaller, for a given number of degrees of freedom, when the former scheme is employed.