CURVILINEAR MESH GENERATION FOR THE HIGH-ORDER VIRTUAL ELEMENT METHOD (VEM)

Kaloyan Kirilov[1] Joaquim Peiró[1] Mashy Green[2] David Moxey[2]
Lourenço Beirão da Veiga[3] Franco Dassi[3] Alessandro Russo[3]

[1]*Department of Aeronautics, Imperial College London, U.K.*
{kaloyan.kirilov19;j.peiro}@imperial.ac.uk
[2]*Department of Engineering, King's College London, U.K.*
{mashy.green;david.moxey}@kcl.ac.uk
[3]*Department of Mathematics and Applications, University of Milano-Bicocca, Italy.*
franco.dassi;alessandro.russo;lourenco.beirao@unimib.it

ABSTRACT

We present a proof-of-concept methodology for generating curvilinear polygonal meshes suitable for high-order discretizations by the Virtual Element Method (VEM). A VEM discretization requires the definition of a set of boundary and internal points that are used to interpolate the approximation functions and to evaluate integrals by means of suitable quadratures. The procedure to locate these points on the boundary borrows ideas from previous work on *a posteriori* high-order mesh generation in which the geometrical inquiries to a B-rep of the computational domain are performed via an interface to CAD libraries. Here we describe the steps of the procedure that transforms a straight-sided polygonal mesh, generated using third-party software, into a curvilinear boundary-conforming mesh. We discuss criteria for ensuring and verifying the validity of the mesh. Finally, using the Laplace equation with Dirichlet boundary conditions as a model problem, we show that VEM discretizations on such meshes achieve the expected rates of convergence as the mesh resolution is increased.

Keywords: Virtual element method; High-order mesh generation; Curvilinear polygonal meshes

1. INTRODUCTION

Polytopal meshes, with arbitrarily shaped 2D polygonal or 3D polyhedral computational cells, have been routinely used for the discretization of partial differential equations (PDEs) by finite volume methods [1]. The main advantages of using unstructured polytopal meshes are their ability to discretize complex computational domains and their potential to reduce the computational complexity of the PDE solver. Recently, there has been a growing interest in developing discretisation methods that support polygonal/polyhedral cell shapes meshes with low and high approximation orders. A literature review of the large variety of polytopal methods is given in reference [2].

The majority of these methods make use of polytopal meshes with straight edges and faces that, especially for high-order methods, can deteriorate the accuracy of the solution in the presence of curved boundaries or interfaces. As it is well known from the finite element method literature, the representation of the domain geometry with planar facets introduces an error that can stagnate convergence if the order of the approximation is increased. It is therefore important to ensure that the curved interfaces and boundaries are accurately approximated to guarantee the expected order of convergence. To achieve this, one should aim at defining discrete spaces on curved elements in such a way that the domain geometry is accurately represented. Examples of this are the high-order polyno-

mial maps employed in isoparametric finite elements [3], and the use of a CAD representation of the computational domain in isogeometric analysis [4]. The Virtual Element Method (VEM) [5] is arguably one of the very few approaches that permits the definition of those discrete spaces in the context of curvilinear polytopal meshes.

To the best of our knowledge, very few methods exist for the generation of boundary-conforming curvilinear polytopal meshes. One exception is reference [6] that adopts a NURBS-enhanced VEM strategy where the edges on boundaries and interfaces are NURBS curves, each defined to exactly match the CAD description of the boundary. The approach that we propose here differs for that in reference [6] in that the geometrical information required by the VEM discretization is obtained through an application programming interface (API) that performs all the required geometrical enquiries on a standard CAD representation of the boundary which eliminates the need to define an individual NURBS representation of each edge that matches the CAD definition. This is more general, employs similar procedures to those employed by current state-of-the-art curvilinear high-order mesh generators [7, 8], and thus facilitates the extension of the methodology to 3D problems. However, we will present the main ideas using a 2D proof-of-concept in the following sections.

2. HIGH-ORDER VEM BASICS

This section describes the basics of the virtual element method for the discretization of PDEs in two-dimensional domains with curved boundaries or interfaces. More specifically, we will focus on the information required to proceed with the workflow of the VEM curvilinear mesh generation pipeline, see Figure 8. A more detailed description of the VEM with curved edges can be found in [5, 9].

In the remainder of the paper we will use the Laplace equation with Dirichlet boundary conditions, i.e. find $u(x, y)$ in such that

$$-\Delta u = f \ \text{ in } \Omega; \quad u = u^* \ \text{ in } \partial\Omega, \tag{1}$$

as the model problem to illustrate the main features of the VEM discretization and identify the geometrical operations required to formulate it.

We will firstly recall how to formulate a VEM discretization on a straight-sided polygonal mesh [10] and then proceed to describe how this approach can be modified to deal with curved edges.

Let E be a polygon with all straight edges, we define the space

$$V_h^k(E) := \Big\{ v \in H^1(E) \ \text{s.t.} \ v|_{\partial E} \in C^0(\partial E)\,, \quad \Delta v \in \mathbb{P}_{k-2}(E), \quad v|_e \in \mathbb{P}_k(e)\ \forall e \subset \partial E \Big\}. \tag{2}$$

where $\mathbb{P}_s(\mathcal{O})$ denotes the set of polynomials of order s on the set $\mathcal{O}$.

A function $v \in V_h^k(E)$ is uniquely determined by the following degrees of freedom, see Figure 1(a):

D1: value of v at the vertices;

D2: $k-1$ values of v on the edge nodes;

D3: $k(k+1)/2$ moments $\displaystyle\int_E v\, p_{k-2}\ \mathrm{d}E$.

Such values are the only ingredients you need to set the virtual element method, that is to create projection operators, assemble the global matrix and compute the solution [11]. Note that the function v is not known *a priori*; indeed there is no need to have the explicit expression of $v \in V_h^k(E)$. For this reason we denote v as *virtual*: it will never be computed explicitly, and it is known only via its degrees of freedom.

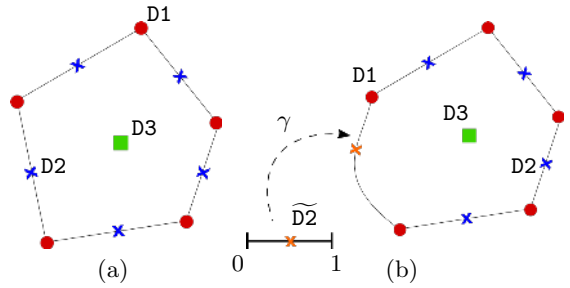


Figure 1: Degrees of freedom for the straight case (a) and for the curved case (b) with polynomial order $k = 2$. The mapping $\gamma(t)$ $(0 \le t \le 1)$ is used to define the curved edge.

Before describing how the VEM deals with curved elements, we recall the definition of the Π_k^∇ projection operator, which is an essential tool to assemble the global linear system arising from a VEM discretization. Given a function $v \in V_h^k(E)$, we define $\Pi_k^\nabla : V_h^k(E) \to \mathbb{P}_k(E)$ as

$$\begin{cases} \displaystyle\int_E \nabla\Pi_k^\nabla v \cdot \nabla p_k \ \mathrm{d}E &= \displaystyle\int_E \nabla v \cdot \nabla p_k \ \mathrm{d}E \\ \displaystyle\int_{\partial E} \Pi_k^\nabla v \ \mathrm{d}s &= \displaystyle\int_{\partial E} v \ \mathrm{d}s \end{cases} \tag{3}$$

where p_k denotes any polynomial of order k. The left-hand sides are polynomials so, if an integration quadrature rule for polygons is available, they are

computable. The right-hand side of the second equation is an edge-wise continuous polynomial since $v|_e \in \mathbb{P}_k(e)\ \forall e \in \partial E$. Such polynomials are uniquely determined by the degrees of freedom `D1` and `D2` so they are also computable. In this framework we are able to compute also the right hand side of the first equation as follows. Integrating by parts we get

$$\int_E \nabla v \cdot \nabla p_k \, \mathrm{d}E = -\int_E v \Delta p_k \, \mathrm{d}E + \int_{\partial E} (\mathbf{n} \cdot \nabla p_k)\, v \, \mathrm{d}s\,,$$

where $\mathbf{n}$ is the outward normal. The value of the bulk integral is known since it uses the internal degrees of freedom of v, i.e., `D3`. Finally, we can compute the boundary contribution. As a consequence, we observe that although v is not explicitly known, we are able to get its Π_k^∇ −projection directly form the degrees of freedom.

2.1 Extension to curved edges

To deal with polygons characterized by curved edges, the idea is to put geometry information within $V_h^k(E)$. Given a polygon E, we denote by ∂E and $\widetilde{\partial E}$ the set of straight and curved edges, respectively. Then, we define a new space

$$\begin{aligned}\mathcal{V}_h^k(E) := \Big\{ v \in H^1(E) \text{ s.t. } & v|_{\partial E \cup \widetilde{\partial E}} \in C^0(\partial E \cup \widetilde{\partial E})\,, \\ & \Delta v \in \mathbb{P}_{k-2}(E), \\ & v|_e \in \mathbb{P}_k(e)\ \forall e \subset \partial E\,, \\ & v|_{\mathfrak{e}} \in \widetilde{\mathbb{P}}_k(\mathfrak{e})\ \forall \mathfrak{e} \subset \widetilde{\partial E} \Big\}. \quad (4)\end{aligned}$$

The key point is the definition of $\widetilde{\mathbb{P}}_k(\mathfrak{e})$, as a polynomial space of degree k in the one variable parameter space of the curved edge, i.e.,

$$\widetilde{\mathbb{P}}_k(\mathfrak{e}) := \mathbb{P}_k([0,1]) \circ \gamma\,,$$

where γ is a sufficiently regular map that describes the curved edge of the polygon (see Figure 1 (b)). Then, to uniquely determine a function $v \in \mathcal{V}_h^k(E)$, we need the following degrees of freedom:

- `D1`: value at the vertices;
- `D2`: $k-1$ values on straight edges;
- $\widetilde{\texttt{D2}}$: $k-1$ values on the parameter space [0, 1] associated with the curved edge $\mathfrak{e}$;
- `D3`: $k(k+1)/2$ moments $\displaystyle\int_E v\, p_{k-2}\, \mathrm{d}E$.

Comparing the definition of the spaces $V_h^k(E)$ and $\mathcal{V}_h^k(E)$, given by equations (2) and (4), we appreciate that the curved space is an extension of the straight one. Indeed, if a polygon E does not have any curved edges, equations (2) and (4) yield identical spaces. A further proof about this fact are the degrees of freedom: these two spaces share the degrees of freedom `D1`, `D2` and `D3`, but $\mathcal{V}_h^k(E)$ has the additional degrees of freedom $\widetilde{\texttt{D2}}$ that allows the presence of polygons with curved edges.

The fact that $\mathcal{V}_h^k(E)$ is an extension of $V_h^k(E)$ also gives same benefits from a more practical point of view. Indeed, if we are able to compute integrals on polygons with curved edges and on curved edges themselves, the virtual element framework stays the same. Consider for instance the computation of the Π_k^∇ projection. If we have a quadrature rule to integrate polynomials on curved domains, the left-hand side of the first equation can be computed. Then, if we are able to integrate polynomials over curved edges, all the integrals over the boundary of E can also be computed.

As a result, the high-order VEM discretization for domains characterized by curved boundaries or interfaces requires the following geometrical information:

1. The coordinates of a set of points on the vertices and edges of the polygonal mesh which are used to interpolate the numerical solution v.
2. The coordinates of a set of quadrature points and their corresponding weights for evaluating integrals over:
   (a) curved edges, and
   (b) polygons characterized by curved edges.
3. The mapping $\gamma(t)$ defining the curved edge which is used to compute tangent and normal vectors appearing in some of the integrals.

The numerical integration over curved edges uses the standard quadrature rules on the parameter space which require the evaluation of the Jacobian of the map γ. Figure 2 depicts the location of the quadrature points on the edges of the mesh.

The integration within curved polygons follows the quadrature approach described in reference [5]. In brief and following the notation of Figure 2, a reference line (e.g. a diagonal of the element) is defined to trace perpendicular lines through the quadrature points of the edges. In each these lines, standard quadrature points are located on the segment within the edge quadrature point and the intersection point with the reference line. The process is repeated for each edge of the element and the resulting set of quadrature points is used for the approximation of the integrals. In practice the reference line is chosen so that no quadrature points may fall outside of the polygon. To reduce

the number of quadrature points whilst retaining accuracy, reference [5] proposes the use of a compression procedure. Note that the evaluation of integrals over curved polygonal cells is an area of ongoing research.

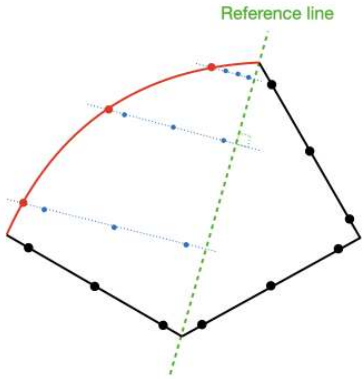


Figure 2: VEM quadrature points: The edge quadrature points are shown as large dots on the edges of the polygon. The generation of internal quadrature points is illustrated for the curved edge (in red) only. Here we define a reference line and trace perpendicular lines to it passing through the edge quadrature points. On the perpendicular lines, standard quadrature points (small dots) are located on the segment within the edge quadrature point and the intersection point with the reference line.

The following sections describe how this geometrical information is processed as part of the mesh generation procedure.

3. "A POSTERIORI" HIGH-ORDER VEM MESH GENERATION

We seek to generate meshes suitable for high-order VEM discretizations that conform to a computational domain boundary defined in terms of a standard CAD boundary representation (B-rep) [12]. We follow essentially an *a posteriori* high-order mesh generation approach where we modify a straight-sided polygonal mesh and transform it into a curvilinear mesh that conforms to the boundary. This process is illustrated in Figure 3.

The methodology aims to be completely detached from the VEM solver and to support every user-defined geometrical order. All this is made possible by extending the current capabilities of the open-source high-order mesh generator *NekMesh* [13] and its application programming interface (API) for geometrical inquiries to external CAD libraries such as Open Cascade [14] and CADfix [15]. In the following we will refer to these libraries as the CAD engine or the API.

As in the classical *a posteriori* high-order mesh generation pipelines, it is necessary to generate a valid

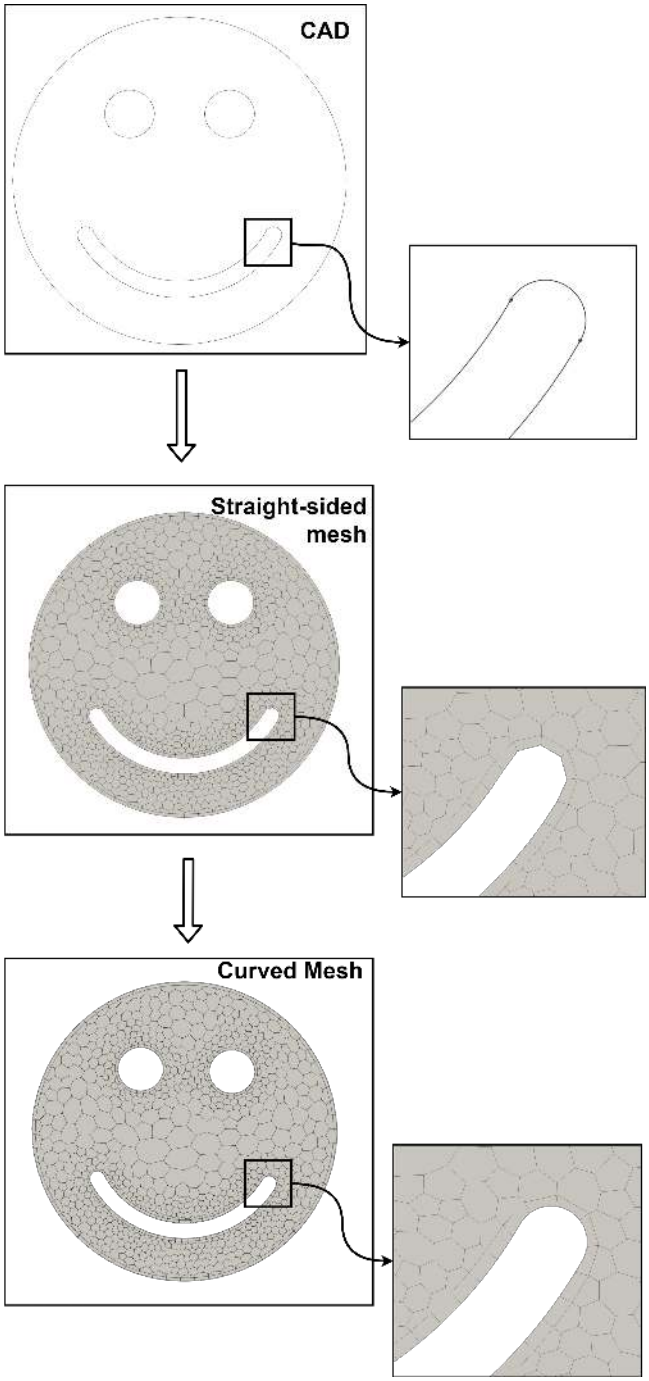


Figure 3: The *a posteriori* approach to high-order VEM mesh generation. From top to bottom: CAD B-Rep definition of the domain, straight-sided polygonal mesh, and curvilinear high-order mesh.

straight-sided mesh first, and then use high-order tools to curve the boundary and interface mesh edges whilst ensuring the new boundary-conforming elements are valid and of high quality. A graphical illustration of this process is given in Figure 3. We follow an approach where the straight-sided polytopal mesh is generated using a third-party software, in our case STAR-CCM+ [16]. Then we perform *a posteriori* connectivity identification between the linear polygonal vertices, edges and the corresponding CAD-objects. We construct and project the high-order nodes on the CAD as specified by a particular combination of quadrature rules. Finally, using these projected nodal points, we use the CAD API to retrieve all the geometrical information relevant to the VEM solver.

3.1 Generation of the straight-sided polygonal mesh

Two main strategies could be employed to generate the straight-sided polygonal mesh. First is the classical bottom-up strategy, where the vertices are inserted directly onto CAD objects (B-Splines, NURBS, etc.) inside *NekMesh*. Then one can generate seed points and perform Voronoi tesselations, including refinements, as described in reference [6]. This ensures CAD conformity and would allow direct use of the isogeometric VEM without any further mesh manipulations.

To detach the geometrical information from the numerical discretization, so that the interaction with the CAD B-rep is not handled by the VEM solver, we start the generation process from a straight-sided polytopal mesh created using a third-party software. Following the existing *NekMesh* pipeline for unstructured triangular and quadrilateral meshes, we choose STAR-CCM+[16], which has a robust commercial polyhedral mesh generator. It can read CAD information directly and can be combined with multiple fine control features, including anisotropic prism/quad layers, curvature refinement, maximum edge deviation from the CAD, vertex projection on CAD surfaces, multi-surface proximity mesh control and separate patch (curve and surface) control.

The main requirements on the user side for this step are ensuring boundary conformity of the polygonal vertices and a CAD deviation distance within the minimum edge length. Additionally, the user should ensure that the first layer of elements is thick enough to accommodate the edge projection on the CAD. In the following, and for simplicity, we will use a circular ring domain to illustrate some of these features. An example of a straight-sided polygonal mesh generated using STAR-CCM+ is depicted in Figure 4.

Once created in STAR-CCM+, the straight-sided mesh is exported to a `.ccm` file and it is read by *NekMesh* through the CCM OpenFoam importer [17]. More specific details about the implementation are given in section 3.5. This input process populates the classical *NekMesh* data structure of elements, edges and vertices. These also include topological information, for instance the connectivity between the mesh entities, and potential boundary flags set by the user in STAR-CCM+. However these do not contain the CAD information at this stage.

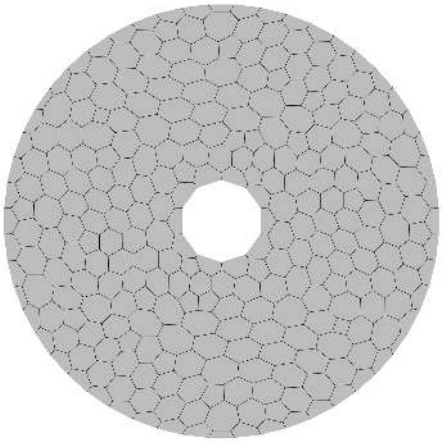


Figure 4: A coarse uniform linear mesh of a ring with interior radius $R_1 = 0.2$ and exterior radius $R_2 = 1$.

3.2 API to a CAD engine for geometrical queries

In order to use any of the high-order tools available, one needs to first obtain the CAD information and then link the boundary mesh entities with the corresponding CAD objects. *NekMesh* achieves that through an API that links to CAD engines. At this moment, *NekMesh* supports OpenCascade Community Edition(OCE) [14] and ITI CADFix [15]. This API can read geometrical objects such as points, lines, and topological information from a standard STEP file format [18] which can be created using state-of-the-art CAD software. Moreover, the API is responsible for calculating the necessary geometrical information related to the CAD for the following functions:

- mapping parametric location t on a CAD curve to its Cartesian location $\overline{x}$,
- mapping Cartesian locations $\overline{x}$ to parametric coordinates t,
- calculating the closest distance d to a CAD curve given a coordinate $\overline{x}$, and

- evaluating normal $\overline{N}$ and tangent vectors $\overline{T} = \overline{x}'(t)$ to the curve.

At this point, both the linear mesh and the CAD objects are available and we can connect the entities by shortlisting the edges, boundary vertices and boundary elements from the edge boundary flags.

Due to the tendency of STAR-CCM+ polygonal mesh generator to place vertices further away from the CAD curve, one cannot just use these boundary flags, but needs to identify the closest curve to every vertex. In order to do this, *NekMesh* creates a thin bounding box in Cartesian space around every CAD curve, within a geometrical tolerance in the range of 0.001 to 0.01 times the maximum length of the box, and stores it in a *k-D* tree data structure [19]. Then exploiting this *k-D* tree, we shortlist only several potential CAD curve candidates for every vertex. For these, we calculate the distance of the vertex to the parametric CAD curves with the help of the API. If the shortest one is further away than a small distance, this vertex and the corresponding edge are left straight to avoid mesh entanglement. Otherwise, the vertex is projected to the CAD curve with its corresponding parametric location, t.

The extension for edges is straightforward. If the two vertices belong to the same CAD curve, then this edge is clearly part of this CAD object, and it is marked as such. An important exception is when the two edge vertices have no common CAD object. This could happen in the junction between two connected CAD curves, where STAR-CCM+ inserts an edge. As will be discussed later, this rare case requires a special curving technique.

This strategy has also been used extensively for 3D element tetrahedral, hexahedral and prismatic elements. Therefore, the extension to a 3D polyhedral one is relatively straightforward, but with the API now performing geometrical queries on 3D CAD objects: curves and surfaces.

3.3 CAD projection of additional points

The main difficulty in curving the edges occurs when the two vertices of an edge lie on the same CAD object. Here *NekMesh* employs the CAD curve and the API to parametrically create the high-order edge-nodes according to a quadrature rule, defined on the reference segment $[-1, 1]$, which is typically a form of Gaussian quadrature. To evaluate their positions on the curve, we utilise the mapping $\gamma(t)$ which defines the edge in the region $0 \leq t_1 \leq t \leq t_2 \leq 1$. We therefore construct a mapping between the intervals $[-1, 1]$ and $[t_1, t_2]$, then apply $\gamma(t)$ to locate the points in Cartesian space along with their parametric coordinates t_j. Finally, we calculate the distance, d_j, between the quadrature points on the straight edge and the corresponding one on the curve. In an unlikely scenario that the distance d_j is larger than the edge length, a projection error could have occurred in the CAD engine, and therefore, the edge is linearised. This process ensures exact parametric projection to the CAD curves, mesh boundary conformity and easy access with the API to the necessary geometrical information for output.

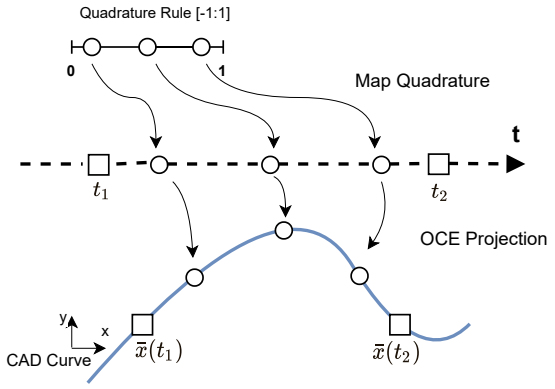


Figure 5: Parametric projection for an edge with a single CAD curve.

A rare exception to this process happens when STAR-CCM+ generates an element that spans two CAD objects. In this case, the edge is located across two CAD curves, so the previous approach cannot be applied. Therefore, we generate the quadrature points on the straight edge first. Then we project the edge node j to the closest cartesian location of the two CAD objects. Note that this introduces a small error in the location of the quadrature points but, if the curves are smooth, it has a negligible effect on the solution accuracy. A schematic of the two processes can be seen in Figure 6.

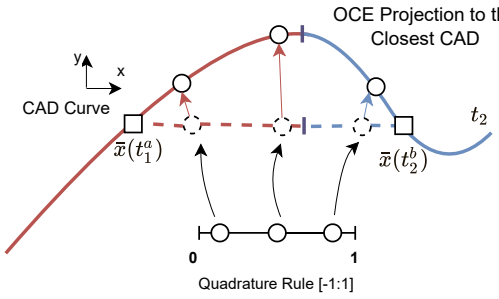


Figure 6: Projection for a straight edge with 2 CAD curves.

The projection of points employs the geometrical procedures available in the API. These procedures rely upon third-party implementations such as OpenCascade which are reasonably robust in general. However, their applicability may be limited in some instances such as, for instance, when the curve edges exhibit inflections or very rapid changes in curvature.

Following these steps, *NekMesh* can produce curved meshes with arbitrary user-defined order. Moreover, the mesh generator is completely detached from the VEM solver, it is based on the user's requirements, and supports any combinations of classical 1D quadrature rules such as Gauss, Gauss-Lobbato, Gauss-Radau, etc. However, the construction of the quadrature points within the polygonal element is left to the solver side, due to the variety of techniques adopted by the different VEM solvers.

3.4 Ensuring mesh validity

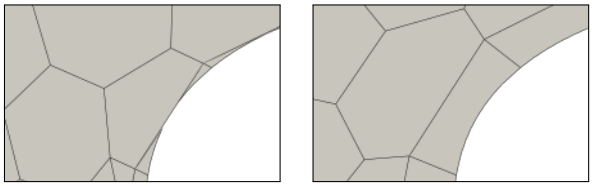
In regions with high curvature, it is possible to generate tangled polygonal elements after the edge projection step, where the addition of curvature causes the element to self-intersect. Therefore, in the legacy *NekMesh* pipeline with *standard* element shapes such as triangles and quadrilaterals, we calculate the distortion of each element using the Jacobian of the mapping from the standard reference space [20]. A negative value indicates an invalid tangled element and hence this element needs to be either linearized or refined. This approach is not easily applicable to arbitrary polygons in the VEM due to the difficulty in defining such mapping. In 2D domains a visual inspection of the mesh often helps identifying invalid elements, however this is not feasible in 3D. One can devise a method to detect the presence of invalid elements by calculating the signed area of a polygon as an integral over its boundary. If the area of the computational domain, its boundary viewed as a polygon, differs from the sum of the areas of the polygonal elements calculated in the same fashion, then there are tangled elements in the mesh. These elements can then be identified (in 2D) by calculating the winding number of the polygon, which will be different from zero if self-intersection occurs.

An alternative method for imposing mesh validity is to construct a single sufficiently thick boundary layer in the regions of concern which accommodates the curving of the mesh effected by the CAD projection. It has been shown in the literature [3] that this minimum thickness $\delta_{\min}$ for a quadrilateral element can be found using the relationship

$$\frac{\delta_{\min}}{R} \geq \frac{c^2}{8R^2} \tag{5}$$

where R is the radius of curvature and c the length of the straight-sided edge. This value can also be used as a conservative estimate of the mesh size required to ensure validity for convex polygonal elements with four edges or more. Figure 7 illustrates the application of this criterion in the case of a mesh with a layer of quadrilateral cells near the boundary with values of δ above and below $\delta_{\min}$, with the later leading to an invalid mesh.

It is worth noting that techniques currently employed in high-order meshing for deforming a straight-sided mesh to accommodate boundary curvature, see for instance [7, 8], could also be implemented using a VEM formulation and applied in this context. However, such VEM implementation is left for future work.



(a) (b)

Figure 7: A mesh with a layer of stretched quadrilaterals: (a) A layer thickness below the value $\delta_{\min}$ given by equation (5) leads to self-intersection and thus invalid elements; (b) A valid mesh is obtained with a value above the minimum thickness.

3.5 Implementation

The various steps of the mesh generation method described in previous sections have been implemented within the open-source code *NekMesh*. A schematic flowchart of the implementation is presented in Figure 8.

Note that the implementation performs all the required geometrical queries via the API to the CAD engine, and that the communication with the VEM solver is via a simple file-based interface system. The solver is asked to read the information defining the linear mesh (`.mesh`) and the high-order geometrical information on the curved edges (`.nmg`).

The `.mesh` file includes only information about the linear mesh: the vertices and their cartesian coordinates, the edges with the IDs of the two vertices, and an additional flag showing the corresponding boundary condition. Finally, for the polygonal elements, we first indicate the number of vertices of the polygon and their IDs.

The `.nmg` file, on the other hand, does not have any connectivity details or elements. Instead, it communicates only a minimal amount of geometrical information about the previously populated curved edges.

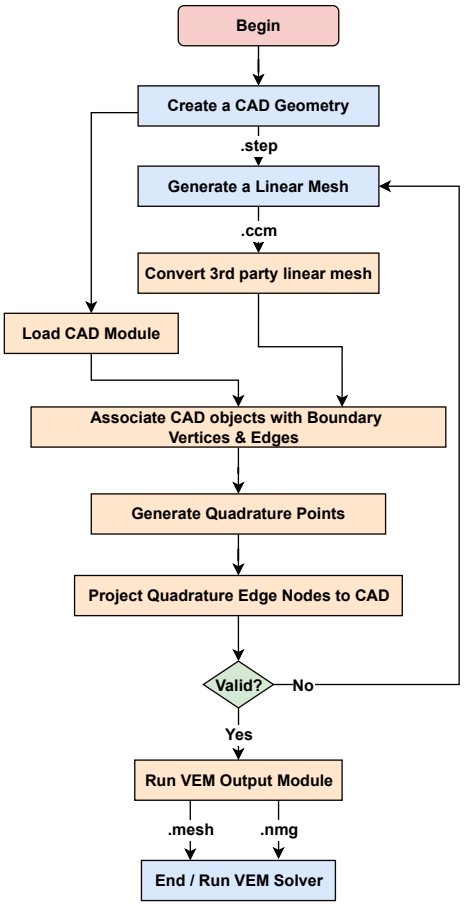


Figure 8: The workflow of the proposed pipeline form the CAD definition to the curvilinear high-order mesh.

NekMesh first provides the ID of the curved edge and the data from the `.mesh` file. Then, for every quadrature point $\overline{x}_j$, determines and writes to the file the following geometrical information with the help of the CAD engine:

- location inside the standard element: $0 \leq t_j \leq 1$,
- Cartesian location: $\overline{x}$,
- unit normal to the CAD curve: $\overline{N}$ (pointing inside the curvature), and
- tangent to the CAD curve: $\overline{x}'(t)$.

Considering the test ring geometry from Figure 12 and the VEM solver by [9] at order $k = 2$, *NekMesh* constructs the combination of quadrature rules automatically as required by the solver: a Gauss-Lobatto rule with $n = 3$, and Gauss rules with $n = 2, 3$. The geometrical information evaluated on the inner circle from the `.nmg` file is displayed in Figure 9.

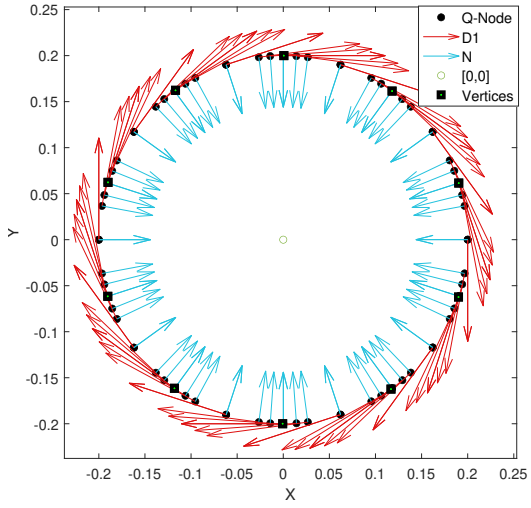


Figure 9: Visualization of the high-order geometrical information (with $k = 2$) communicated to the VEM solver via the `.nmg` file.

Our choice of using a simple file-based interface system is informed by the need to provide state-of-the-art VEM solvers with easy and robust access to the high-order curvilinear information without the need to interact with the B-rep of the computational domain.

4. VERIFICATION AND EXAMPLE OF APPLICATION

4.1 VEM Verification

This section aims at illustrating the validity of the meshes generated by the proposed methodology. It will show that the VEM discretizations of the Laplace equation in a simple domain, a circular ring, achieve the expected rate of convergence in both straight-sided and curvilinear meshes.

From the numerical analysis approximate curved boundaries may corrupt the numerical approximation of the discrete solution. Indeed, the error in a numerical solution, ϵ, can be split in two main contributions:

$$\epsilon = \epsilon_f + \epsilon_g \,,$$

where the error ϵ_f arises due to the discretization of the functional spaces and the involved (bilinear) forms, and the ϵ_g represents the error on the solution that stems from the approximation of the geometry of the computational domain.

When we are dealing with domains with straight boundaries the contribution of ϵ_g is negligible since a piece-wise straight segments perfectly match a straight

boundary. Therefore the error of a numerical solution is *only* ϵ_f due to the discrete functional spaces used.

However, when the domain is curved and we approximate curved boundaries with piecewise straight segments $\epsilon_g \approx h^2$ where h is the size of the mesh [9]. As a consequence, improving the approximation provided by the discrete functional spaces may not decrease the whole numerical error we have.

In the following we will perform numerical experiments to verify these claims. Moreover, we will see that the high-order meshes generated by *NekMesh* combined with the curved virtual element spaces reviewed in Section 2 overcome this issue.

Let Ω denote a domain consisting of a circle of radius 1 centred at the origin, with a circular hole removed at the origin with radius 0.2. We define the right-hand side and the boundary conditions in such a way that the solution of a Laplacian problem on Ω is the function

$$u(x,\, y) = \log(x^2 + y^2)\,.$$

We discretize the computational domain Ω in two ways: one with straight-sided edges, referred to as `noGeo`, and one with curved edges, denoted `withGeo`. For each of these mesh types, we construct a sequence of four meshes with decreasing mesh size. Figure 14 shows an example of a `withGeo` mesh.

The VEM approach reviewed in Section 2 with $k = 2$ is then used to solve this problem. For each of these meshes we compute the errors in the L^2 norm and the H^1 semi-norm.

The trend of these error is depicted in Figure 11. We begin by analysing the error in the L^2 norm. For the straight-sided case we know that

$$\epsilon_f \approx h^3 \quad \text{and} \quad \epsilon_g \approx h^2\,.$$

The geometrical error is dominant, and so we observe a decay of order two of the error. However, when we consider curved meshes using the exact geometry, the error in approximating the geometry decreases as $h \to 0$. For instance, assuming regularity of the domain, a second-order curved edge approximation leads to an error $\varepsilon_g = O(h^3)$. Furthermore we have $\epsilon_f \approx h^3$ and, as a consequence, the trend of the error is not affected by the geometrical approximation error and we observe an error decay of order 3.

In the case of the H^1 semi-norm error, we obtain the expected error decay of order 2 for both type of meshes, but the absolute value of the error computed with the curved mesh is smaller. Also the better convergence trend observed using the `withGeo` meshes is due to the better approximation of the geometry. Indeed, the total error in the `noGeo` mesh have two contributions with the same order in the H^1 norm, namely

$$\epsilon_f \approx h^2 \quad \text{and} \quad \epsilon_g \approx h^2\,.$$

Consequently the error trend is not corrupted but its value is affected by two contributions. While in the `withGeo` approach the error due to the geometry is null, the final error is *only* influenced by ϵ_f and, as a consequence, it is smaller.

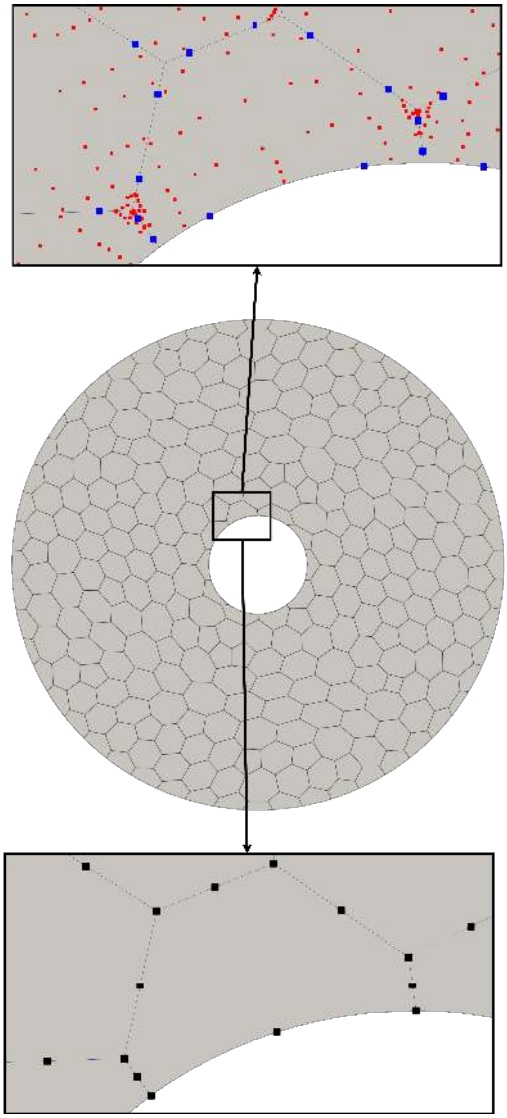


Figure 10: A curvilinear high-order polygonal mesh of the domain Ω. The mesh of the domain only displays the edges of the polygonal elements. An enlargement of the mesh near the boundary illustrates the additional degrees-of-freedom required for the high-order discretization. The top window shows the locations of the quadrature points on the edges (blue) and on the interior (red) of the polygonal cells. The bottom window shows the locations of the points used for interpolation.

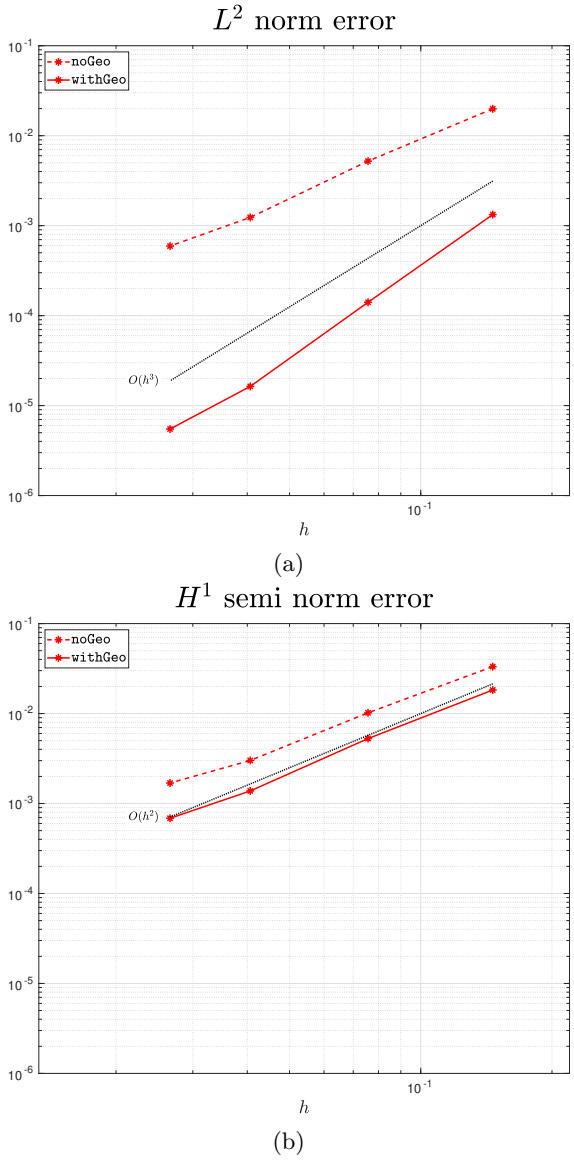


Figure 11: Mesh convergence of the high-order VEM discretization: (a) error in the L^2 norm; and (b) error in the H^1 semi-norm.

4.2 A Practical 2D Geometry

This section illustrates the application of the high-order mesh generation procedure to a computational domain for an automotive aerofoil geometry exhibiting variable curvature along its length. The geometry of the aerofoil is defined by four NURBS curves.

The linear polygonal mesh generated by STAR-CCM+ is depicted in Figure 13. The mesh resolution has been purposely increased in the regions near the leading and trailing edges of the aerofoil.

Figure 14 shows the the edges of the polygonal mesh

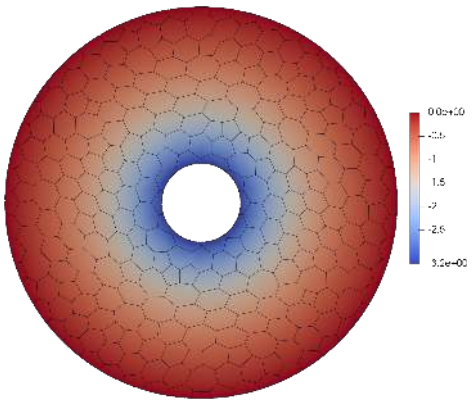


Figure 12: High-order VEM solution ($k = 2$) of the Laplace equation with Dirichlet boundary conditions computed on a curvilinear mesh (`withGeo`).

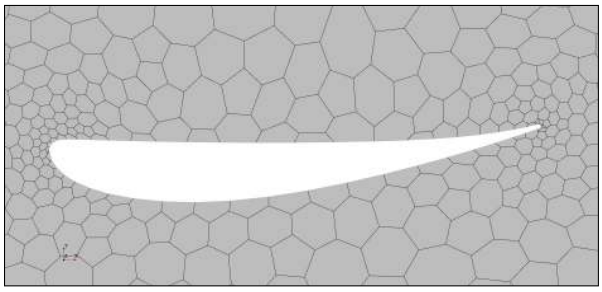


Figure 13: A straight-sided linear mesh of an automotive aerofoil.

together with two enlargements showing the location of the interpolation and quadrature points used in the high-order discretization.

5. CONCLUSIONS AND FURTHER WORK

We have proposed a proof-of-concept for a polygonal high-order curvilinear mesh generator for the Virtual Element Method (VEM) for arbitrary geometries defined through a standard CAD B-rep of the domain. The realisation is that the interpolation and integration of functions within a VEM discretization only requires the definition of a set of boundary, interface and internal points. This allows us to interpret the problem of finding the location of these interpolation and integration points as geometrical inquiries to a B-rep of the computational domain, performed via an interface to CAD libraries.

As a consequence, we can adopt an *a posteriori* ap-

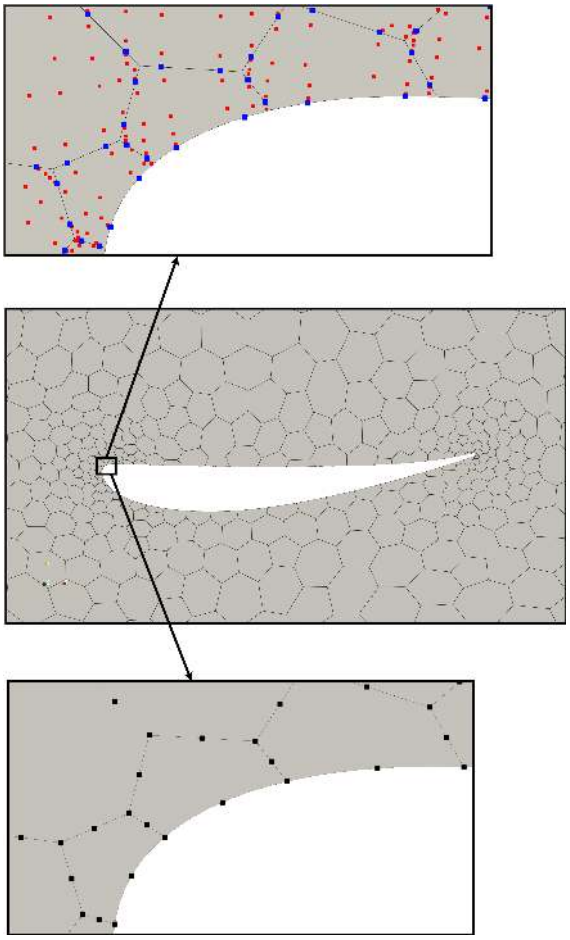


Figure 14: A curvilinear high-order polygonal mesh of an automotive aerofoil. The mesh of the domain only displays the edges of the polygonal elements. An enlargement of the mesh near the boundary illustrates the additional degrees-of-freedom required for the high-order discretization. The top window shows the locations of the quadrature points on the edges (blue) and on the interior (red) of the polygonal cells. The bottom window shows the locations of the points used for interpolation.

proach to high-order curvilinear mesh generation for the VEM. The starting point of the process is the generation of a straight-sided mesh in STAR-CCM+. The next step is to use the CAD API implemented in *NekMesh* for reconstructing the CAD information from STEP files and, according to the user-defined order and quadrature rules, parametrically projecting the quadrature nodes on the curved geometry. Finally, *NekMesh* communicates with the VEM solver through a two-file interface system: one for the linear mesh with its connectivity and a second for the geometrical information at the quadrature nodes. Using an exact solution of the Laplace equation on a ring geometry we show that the generated high-order curvilinear meshes for the VEM are valid, accurately reproduce the analytical solution, and converge at the expected rate as the mesh size is decreased.

Although we have discussed strategies for assessing the validity of the mesh, we have made no attempt to evaluate mesh quality. This is a topic that has received little attention in the VEM literature until recently [21, 22], but it is of significant importance for VEM-based simulation and adaptation [23] and thus deserves further investigation. Most of the current mesh quality criteria for high-order meshing rely on the evaluation of the Jacobian of a mapping from a reference element, usually a regular polytope, as a measure of its deformation. Even tough such mappings have been proposed for convex straight-sided polygons, e.g. using barycentric coordinates [24], such mappings are, to the best of our knowledge, not available for non-convex curvilinear polytopes.

We believe that this work has laid a strong foundation for the extension of the methodology to the generation of curvilinear polyhedral meshes. However, devising quadrature rules suitable for curvilinear polyhedral is an area of current active development. This represents a technical bottleneck that must be addressed before we can extend the proposed methodology to 3D.

6. ACKNOWLEDGEMENTS

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 955923. David Moxey, Joaquim Peiró and Mashy Green acknowledge funding from EPSRC under grant EP/R029423/1.

References

[1] Versteeg H., Malalasekera W. *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*. Pearson, second edn., 2007

[2] Di Pietro D.A., Droniou J. *The Hybrid High-Order Method for Polytopal Meshes*. Springer, 2020

[3] Moxey D., Green M., Sherwin S., Peiró J. "An isoparametric approach to high-order curvilinear boundary-layer meshing." *Computer Methods in Applied Mechanics and Engineering*, vol. 283, 636–650, 2015

[4] Cottrell J.A., Hughes T.J.R., Bazilevs Y. *Isogeometric Analysis: Toward Integration of CAD and FEA*. Wiley, 2009

[5] Beirão da Veiga L., Russo A., Vacca G. "The virtual element method with curved edges." *ESAIM:*

Mathematical Modelling and Numerical Analysis, vol. 53, no. 2, 375–404, 2019

[6] Ferguson J.A., Kópházi J., Eaton M.D. "NURBS Enhanced Virtual Element Methods for the Spatial Discretization of the Multigroup Neutron Diffusion Equation on Curvilinear Polygonal Meshes." *Journal of Computational and Theoretical Transport*, vol. 51, no. 4, 145–204, 2022

[7] Turner M., Peiró J., Moxey D. "Curvilinear Mesh Generation Using a Variational Framework." *Computer-Aided Design*, vol. 103, 73–91, 2018

[8] Ruiz-Gironés E., Roca X., Sarrate J. "High-order mesh curving by distortion minimization with boundary nodes free to slide on a 3D CAD representation." *CAD Computer Aided Design*, vol. 72, 52–64, 2016

[9] Dassi F., Fumagalli A., Mazzieri I., Scotti A., Vacca G. "A Virtual Element Method for the Wave Equation on Curved Edges in Two Dimensions." *Journal of Scientific Computing*, vol. 90, no. 50, 2022

[10] 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.*, vol. 23, no. 1, 199–214, 2013

[11] Beirão da Veiga L., Brezzi F., Marini L.D., Russo A. "The Hitchhiker's Guide to the Virtual Element Method." *Math. Models Methods Appl. Sci.*, vol. 24, no. 08, 1541–1573, 2014

[12] Stroud I. *Boundary Representation Modelling Techniques*. Springer, 2006

[13] "NekMesh: An open-source high-order mesh generator.", Last accessed October 2022. URL `https://www.nektar.info/`

[14] Capgemini Engineering. "Open Cascade.", Last accessed October 2022. URL `https://www.opencascade.com`

[15] ITI Global. "CADfix.", Last accessed October 2022. URL `https://www.iti-global.com/cadfix`

[16] Siemens. "STAR-CCM+.", Last accessed October 2022. URL `https://www.plm.automation.siemens.com/global/en/products/simcenter/STAR-CCM.html`

[17] OpenFOAM. "API Guide: ccmToFoam.C File Reference.", Last accessed October 2022. URL `https://www.openfoam.com/documentation/guides/latest/api/ccmToFoam_8C.html`

[18] ISO. *ISO 10303-21:2016 Industrial automation systems and integration – Product data representation and exchange – Part 21: Implementation methods: Clear text encoding of the exchange structure*. International Organization for Standardization, 2016. The STEP standard

[19] Bentley J.L. "Multidimensional binary search trees used for associative searching." *Communications of the ACM*, vol. 18, no. 9, 509–517, 1975

[20] Karniadakis G.E., Sherwin S. *Spectral/hp element methods for computational fluid dynamics*. Oxford University Press, second edn., 2013

[21] Berrone S., D'Auria A. "A new quality preserving polygonal mesh refinement algorithm for Polygonal Element Methods." *Finite Elements in Analysis and Design*, vol. 207, no. 103770, 2022

[22] Sorgente T., Biasotti S., Manzini G., Spagnuolo M. "The role of mesh quality and mesh quality indicatorsin the virtual element method." *Adv. Comput. Math.*, vol. 48, no. 3, 2022

[23] Antonietti P.F., Berrone S., Borio A., D'Auria A., Verani M., Weisser S. "Anisotropic a posteriori error estimate for the virtual element method." *IMA Journal of Numerical Analysis*, vol. 42, no. 2, 1273–1312, 2021

[24] Budninskiy M., Liu B., Tong Y., Desbrun M. "Power Coordinates: A Geometric Construction of Barycentric Coordinates on Convex Polytopes." *ACM Trans. Graph.*, vol. 35, no. 6, dec 2016