A Graphical Framework to Study the Correlation between Geometric Design and Simulation

D. Cabiddu[†1] ![ID], G. Patané[1] ![ID], and M. Spagnuolo[1] ![ID]
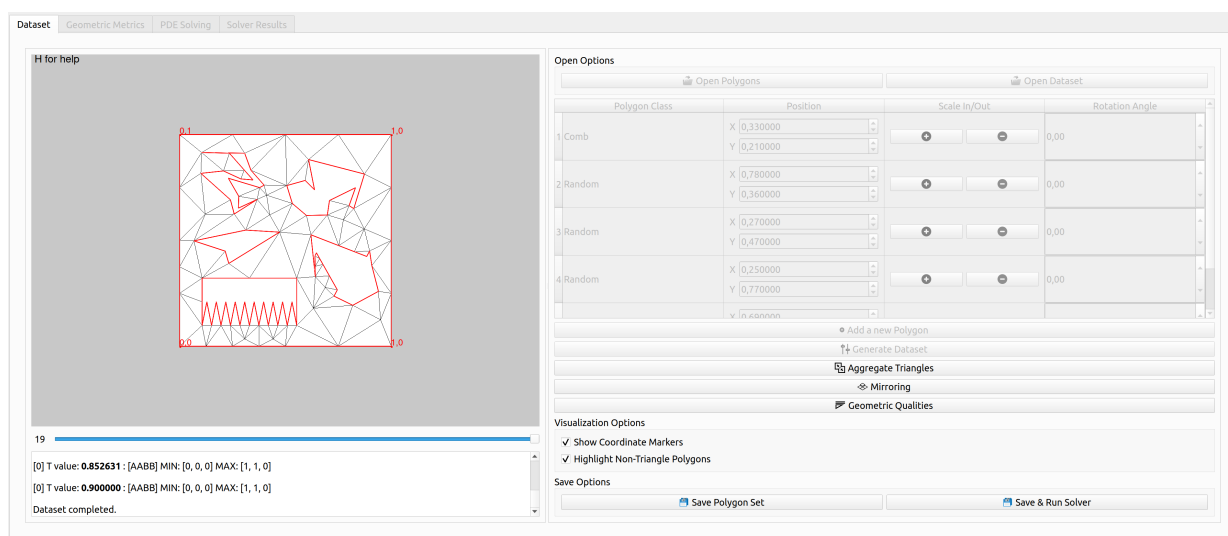
[1]CNR-IMATI, Genova, Italy

Figure 1: *Main window. On the left, a canvas allows visualizing the input mesh(es) and the bottom text box reports some logs. On the right, an advanced panel allows setting multiple parameters to both guide the dataset generation according to desired geometric features and start the geometric analysis. Also, a button is provided to quickly start the PEM solver and run the simulation.*

Abstract

Partial differential equations can be solved on general polygonal and polyhedral meshes, through Polytopal Element Methods *(PEMs). Unfortunately, the relation between geometry and analysis is still unknown and subject to ongoing research to identify weaker shape-regularity criteria under which PEMs can reliably work. We propose a graphical framework to support the analysis of the relation between the geometric properties of polygonal meshes and the numerical performances of PEM solvers. Our framework, namely* PEMesh, *allows the design of polygonal meshes that increasingly stress some geometric properties, by exploiting any external PEM solver, and supports the study of the correlation between the performances of such a solver and the geometric properties of the input mesh. Furthermore, it is highly modular, customisable, easy to use, and provides the possibility to export analysis results both as numerical values and graphical plots. The framework has a potential practical impact on ongoing and future research activities related to PEM methods, polygonal mesh generation and processing.*

CCS Concepts
*• **Software and its engineering** → Open source model; • **Computing methodologies** → Mesh geometry models; • **Mathematics of computing** → Partial differential equations;*

---

† Corresponding Author

1. Introduction

Over the last fifty years, computer-based simulation has dramatically increased its impact on research, design, and production, and is now an indispensable tool for development and innovation in science and technology. In particular, Partial Differential Equations (PDEs) offer a broad and flexible framework for modelling and analyzing several phenomena arising in fields as diverse as physics, engineering, biology, and medicine. Computer-based simulation of PDEs also relies on a suitable description of geometrical entities, such as the computational domain and its properties. However, the representation of geometric entities has been studied mainly in the field of geometric modelling, and often the requirements of shape design are different from those of numerical simulation.

In this context, *Polytopal Element Methods* (PEMs) allow solving differential equations on general polygonal and polyhedral meshes, thus offering great freedom in the definition of mesh generation algorithms. Similarly to *Finite Elements Methods* (FEMs), the performance of PEMs (i.e., accuracy, stability, and effectiveness of preconditioning) depends on the quality of the underlying mesh. Differently from FEMs, where the relation between the geometric properties of the mesh and the performances of the solver is well known [She02; Cia02; BKK08], the definition of the quality of polytopal elements is still subject to ongoing research [Lip13; MWW15; ABB*21].

The proposed graphical framework is intended to support the analysis of the relation between the geometric properties of the mesh and the numerical performances of the solver, in terms of basis degree, conditioning number of the stiffness matrix, etc. To this end, our work covers several aspects, such as the design and generation of meshes to increasingly stress geometric properties, the study of the performances of PEM solvers, and the correlation between such performances and the main geometric properties of the input meshes. Each step is performed by exploiting existing tools mainly coming from two related but independent research areas: geometric design and numerical methods for PEMs. Indeed, these tools rely on different representations of the same domain, and researchers are often required to be skilled programmers and expert tool users to allow such tools to be part of the same experimental pipeline.

We introduce `PEMesh` (Fig. 1), as an open-source software tool designed to help researchers to perform experiments on the analysis and design of polytopal meshes for PEM solvers. It is an advanced graphical tool that seamlessly integrates geometric design pipelines and PEM simulations. Specifically, it supports the design and generation of complex input polygonal meshes by stressing geometric properties while providing the possibility to solve PEMs on the generated meshes. Furthermore, `PEMesh` allows the user to correlate one or more geometric properties of the input polytopal mesh with the performances of PEM solvers, and to visualize the results through customisable and interactive plots.

The proposed tool is highly modular and customisable. It allows researchers to simulate any PEM solver, by simply calling the PEM solver executable from an internal command line and providing possibly additional input parameters other than the geometric data set. To the best of our knowledge, our proposal is the first graphical tool to generate complex discrete polytopal meshes and to support

a study of the correlation between their geometric properties and numerical PEMs solvers. Indeed, it has a potential practical impact on research activities on this subject.

The paper is organised as follows. We briefly review previous work on PEMs and existing tools both to perform PEM simulations and to design geometric data sets (Sect. 2). Then, we describe the structure of our framework and its capabilities (Sect. 3), with technical details about its implementation (Sect. 4). Finally, we discuss some directions for future research (Sect. 5).

2. Background and related work

We briefly review previous work on numerical FEM solvers and libraries (Sect. 2.1) and meshing tools (Sect. 2.2).

2.1. Numerical FEM solvers and libraries

Main PEMs include Mimetic Finite Differences [BLM14; BLS05], Discontinuous Galerkin-Finite Element Method (DG-FEM) [ACC*16; CGH14], Hybridisable and Hybrid High-Order Methods [CDG08; DE15], Weak Galerkin Method [WY13], BEM-based FEM [RW13], Poly-Spline FEM [SDG*19], and Polygonal FEM [ST04]. Main existing tools for the numerical solution of PDEs include (i) *VEMLab* [OAH*20], which is an open-source MATLAB library for the virtual element method and (ii) *Veamy*, which is a free and open source C++ library that implements the virtual element method (C++ version of [OAH*20]). The current release of this library allows the solution of 2D linear elastostatic problems and the 2D Poisson problem [OAH*19]. Other libraries are (iii) the 50-lines MATLAB implementation of the lowest order virtual element method for the two-dimensional Poisson problem on general polygonal meshes [Sut17], and (iv) the MATLAB implementation of the lowest order Virtual Element Method (VEM) [Mas18].

As a matter of example, we demonstrate how a PEM solver can be integrated into our framework. Our use case exploits the *Virtual Element Method* (VEM) [BBC*13], which can be considered as an extension to FEM for handling general polytopal meshes.

2.2. Mesh generation tools

Nowadays, meshes are commonplace in several applications ranging from engineering to bio-medicine and geology. Depending on the application field, automatic mesh generation may be a difficult task, due to specific geometric requirements to be satisfied.

Concerning simulation with FEMs, the principle behind meshing algorithms in commercial FEM solvers is described in [OK18], and an open-source tool is provided. Free-FEM [Hec12] is a popular 2D and 3D partial differential equations (PDE) solver used by thousands of researchers across the world, including its mesh generation module. Although it provides plenty of functionalities, it is based on its language and it has no graphical interface. Similarly, the MATLAB® suite provides its own FEM mesh generator [Oro20]. Both solutions focus on FEM requirements and enable the possibility to generate triangle meshes, but they do not allow the generation of generic polygon meshes.

Concerning PEM methods, available Voronoi-based meshing tools (e.g. [DFG99; TPPM12]) are not suited for our study, because they produce convex elements that are not challenging enough to stress PEM solvers. Recent works [HSL*21; SBMS22] propose ad-hoc polyhedral mesh datasets, specifically designed for their experimental phase. The dataset generation procedures are well described, but they are not easily modifiable to fit analyses other than the proposed approach. To the best of our knowledge, the benchmark proposed in [ABB*21] is the only one providing a polygonal mesh generation approach specifically designed for PDE solvers. Unfortunately, the proposed approach is not easily customisable and allows the generation of polygon meshes having a single non-triangle element.

Our framework provides an advanced mesh generation module which enables the creation of generic polygonal meshes specifically designed for PDE solvers.

3. Proposed framework

Our framework is aimed at evaluating the dependence of the performances of a PEM solver on the geometrical properties of the input polygonal mesh, which is either generated by using the framework itself or provided as an external resource. Mainly, the framework is composed mainly of four modules, each of which is provided as a specialized window.

Polygon mesh generation & loading allows the user to load one or more existing meshes or generate a new one from scratch by exploiting a set of provided polytopal elements or providing an external one. The generation of new meshes is highly customisable, and the user is allowed to play with a large set of options and parameters (Sect. 3.1).

Geometric analysis allows the user to perform a deep analysis of the geometric properties of the input polygonal meshes and to correlate each of them with the others. Results of such an analysis are shown through advanced plots (Sect. 3.2).

PEM solver allows the user to run a PEM solver and analyse its performances on input polygonal meshes. Any PEM solver may be exploited, as long as it can be run from the command line and provides its output according to a specific textual format. Both the solution and the ground-through of the PEM are shown directly on the meshes, while the performances of the solver are visualized through linear plots (Sect. 3.3).

Correlation visualization supports the analysis of the correlation between the geometric properties of the polygonal meshes and the numerical performances of the selected PEM solver. Results are made available in the form of customizable scatter plots (Sect. 3.4).

The framework provides the possibility to show the results of each analysis step on the display, customize visualization aspects of plots (i.e., colour, font sizes, etc.) and interactively analyze them by clicking on visualised points and lines to show data values in the selected point. Furthermore, such results can be saved on disk as images and as textual files, to be possibly re-used by other applications.

3.1. Polygonal mesh generation & loading

This module is started as soon as the application is run (Fig. 1). It provides the possibility to load an existing polygonal mesh or generate a new one from scratch. The mesh generation method takes a cue from the approach described in [ABB*21], where the domain is supposed to be a squared canvas, and the area of the domain which is not covered by a polygon is filled with triangles using [She96].

Differently from [ABB*21], our framework supports the generation of meshes with more than one non-triangle polygon, whose position, scale and rotation are chosen by the user before applying the triangulation of the external domain (Fig. 2). Furthermore, triangulation parameters are set according to the user needs (e.g., fixing the area of the triangles or the minimum angle) and the mirroring approach proposed in [ABB*21] can be eventually applied after the triangulation.
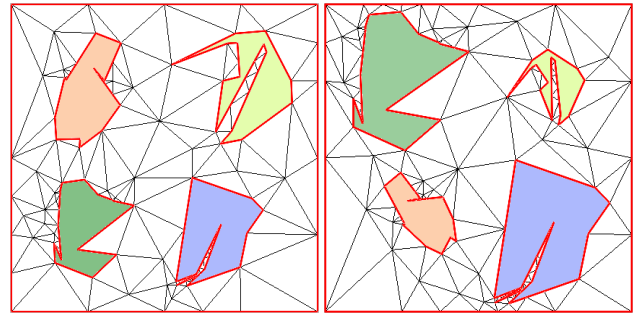

Figure 2: *Meshes generated by selecting the same set of polygons, but editing each element differently.*

An additional feature is the aggregation of the generated triangles to create generic polygons (Fig. 3). This feature allows the generation of generic polygonal meshes where the number of triangles is reduced almost to zero and some geometric properties are stressed all over the discretised domain. The aggregation criterion guarantees that the diameter of the polygons generated by aggregation is at most equal to the diameter of the smallest user-selected polygon.

3.2. Geometric analysis

When one or more polygonal meshes are available, either generated from scratch or loaded from disk, our framework allows deep geometric analysis and provides a visual summary of geometric properties. Specifically, our approach considers a set of polygonal metrics (Table 1, Fig. 4), also considering their minimum, maximum and average values.

Our polygonal metrics are classified into 6 main classes:

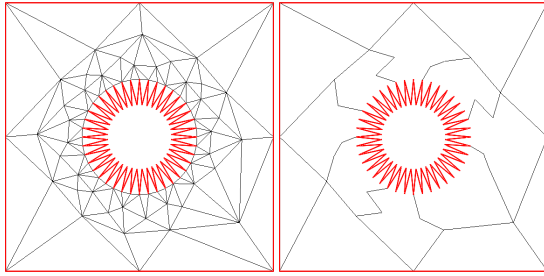- *edges*: number of edges (nE) of the input polygon, shortest edge (SE);

Figure 3: *Mesh with aggregated triangles.*

Table 1: *Proposed polygonal metrics. For scale invariant measures, the fourth column indicates whether optimal values are at the top (↑) or bottom (↓) of the definition range. Polygon measures: inscribed circle (IC), circumscribed circle (CC), polygon area (AR), kernel area (KE), minimum angle (MA), shortest edge length (SE), and minimum point-to-point distance (MPD).*

Metric	Abbr.	Range		Scale inv.
♯ Edges	nE	$(1, +\infty)$	–	Yes
Inscribed radius	IC	$(0, \infty)$	–	No
Circumscr. radius	CC	$(0, \infty)$	–	No
Circle ratio	CR	$[0, 1]$	↑	Yes
Area	AR	$[0, \infty)$	–	No
Kernel-area	KE	$[0, \infty)$	–	No
Kernal-area ratio	KAR	$[0, 1]$	↑	Yes
Perimeter-area ratio	PAR	$(0, \infty)$	↑	Yes
Min. angle	MA	$(0, \pi)$	↑	Yes
Max. angle	mA	$(0, \pi)$	↑	Yes
Shortest edge	SE	$(0, \infty)$	–	No
Edge ratio	ER	$(0, 1]$	↑	Yes
Min p2p distance	MPD	$(0, \infty)$	–	No
Normal. point dist.	NPD	$(0, 1]$	↑	Yes
Shape regularity	SRG	$(0, 1]$	↑	Yes

- *angles*: ratio MA/mA, with MA, mA minimum, and maximum inner angle of the polygon, respectively;
- *areas*: area (AR) of the polygon, kernel area (KE), kernel-area ratio (i.e., the ratio between the area of the kernel of the polygon and its whole area), area-perimeter ratio (APR);
- *radii*: inscribed circle radius (IC), circumscribed circle radius (CC), circle ratio (CR:=IC/CC);
- *distances*: minimum point to point distance (MPD), normalized point distance (NPD) (i.e., normalized version of MPD);
- *shape regularity* (SRG), as the ratio between the radius of the circle to the circumscribed polygon and the radius of the circle inscribed in the kernel of the element.

Three different visualizations of the results of the geometric analysis are available, thus enabling the possibility to either analyze each mesh as a standalone object or to consider each mesh as a part of a full data set (when more polygonal meshes are available).

The former visualization focuses on the polygonal mesh as a standalone object (Fig. 5). On each available mesh, it visually highlights the element (either triangle or polygon) where the minimum and the maximum values are located, together with textual infor-
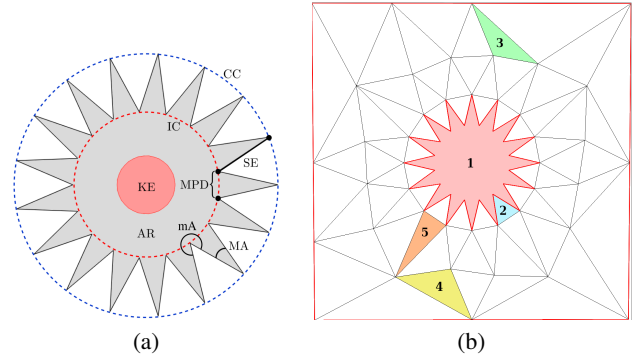

Figure 4: *(a) Geometric metrics of a polygon. (b) Minimum geometric metrics on a polygonal mesh: #1 satisfies min(MPD) and min(KAR); #2 satisfies min(IC), min(CC), min(AR), min(KE), min(MPD), min(SE); #3 satisfies min(CR); #4 satisfies min(KAR); #5 satisfies min(mA).*
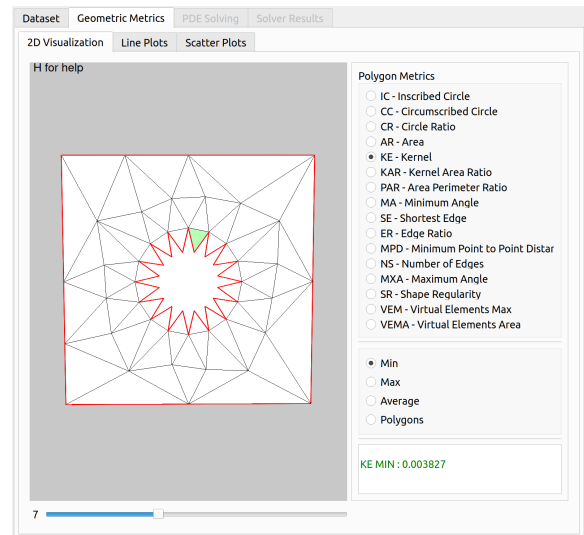

Figure 5: *Graphical visualization of geometric polygonal metrics. Given a polygonal mesh and a selected geometric metric, the mesh element (either a triangle or a non-triangle polygon) satisfying the metric is highlighted in the canvas. Numerical values are reported in the bottom text box.*

mation about them (e.g., the numerical values). This visualization enables the detection of possible geometric degeneracies.

The other two visualization approaches consider the full data set as a single object and plot how the evaluated geometric metrics evolve in the data set, and how they correlate with each other. Specifically, a former window (Fig. 6) shows the variation of the minima and maxima of each geometric metric. Each one of these linear plots corresponds to a single geometric metric, and it is generated by setting its *x*-axis to the indices of the meshes in the data set and its *y*-axis to the minimum and the maximum values. The second window is specialized in correlating two user-defined geo-
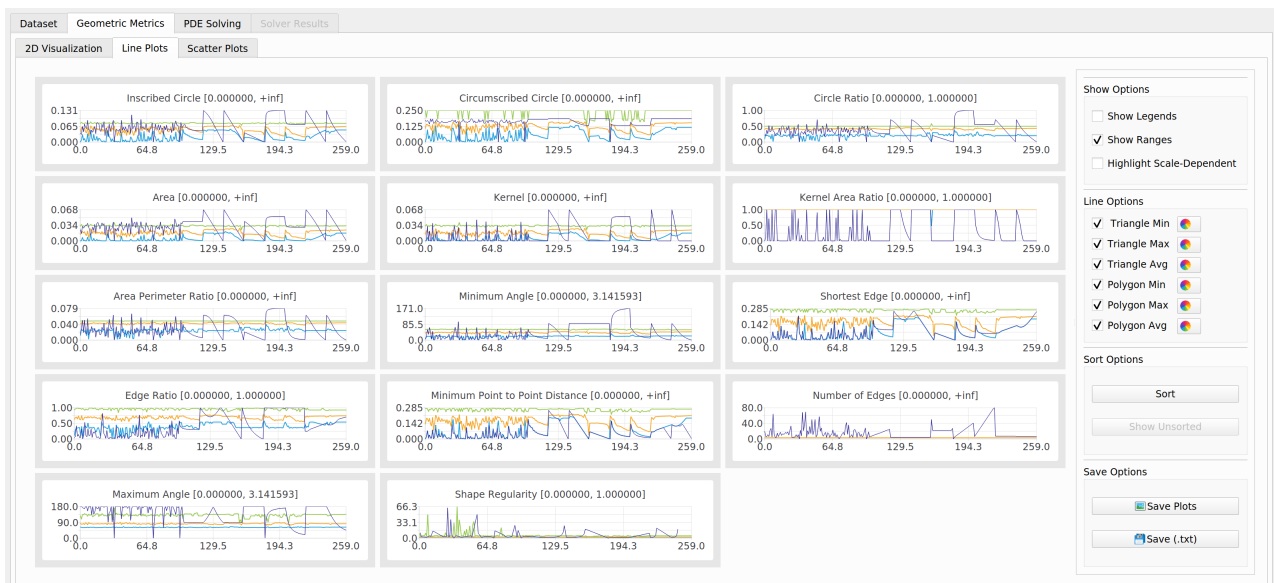
Figure 6: *Screenshot of the window visualizing the variation of the minima and maxima of each geometric metric through line plots. By double-clicking on a single plot, the selected plot detaches from the window and enables a full-screen visualization. On the right side, a specialized panel allows graphical settings and image saving.*

metric metrics with each other. The correlation is shown by scatter plots, each of them built by setting its axes to the two selected metrics, respectively (Sect. 3.4). In both windows, plots are interactive: a single click on a point or line in the plot shows the value corresponding to such a point. Also, both windows are highly customisable by enabling the user to set plot colours and point/line sizes. Furthermore, plots can be stored on disk as images, and numerical information can be stored as textual files.

3.3. PEM solver

PEMesh provides the possibility to solve PEMs on a polygonal mesh of the input domain and to visualize the performances of any PEM solver. To this end, the PEM solver is not part of the tool but is rather considered an external resource that is invoked from the graphical interface. Without loss of generality, our framework assumes that the selected PEM solver takes an input mesh and returns both the solution and the ground truth (if any) of a PDE, together with statistics (e.g., approximation error, conditioning of the stiffness matrix) on the numerical solvers (Sect. 4.3).

When the results of the PEM solver are available, a specialized window graphically shows both the solution and the ground truth on the mesh through a colour map, while numerical and geometric metrics are represented by customisable linear plots (Fig. 7). A double click on each plot detaches the plot itself from the window and enables a full-screen visualization. On the right side of the window, a set of visualization options is provided to customize both colour maps and linear plots.

3.4. Correlation visualization

PEMesh is intended to be a support for the investigation of possible correlations between geometric and solver performance metrics. To reach the goal, it provides specialized windows for the graphical visualization via scatter plots of such correlations (Fig. 8). A scatter plot is a plot displaying the relationship between two quantitative variables measured on the same input. The values of one variable appear on the horizontal axis, and the values of the other variable appear on the vertical axis. This representation allows us to analyse if the two variables are correlated. In the former case, it describes the relationship's direction, form, and strength. Direction can be either positive (rising) or negative (falling), while the form can be linear or curvilinear. Finally, the strength is derived from the plot's slope, indicating if the relationship is strong, moderate or weak.

PEMesh provides visualization of possible correlations between two different geometric properties and between a geometric metric and a solver performance evaluation. The former is enabled when both geometric metrics and PEM performance evaluations are available.

Given two variables G and G' computed on the same set of input polygon meshes, where G is a geometric metric and G' may either be a geometric metric or a PEM performance evaluation, this module visualizes their correlation through a specialized scatter plot, with G-values on the *x*-axis and G'-values on the *y*-axis. Similarly to any other plot in the application, these scatter plots are highly customisable in terms of the point colour, size, and labels, and can be exported as images.
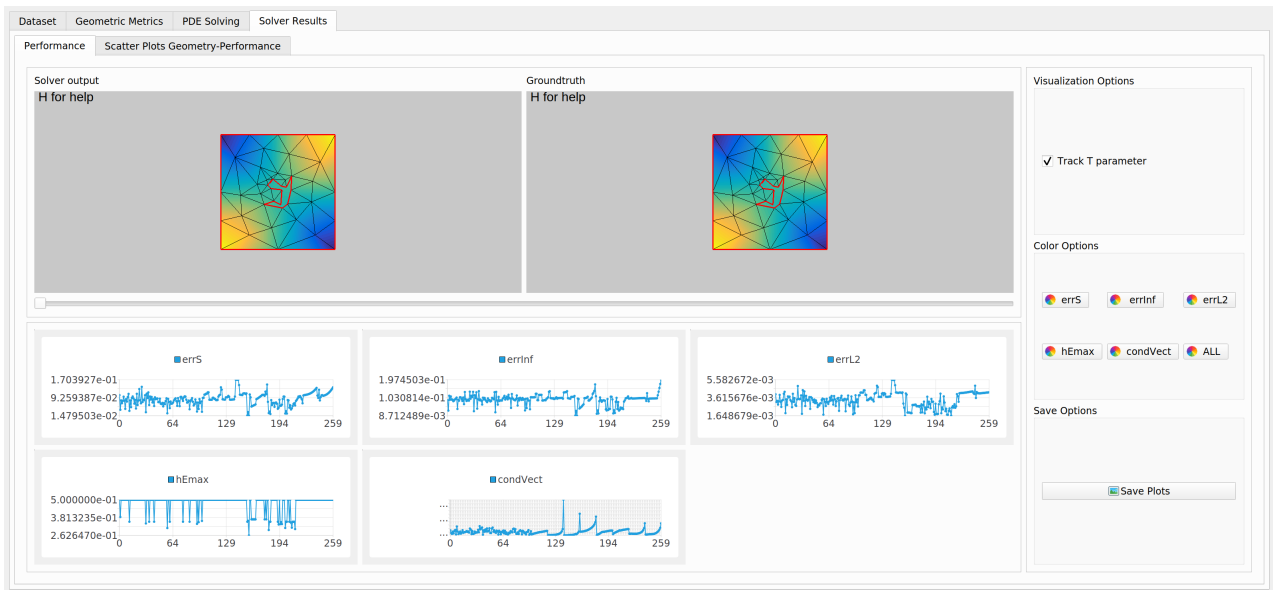
Figure 7: *Screenshot of the window visualizing PEM solver results. On the top, both solver output and ground truth are colour-mapped on the input polygon meshes, while on the bottom a set of linear plots show how solver performances vary in the data set.*
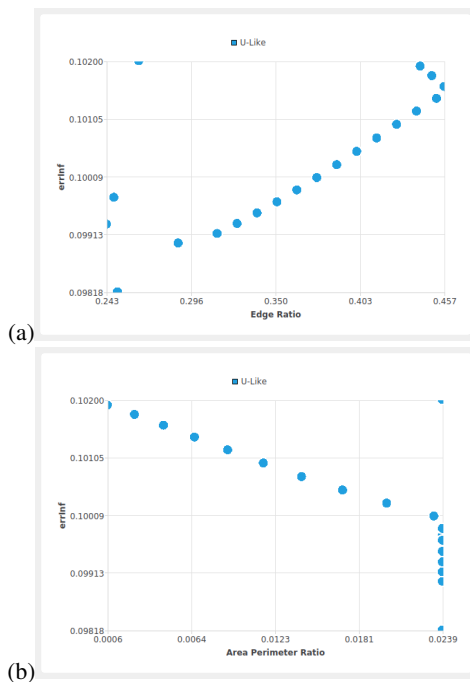

Figure 8: *Examples of scatter plots. (a) Positive strong correlation. (b) Negative strong correlation.*

4. Implementation

PEMesh is a standalone multi-platform desktop application implemented in C++ and exploits Qt libraries for the design and implementation of the graphical user interface (Fig. 1) and Cino-

lib [Liv19] to generate and visualize meshes. The software we developed is open-source and publicly available. We now discuss the supported data formats (Sect. 4.1), the family of parametric polygons available for the generation of polygonal meshes (Sect. 4.1), and a description (Sect. 4.3) of the PEM solver used for the experiments on the interplay between geometry and analysis.

4.1. Supported data formats

As mentioned, PEMesh provides the possibility to either load an existing data set or generate a new one from scratch. In both cases, it supports the most widely used file formats for the exchange of polygonal meshes, namely OBJ, OFF and STL. Furthermore, an additional output format is provided and produces *.node* and *.ele* files encoding vertices and polygons respectively. This latter mesh format is provided to support a large amount of PEM solvers requiring this kind of input.

4.2. Input polygons

To generate new polygon meshes from scratch, the user is asked to select one or more polygons to be added to the domain. Two types of predefined polygons are made available for user selection: parametric and random [ABB*21]. When a set of random polygons is selected, the generated data set is made of a single mesh; if at least one parametric polygon is chosen, then a family of meshes $D = \{M(0), \ldots, M(1)\}$, is generated, where $M(0)$ contains all the parametric polygons at its initial phase (e.g., they do not present critical geometric features (Fig. 9(a)) and they are progressively made worse by a deformation, controlled by the parameter $t \in [0,1]$ (Fig. 9(b)). In the latter case, the number of generated meshes is user-defined.
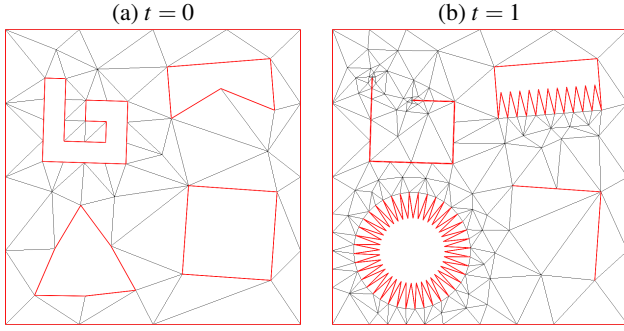
(a) $t = 0$ (b) $t = 1$

Figure 9: *Multi-parametric polygon meshes.*

Other than the predefined parametric polygons, the user is allowed loading polygons from a file. Such a polygon is automatically scaled and translated to be placed inside the domain, and additional editing can be applied by the user. Users are allowed to save the polygon configuration into a CSV file to be possibly reloaded during any following experimental session. The CSV stores, for each polygon, any data necessary to rebuild the configuration (i.e., position, scale, rotation).

4.3. PEM solver

As aforementioned, `PEMesh` does not include PEM solvers but is intended to be a support for the analysis of any kind of PEM solvers, with no limitation. The only requirement to be satisfied is related to the solver output format, which must be available according to a very simple text file format. Specifically, both the numerical solution computed by the VEM solver and the ground-truth solution (if any) must be as a list of their values at the mesh vertices. Finally, these two arrays are saved in a *.txt* file whose name is composed of the input filename and an additional ending to indicate which output it encodes (i.e. either *"-solution"* or *"-ground-truth"* respectively). For each performance evaluation, an additional *.txt* file is generated and its name is composed of the input filename and an additional string to indicate the performance name. The single value representing the solver performance must be written in the file.

4.3.1. Example of an Application

As a matter of example and test, we exploit the family of parametric and random polygon meshes defined in [ABB*21] as an input dataset and the *Virtual Element Method* (VEM) [BBC*13] as an external PEM solver. Specifically, we considered a test problem corresponding to solutions to the Poisson equation in the domain $\Omega = (0,1)^2$. For our test case, the ground truth is the Franke function, namely

$$u_2(x,y) := \frac{3}{4}e^{-\left((9x-2)^2+(9y-2)^2\right)/4} + \frac{3}{4}e^{-\left((9x+1)^2/49+(9y+1)/10\right)}$$
$$+ \frac{1}{2}e^{-\left((9x-7)^2+(9y-3)^2\right)/4} + \frac{1}{5}e^{-\left((9x-4)^2+(9y-7)^2\right)} \tag{1}$$

The MATLAB® code of the method computes the PDE solution, provides the solution ground truth and also computes some evaluations of PEM solver quality, such as

- $\mathcal{L}_1$ *condition number* $\kappa_1(\mathbf{S}) = \|\mathbf{S}\|_1 \|\mathbf{S}^{-1}\|_1$ *of the PEM stiffness matrix* $\mathbf{S}$;
- *relative error* $\varepsilon_S := \|\mathbf{u} - \mathbf{u_h}\|_S / \|\mathbf{u}\|_S$, with weighted norm $\|\mathbf{v}\|_S^2 = \mathbf{v}^\top \mathbf{S} \mathbf{v}$;
- *relative* $\mathcal{L}_\infty$-*error* $\varepsilon_\infty := \|u - u_h\|_\infty / \|u\|_\infty$, between the ground-truth u and the computed u_h solutions.

To enable the visualization of the results and the correlation between PEM solver performances and geometric properties, we wrap the code into a MATLAB function to be called from a command line and we redirect the output of the *Virtual Element Method* to file, according to the file format described in Sec. 4.3. The MATLAB function requires both the input mesh and the output directory as parameters. These two simple operations are sufficient to make our framework and the *Virtual Element Method* communicate. Fig. 7 shows how `PEMesh` visualizes PEM solver performances, while scatter plots in Fig. 10 are generated by `PEMesh` and show the correlation between two geometric metrics (area perimeter ratio and minimum angle) and the conditioning number of the stiffness matrix of the PEM solver.
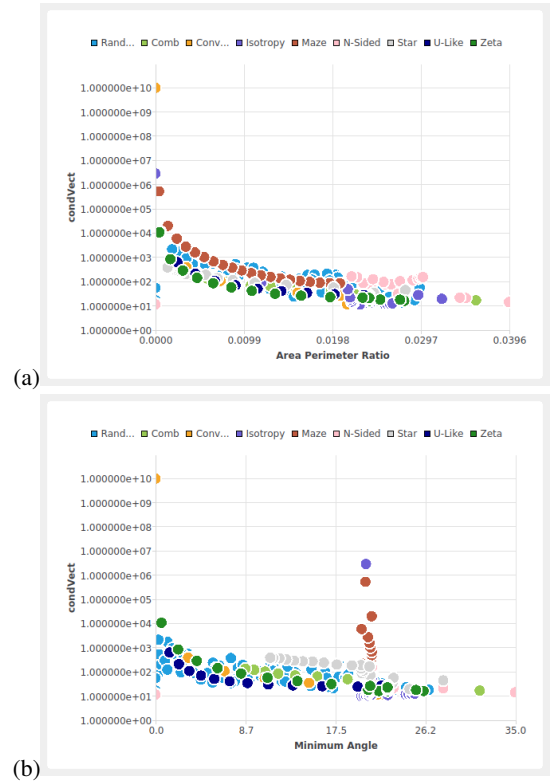
(a)

(b)

Figure 10: *Scatter plot of the correlation between geometric properties and performance metrics of the PEM solver: correlation between (x-axis) (a) the area-perimeter ratio and (b) the minimum angle with (y-axis) the conditioning number of the stiffness matrix of the PEM solver.*

5. Discussion and conclusions

We presented a novel tool helping researchers to perform experimental design and analysis of polygonal meshes for PEM solvers. Its easy-to-use graphical interface simplifies the execution of experimental pipelines, from the design of polygon meshes stressing specific geometric properties to the analysis of performances of user-provided PEM solvers. It also allows correlating geometric properties with PEM solver performances and provides advanced visualization modalities of the results of such an analysis. `PEMesh` is available as an open-source project (https://github.com/DanielaCabiddu/PEMesh) and we expect it can be employed in several research activities.

Current limitations and future works There are several directions in which our framework can be improved. First, the current implementation allows the definition of polygon meshes by loading already exiting polygons, possibly designed by exploiting external tools. Since our proposal is intended to support activities in several research fields, additional features, such as the possibility to draw polygons freehand, would simplify the mesh generation process for users coming from fields other than mesh design. A deeper analysis, including user studies, can support the development of an improved version of the tool according to user needs.

Also, `PEMesh` supports 2D meshes, but the entire architecture is agnostic to the dimension of the geometric input. It is almost trivial to extend the graphical user interface to support 3D meshes, but future investigations are necessary on 3D mesh generation approaches and the definition of geometric properties that would be likely to be of interest to the research community.

Finally, `PEMesh` is designed as a desktop application exploiting RAM to both generate meshes and run PEM solvers. At each run of the application, the complete input dataset is kept in core to improve efficiency. Nevertheless, the power of `PEMesh` is RAM-bounded and it is not robust enough to support arbitrary large input datasets (i.e. made up of a large number of large meshes). Future activities will be addressed to improve the underlying architecture and guarantee efficiency independently of the input dataset size.

Acknowledgments

References

[ABB*21] ATTENE, MARCO, BIASOTTI, SILVIA, BERTOLUZZA, SILVIA, et al. "Benchmarking the geometrical robustness of a Virtual Element Poisson solver". *Mathematics and Computers in Simulation* 190 (2021), 1392–1414. ISSN: 0378-4754 2, 3, 6, 7.

[ACC*16] ANTONIETTI, P. F., CANGIANI, A., COLLIS, J., et al. "Review of Discontinuous Galerkin Finite Element Methods for Partial Differential Equations on complicated domains". *Connections and Challenges in Modern Approaches to Numerical Partial Differential Equations*. Springer, 2016, 279–308 2.

[BBC*13] BEIRÃO DA VEIGA, L., BREZZI, F., CANGIANI, A., et al. "BASIC PRINCIPLES OF VIRTUAL ELEMENT METHODS". *Math. Mod. Meth. Appl. Sci.* 23.1 (2013), 199–214 2, 7.

[BKK08] BRANDTS, JAN, KOROTOV, SERGEY, and KŘÍŽEK, MICHAL. "On the equivalence of regularity criteria for triangular and tetrahedral finite element partitions". *Computers & Mathematics with Applications* 55.10 (2008), 2227–2233 2.

[BLM14] BEIRÃO DA VEIGA, L., LIPNIKOV, K., and MANZINI, G. *The Mimetic Finite Difference Method for Elliptic Problems*. MS&A 11. Springer, 2014 2.

[BLS05] BREZZI, F., LIPNIKOV, K., and SHASHKOV, M. "Convergence of Mimetic Finite Difference Methods for Diffusion Problems on Polyhedral Meshes". *SIAM J. Numer. Anal.* 43 (2005), 1872–1896 2.

[CDG08] COCKBURN, B., DONG, B., and GUZMÁN, J. "A supercovergent LDG-Hybridizable Galerkin method for second-order elliptic problems". *Math. Comp.* 77.264 (2008), 1887–1916 2.

[CGH14] CANGIANI, A., GEORGOULIS, E. H., and HOUSTON, P. "hp-version Discontinuous Galerkin Methods on polygonal and polyhedral meshes". *Math. Mod. Meth. Appl. Sci.* 24.10 (2014), 2009–2041 2.

[Cia02] CIARLET, P. *The Finite Element Method for Elliptic Problems*. Society for Industrial and Applied Mathematics, 2002 2.

[DE15] DI PIETRO, D. A. and ERN, A. "Hybrid High-order Methods for variable-diffusion problems on general meshes". *Comptes Rendus Mathèmatique* 353.1 (2015), 31–34 2.

[DFG99] DU, QIANG, FABER, VANCE, and GUNZBURGER, MAX. "Centroidal Voronoi tessellations: Applications and algorithms". *SIAM review* 41.4 (1999), 637–676 3.

[Hec12] HECHT, F. "New development in FreeFem++". *Journal of Numerical Mathematics* 20.3-4 (2012), 251–265. ISSN: 1570-2820. URL: https://freefem.org/ 2.

[HSL*21] HUANG, ZIZHOU, SCHNEIDER, TESEO, LI, MINCHEN, et al. "A Large-Scale Benchmark for the Incompressible Navier-Stokes Equations". *arXiv preprint arXiv:2112.05309* (2021) 3.

[Lip13] LIPNIKOV, KONSTANTIN. "On shape-regularity of polyhedral meshes for solving PDEs". *Research Notes, 22nd International Meshing Roundtable*. 2013 2.

[Liv19] LIVESU, MARCO. "Cinolib: A Generic Programming Header Only C++ Library for Processing Polygonal and Polyhedral Meshes". https://github.com/mlivesu/cinolib/. Aug. 2019, 64–76 6.

[Mas18] MASCOTTO, LORENZO. "Ill-conditioning in the virtual element method: Stabilizations and bases". *Numerical Methods for Partial Differential Equations* 34.4 (2018), 1258–1281 2.

[MWW15] MU, LIN, WANG, XIAOSHEN, and WANG, YANQIU. "Shape regularity conditions for polygonal/polyhedral meshes, exemplified in a discontinuous Galerkin discretization". *Numerical Methods for Partial Differential Equations* 31.1 (2015), 308–325 2.

[OAH*19] ORTIZ-BERNARDIN, ALEJANDRO, ALVAREZ, CATALINA, HITSCHFELD-KAHLER, NANCY, et al. "Veamy: an extensible object-oriented C++ library for the virtual element method". *Numerical Algorithms* 82.4 (2019), 1189–1220 2.

[OAH*20] ORTIZ-BERNARDIN, ALEJANDRO, ALVAREZ, CATALINA, HITSCHFELD-KAHLER, NANCY, et al. "VEMLab: a MATLAB library for the virtual element method". (Mar. 30, 2020). URL: https://github.com/cemcen/vemlab 2.

[OK18] OKEREKE, M. and KEATES, S. "Finite Element Mesh Generation". *Finite Element Applications: A Practical Guide to the FEM Process*. Cham: Springer International Publishing, 2018, 165–186. ISBN: 978-3-319-67125-3 2.

[Oro20] OROLU, KEHINDE. "Finite Element Triangular Mesh Generator. MATLAB Central File Exchange." (2020). URL: https://www.mathworks.com/matlabcentral/fileexchange/32875-finite-element-triangular-mesh-generator 2.

[RW13] RJASANOW, D. S. and WEISSER, S. "Higher order BEM-based FEM on polygonal meshes". *SIAM J. Numer. Anal.* 241 (2013), 103–115 2.

[SBMS22] SORGENTE, TOMMASO, BIASOTTI, SILVIA, MANZINI, GI-ANMARCO, and SPAGNUOLO, MICHELA. "The role of mesh quality and mesh quality indicators in the virtual element method". *Advances in Computational Mathematics* 48.1 (2022), 1–34 3.

[SDG*19] SCHNEIDER, TESEO, DUMAS, JEREMIE, GAO, XIFENG, et al. "Poly-Spline Finite Element Method". *ACM Transactions on Graphics* (2019) 2.

[She02] SHEWCHUK, JONATHAN RICHARD. "What is a good linear finite element? interpolation, conditioning, anisotropy, and quality measures (preprint)". *University of California at Berkeley* 73 (2002), 137 2.

[She96] SHEWCHUK, JONATHAN RICHARD. "Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator". *Applied computational geometry towards geometric engineering*. Springer, 1996, 203–222 3.

[ST04] SUKUMAR, N. and TABARRAEI, A. "Conforming polygonal finite elements"". *International Journal for Numerical Methods in Engineering* 61.12 (2004), 2045–2066 2.

[Sut17] SUTTON, OLIVER J. "The Virtual Element Method in 50 Lines of MATLAB". *Numer. Algorithms* 75.4 (Aug. 2017), 1141–1159. ISSN: 1017-1398 2.

[TPPM12] TALISCHI, CAMERON, PAULINO, GLAUCIO, PEREIRA, AN-DERSON, and MENEZES, IVAN F.M. "PolyMesher: A general-purpose mesh generator for polygonal elements written in Matlab". English (US). *Structural and Multidisciplinary Optimization* 45.3 (Mar. 2012), 309–328. ISSN: 1615-147X 3.

[WY13] WANG, J. and YE, X. "A Weak Galerkin Finite Element Method for second-order elliptic problems". *Journal of Computational and Applied Mathematics* 241 (2013), 103–115 2.