A Unified Topological Layer for Finite Element Space Discretization

Franz Stimpfl*, Josef Weinbub*, René Heinzl*,
Philipp Schwaha†, and Siegfried Selberherr*

*Institute for Microelectronics, Technische Universität Wien, Gußhausstraße 27-29, 1040 Vienna, Austria
†Shenteq s.r.o., Záhradnícka 7, 81107 Bratislava, Slovak Republic

Abstract. A unified topological layer for mesh generation has been created to benefit from current development, to reuse existing, well tested and reliable methods, such as the Delaunay or the Advancing Front mesh generation approach, and to combine them to be able to interchange these methods, without taking a detour using file formats. In addition, by modularizing existing meshing kernels, this approach allows to not only have one meshing interface for arbitrary dimensions, but also to have the possibility at hand to independently combine different meshing strategies.

Keywords: GSSE, C++, Mesh Generation, Topology, Library Centric Design, Finite Element Method
PACS: 07.05.Bx, 07.05.Tp, 89.20.Ff

INTRODUCTION

Many physical phenomena in science and engineering can be modeled by partial differential equations (PDEs). They range from phenomena such as mechanical deformation, heat transfer, fluid flow, electromagnetic wave propagation, and even to quantum mechanics. The involved PDEs, however, admit analytical solution in only very few, quite special cases, resulting in a high demand for the development of numerical methods for the approximate solution of PDEs. Most common are the finite element method (FEM), the finite volume method (FVM), and the finite difference method (FDM) [1].

These methods approximate numerically the solution of a linear or nonlinear PDE by replacing the continuous system with a finite number of coupled linear or nonlinear algebraic equations. This transformation involves a discretization of two different kinds. The first touches the functional space involved, limiting the occurring functional expressions to preselected forms. The second, which shall be of special interest in the following, regards the spatial discretization of the simulation domain. Shape and nature of the generated elements depend on the particular selected method. The following shall concentrate on elements which are to be used for FEM calculations.

The subdivision of the simulation domain into a collection of elements, which fills the simulation domain completely, allows to associate physical quantities to the elements, or parts of these elements, such as vertices or edges for instance. The process of obtaining an approximation requires traversing all of the generated elements and generating a matrix representation of the physical problem, which can be treated using matrix solution packages and techniques [2]. Each of the quantities recorded throughout the simulation domain results in a degree of freedom in this scheme. It is therefore immediately desirable to keep the number of elements as small as possible to reduce the resulting computational effort. On the other hand the size of the chosen elements arguably influences the accuracy of the obtained approximation [3]. Thus, the generation of the decomposition of simulation domains, which is called mesh generation, is crucial for the success of FEM calculations.

It is essential to be able to have fine grained control over the generation process in order to meet different criteria, such as the Delaunay property or anisotropic mesh elements, which may be derived either theoretically or heuristically. Such criteria are also used to assess the quality of elements [4].

A seemingly simple, yet essential feature is to offer as much control as possible over the element sizes within the mesh. Ideally, this control includes the ability to grade element sizes from small to large elements over a short distance [5].

Another goal of mesh generation, concerns the control of the shapes of the elements, since the elements should usually not have extreme shapes. Elements, which are close to equilateral shape, called rounds, for example, are commonly preferred over shapes, with large or small angles, which can degrade the quality of the numerical solution [3].

CP1281, *ICNAAM, Numerical Analysis and Applied Mathematics, International Conference 2010*
edited by T. E. Simos, G. Psihoyios, and Ch. Tsitouras

Currently there are various meshing kernels available, each with its own strength and weakness, addressing the meshing difficulties [6–9]. One particular bottleneck is that these kernels only address some of the difficulties for a given dimension, but so far none of them offers an approach to address arbitrary dimensions.

Using several meshing tools at the same time also reveals interoperability issues, such as different file formats used, which are not always convertible without loss of information.

The main goals of our developed meshing process are the possibility to introduce more control and flexibility over the meshing process without special regard for the target dimension and without having to convert needlessly between different file formats.

We focus on the task of generating unstructured, simplicial meshes, composed of triangles or tetrahedra. First an overview of the underlying mesh generation process and then the topological layer is presented.

FINITE ELEMENT MESH GENERATION

The process of mesh generation can be divided into three major steps: the extraction of components defining the topology, the meshing of the volume's boundary, i.e. surface meshing, and the meshing of the volume itself. While the basic sequence remains the same no matter, how many dimensions are considered, specializations for each of these steps for a given dimension exist, especially for the two- and three-dimensional cases.

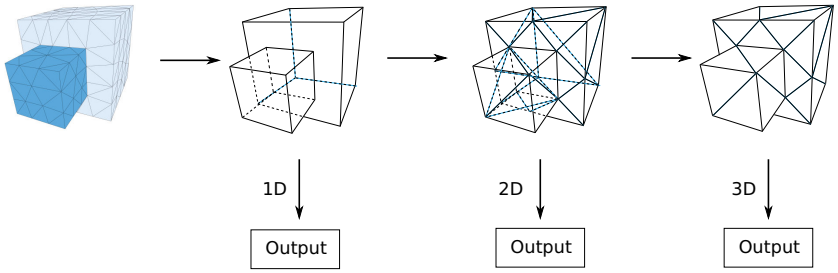

FIGURE 1. This figure depicts the basic meshing process. Starting from the isolation of structural and non-manifold edges, resulting in a one-dimensional output, the boundary of the structure is meshed. This produces a two-dimensional output which is the starting point for the final step, the meshing of the volume of the structure.

The first step is essential in establishing consistently oriented manifold structures which are processable by the subsequent algorithms. To this end structural edges and non-manifold edges [10] are isolated and the input is partitioned accordingly. During this step topological consistency checks, e.g., surface orientation or hole detection, are performed. The input for this step can stem from various sources including tools such as Wings3D [11] or Open Cascade [12] as well as data extracted from measurement or imaging data, such as CT or MRI scans. Thus inputs from constructive solid geometry (CSG) [13] approaches as well as descriptions due to implicit specifications can be processed.

Once the general topology has been identified, the edges need to be adapted taking into consideration not only the input geometry itself but also information related to the subsequent computation and quality considerations derived from it. A very important parameter in this context is the so-called local feature size (LFS) [14] which acts as a local guide for element generation and is first applied in the subdivisions of the boundary elements.

Once the input boundary representation has been transformed to meet all prescribed criteria, the boundary is meshed. The topological dimension of this mesh generation step is reduced by one in relation to the desired output mesh, as is consistent with an abstract boundary operation [15]. As in all following meshing steps, the boundary meshing step aims to generate as coarse a mesh as possible, which meets the dictated criteria such as the LFS.

Figure 1 depicts the meshing process and the possibility to extract a valid mesh after each step of the process, thereby also making the refinement of selected areas available using the presented approach.

Incorporating the Generic Scientific Simulation Environment (GSSE) as the central component of the unified approach, also provides facilities to access not only the highest level elements or the vertices of the resulting mesh, but all the elements in all of the intermediate dimensions. Thus in a three-dimensional simplex mesh, not only the simplices and vertices may be accessed, but also faces and edges.

THE UNIFIED TOPOLOGICAL LAYER

The motivation for the development of a unified layer for mesh generation is to reuse existing, well tested and reliable methods in combination with advances due to current developments as well as to establish interoperability between various different implementations and methodologies, so as to make them easily interchangeable without resorting to often detrimental file operations. Beside providing a single interface for meshing tasks of various dimensions a modularization, which incorporates several meshing kernels, also enables the user to freely combine different mesh generation strategies.

The unified topological layer not only covers the meshing process, presented in the previous section, as a whole but also offers the possibility to exchange single modules to increase the quality of the resulting meshes.

Using orthogonal modules which encapsulate the functionality of the interfaced meshing kernels and by providing these modules with meta-information, the unified topological layer is also able to automatically choose meshing strategies which are applicable for a given domain.

The following meshing kernels have currently been included: Triangle [9], CGAL [7], Tetgen [6], and Netgen [8].

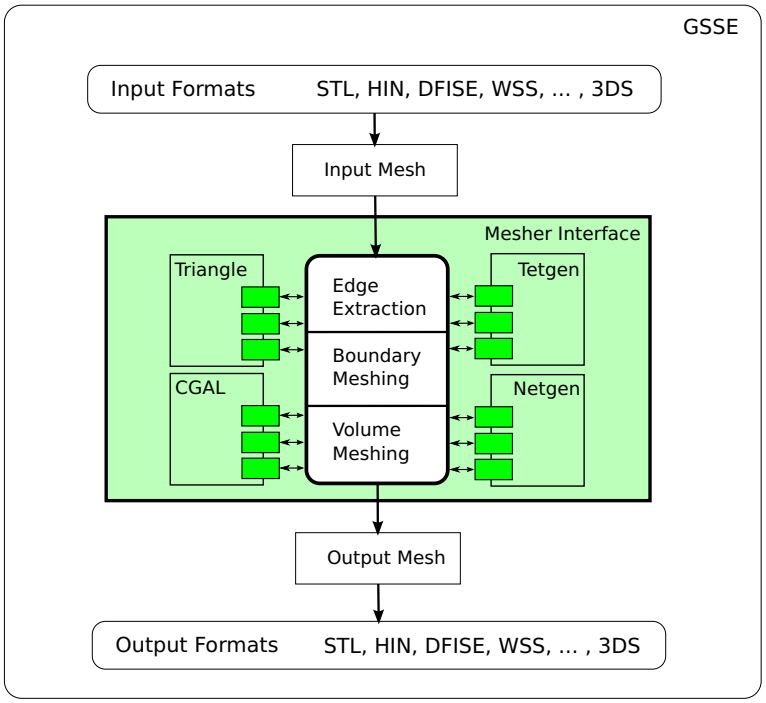

FIGURE 2. This figure depicts the presented meshing interface. It can be observed that the meshing process, presented in the previous section, is completly included. During each step of this process, each module of the interfaced meshing kernels is available, which are appropriately choosen by the meshing interface.

Additional modules may be introduced by registering them in an algorithm container and annotating the modules with tags in order to provide the required meta-information. This allows our implementation to group modules which are appropriate for given tasks at compile time into a look-up container using meta-programming techniques from the Boost library [16].

As already mentioned in the previous section, the unified topological layer is seamlessly interoperable with GSSE, which enables the user to traverse mesh elements and apply topological operators such as the analysis of the structure using Betti numbers [15].

The following depicts a source code example illustrating these possibilities. A meshing related example is presented, depicting code for finding inconsistencies in surface meshes by calculating the size of the coboundary from the current element, whis is an edge in two-dimensions.

```
// checking for inconsistencies in the 2D surface
int holes = 0;
gsse::traverse<at_cl>    // traversing all cells
[
  gsse::traverse<at_ee>  // traversing the boundary of each cell
  [
    if_(gsse::size(coboundary(_1)) != 2) [ ++ref(holes) ]
  ]
](mesh_container)
```

These tools increase the convenience from a user's perspective since a qualified preselection is already prepared, thus reducing the in depth knowledge required by the user.

CONCLUSION

Using this approach the meshing process is not bound to a certain mesh generation method, but can benefit from the combination of various established and well tested methods.

Further development of this meshing layer could include information about the PDE, the discretization method or the problem domain to choose the appropriate algorithms to create even better results. This could also lead to creating an artificial intelligence to support the mesh quality and strategy decision process.

ACKNOWLEDGMENTS

This work has been supported by the Austrian Science Fund FWF, project P19532-N13.

REFERENCES

1. R. Heinzl, *Concepts for Scientific Computing*, Dissertation, Technische Universität Wien, Austria (2007).
2. S. Balay, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang, PETSc Users Manual, Tech. Rep. ANL-95/11 - Revision 2.3.2, Argonne National Laboratory (2006).
3. J. R. Shewchuk, "What is a Good Linear Element? Interpolation,Conditioning, and Quality Measures.," in *Proc. Int. Meshing Roundtable*, 2002, pp. 115–126.
4. M. Bern, and D. Eppstein, "Mesh Generation and Optimal Triangulation," in *Comp. in Eucl. Geometry, Lect. Notes on Comp.*, 1992, vol. 1, URL `citeseer.ist.psu.edu/bern92mesh.html`.
5. F. Labelle, and J. R. Shewchuk, "Isosurface Stuffing: Fast Tetrahedral Meshes with Good Dihedral Angles," in *ACM Transactions on Graphics 26*, ACM Press, 2007.
6. H. Si, *TetGen 1.3 User's Guide*, Weierstrass Institute for Applied Analysis and Stochastics, Germany (2005), `http://tetgen.berlios.de`.
7. *CGAL - The Computational Geometry Algorithm Library*, CGAL (2010), URL `http://www.cgal.org/`.
8. J. Schöberl, *Comput. Visual. Sci.* **1**, 41–52 (1997).
9. J. R. Shewchuk, *Delaunay Refinement Mesh Generation*, PhD Thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA (1997).
10. P. Fleischmann, *Mesh Generation for Technology CAD in Three Dimensions*, Dissertation, Technische Universität Wien (2000).
11. *A Subdivision 3D Modeler*, Wings3D (2010), URL `http://www.wings3d.com`.
12. *A software development platform 3D surface and solid modeling*, Open CASCADE (2010), `http://www.opencascade.org`.
13. R. Heinzl, and T. Grasser, "Generalized Comprehensive Approach for Robust Three-Dimensional Mesh Generation for TCAD," in *Proc. Conf. on Sim. of Semiconductor Processes and Devices*, Tokio, 2005, pp. 211–214.
14. J. Ruppert, *Journal of Algorithms* **18**, 548–585 (1995), `http://citeseer.ist.psu.edu/article/ruppert94delaunay.html`.
15. A. J. Zomorodian, "Topology for Computing," Cambridge University Press, 2005.
16. *Boost C++ Libraries*, Boost (2010), URL `http://www.boost.org`.