P2MESH: an extensible set of generic classes for PDE solvers on 2D unstructured meshes

Enrico Bertolazzi* Gianmarco Manzini†

Finite element (FE) and finite volume (FV) methods using unstructured grids may approximate the solution to problems defined on general shaped computational domains. The effectiveness and the flexibility of a numerical solver implementation strongly depend on how efficiently the data sets of the underlying mesh abstraction are accessible. When a numerical algorithm is defined on an unstructured mesh the necessary topological and geometrical information must be correctly represented and efficiently manipulated by the program. It is well known to any numerical solver developer that this task may be really difficult when complex geometries are to be dealt with.

The software library P2MESH [1] supports the design and practical implementation of mesh-based data types for a wide variety of FV and FE discretization schemes on unstructured 2-D meshes. The library provides the application programmer with a collection of C++ class templates which are suitable for developing *generic* data abstraction and programming them in an object-oriented fashion.

Basically, all relevant informations to be represented and manipulated by a numerical solver, such as geometrical quantities, physical unknowns and auxiliary dependent variables, may be logically associated to an elementary mesh *entity*, namely a *vertex*, an *edge*, or a *polygon* (*triangular* or *quadrilateral*).

Therefore, mesh-based application-dependent types can be specified by using an algorithm-oriented mesh abstraction. A generic mesh-based type, containing all geometrical information, may be parameterized by a set of application-dependent types. The mesh itself is an extension of a generic type, which encapsulates a set of containers for instances of application-dependent mesh-based types.

The non-procedural programming model suggested by the library is in accord with the object-orientation philosophy, where the *different functionalities which operate upon different data are themselves part of the data specification* [4]. The most important issue is that aggregates of data and functions performing operations on data – also known as *abstract data types* [2] – allows a high level of abstraction in the software organization. The main advantage is that reusable and mantainable software is easily developed. A special programming technique, known as *static polymorphism*, has been adopted in the implementation of the library to prevents performance penalties, [3, 5].

References

[1] E. Bertolazzi, G. Manzini, *Template classes for PDE solvers on 2-D unstructured meshes*, Technical Report IAN–1124, IAN – CNR, Dec 1998 (to be submitted to ACM-TOMS).

[2] J. O. Coplien, *Advanced C++. Programming styles and idioms*, Addison-Wesley Publishing Company, New York, 1992.

[3] G. Furnish, *Container-free numerical algorithms in c++*, Computer in Physics **12**, n. 3, pagg. 258–265 (1998).

[4] B. Stroustrup, *The C++ Programming Language (Third Edition)*, Addison-Wesley, 1998.

[5] T. L. Veldhuizen, *Using C++ template metaprograms*, C++ Report **7**, n. 4, pagg. 36–43 (1995). Reprinted in C++ Gems, ed. S. Lippman.

*Dipartimento di Ingegneria Meccanica e Strutturale, Università degli Studi di Trento, via Mesiano 77, I – 38050 Trento, Italy, Enrico.Bertolazzi@ing.untin.it.

†Istituto di Analisi Numerica - CNR, via Ferrata 1, I – 27100 Pavia, Italy, Gianmarco.Manzini@ian.pv.cnr.it.