74

A Computational Environment based on a Domain Decomposition Approach

Edgar A. Gerteisen and Ralf Gruber

1 Introduction

With increasing performance of computational facilities, the complexity of geometries being tackled by scientists and engineers has grown constantly. Simultaneously, the generation of appropriate meshes has become a subject of increasing interest. One popular approach to discretize the physical space is the use of structured mesh blocks which themselves are combined in an unstructured fashion. In FE terminology this approach can be classified as nonoverlapping structured mortar elements with matching grid points. Although domain decomposition (DD) methods recently have received significant attention because of their natural route to be mapped onto distributed memory architectures, the main benefit of block-structured techniques arises from combining the advantages of both structured and unstructured approaches. Interactive tools have been proposed which assist in defining appropriate block topologies [Sei86, Con], though the difficulty and the amount of engineering time for topology generation can become considerable. Simultaneously, the interactive approach militates somewhat against automatization and embedding into parameter optimization procedures.

The present contribution first describes the parametric representation and the corresponding automatized topology generation for a composed complex shape. The system used is a graphically assisted and command language based, which can be regarded as a mixture between fully automatic and purely interactive. Consequently, it can be used to prototype the automatization of specific geometry classes or for

Ninth International Conference on Domain Decomposition Methods
Editor Petter E. Bjørstad, Magne S. Espedal and David E. Keyes

combining predefined classes that themselves are already completely automatized or unstructured representations of the computational space. Aside from the mesh generation basic generic data structures (DS) are required enabling for communication of iterative numeric schemes. The application behind the present study is the computational model of an electrostatic precipitation (ESP) process, which is used in a variety of technical processes to eliminate particles from exhaust gas. Some of the important physical properties are the electric field, the space charge, the flow velocity field, the particle charges, and particle sizes and the velocities of different flow phases [EKG97], all of which are nonlinearly coupled.

2 Computational Environment

The computational environment is a Problem-Solving Environment that aims at the realization of a system for cooperative design and development. It can be considered as a sort of skeleton for embedding applications supplemented by additional, not yet available or alternative modules introduced by well defined interfaces [GEG$^+$95]. Such an open system is especially demanded for multi-disciplinary applications, in which knowledge of specialists in different fields has to be combined [EKG97]. A data driven approach is required for realizing a transparent interaction between different modules and consequently the usage of a common database is an essential part (see Figure 1) [Mer95]. Data flow between modules is realized via the database and controlled by an application-dependent software layer that can be implemented by means of a command language driven user interface. The information system consists of a database monitor and a 3D graphics package, and eventually of additional application-specific postprocessing modules.

3 Geometry

The geometry under consideration consists of a channel (here two planar plates) with tube-like objects placed at different distances from the channel walls (here center plane) along the main channel direction (Figure 3). The tube-like objects have prongs along the tube axe with different angular orientation, the prongs themselves have teeth which again are defined by several parameters (see Figure 2). Parameters include the channel width, the position of the objects within the channel, and the parameters for the tube itself, e.g., diameter of tube, length of prongs, height of prongs, angle between prongs, etc.

The geometry definition procedure includes several aspects. First, the geometric surfaces need to be constructed or a given CAD description has to be transferred into a representation suited for generating a computational mesh. In general, CAD surfaces have to be reconstructed (condensation, closing of gaps, etc.), which can, in principle, be carried out with arbitrary high geometry fidelity by means of a point-wise projection. Additional geometric functionalities are required for generating a parametric topology and for defining boundary conditions fulfilling a complete and proper problem description.

Figure 1 Program environment for the ESP application. Starting from a skeleton that includes basic functionalities and together with already existing modules, additional functionalities are added to form the basis for an ESP simulation environment.

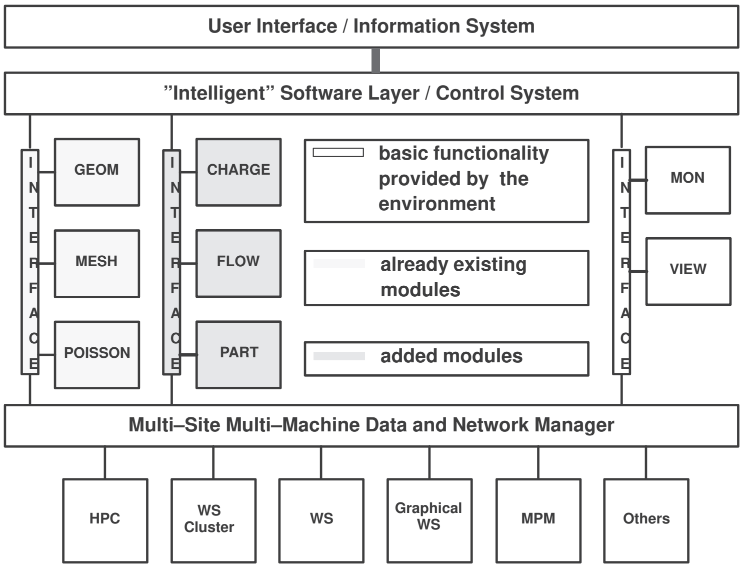
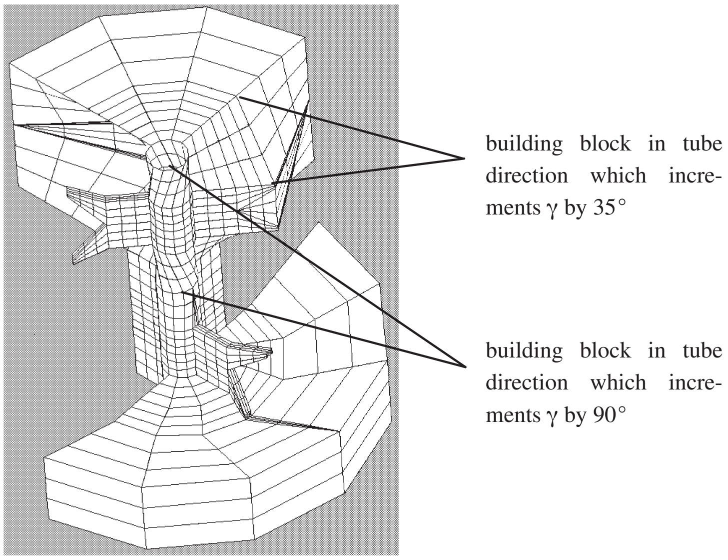


Figure 2 Part of block-structured mesh topology consisting of 380 mesh blocks (each with 3×3 cells in this illustration) demonstrating the representation of arbitrary angles between prongs in tube direction, here 135°.



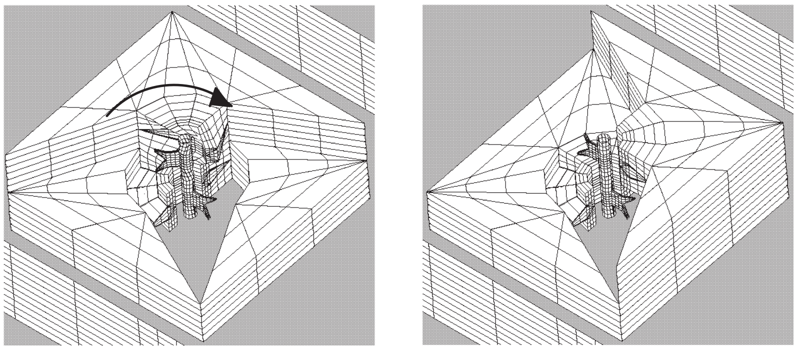
Geometry Surface Description

The basic elements of the command language are described in [MBF+90]. CAD primitives used for the surface description are create point (cp) and create face (CF). The create point function provides the basic functionality of defining CAD points and the create face function enables to generate basic CAD surfaces, which are defined by a structured set of CAD points in the two index directions. Additional important functionalities of the script language are control structures, such as "while" and branching by "if" conditions. It should be mentioned here that parts of the topology introduced afterwards, specifically the building blocks, need to be anticipated already during the construction of surface patches.

Topology Generation

First of all a strategy for the topology design has to be developed. The basic geometry is a rectangular channel. Several of the complicated tube-like structures (Figure 2) need to be fitted into this base configuration for which hexahedron building blocks are used (Figure 3). Inside those elementary geometric objects a transition to a circular tube is defined such that a quantum movement in circular direction of $\gamma = 90°$ is rendered possible, simultaneously maintaining matching grid points. The elementary object is divided further into four substructures in tube direction, two of them for allowing an arbitrary angle between prongs and one for each prong tooth (Figure 2).

Figure 3 Part of the mesh showing the hexahedron building block around tube-like structure. Identical domain numbers are activated and the angle variation between the prongs are causing a movement of the upper part by 90°. Angle of prongs: left picture = 135°, right picture = 180°.



Volumetric objects are created by defining opposite surfaces, again by cp and CF commands, for each domain. The CSD, create structured domain, command defines a domain connected by two faces. The structured domain is a CAD element and should not be mistaken with the structured mesh. A more comprehensive description of the outlined steps is given in [Ger96]. For the given example, with two tube elements and two prongs in tube direction activated, a topology consisting of 380 domains is generated. This number can easily reach the order of several thousand upon parameter variation.

Boundary Conditions

An important issue is the introduction of boundary conditions (BCs). In fact, the topology may vary upon parameter variation (see Figure 3) and the definition of BCs need to vary correspondingly. Therefore, geometric attributes (referring to inflow, outflow, solid wall, etc.) are placed on each CAD surface, which are transferred to the structured mortars and interpreted afterwards by the computational modules. The BCs are introduced within CSD command. The boundary-condition-on-face commands are embedded in control structures, since they are supposed to branch upon parameter variation. Additional modules exist for checking the completeness of the problem definition, e.g., no disjoint internal surface patches exist. All free surface patches are furnished with proper boundary condition codes.

4 Mesh

The computational mesh is derived by applying curvilinear interpolation within each CAD volume separately. The mesh module allows also for adaptive mesh generation based on a mesh density function [Bon90]. This monitor function is, in general, defined separately in the discrete space and it resides as an object in the database. It can be based on a combination of physical properties or on estimates for the discretization error of a numerical scheme. The adaptation algorithm is based on equidistribution principle and it applies a one-dimensional r-refinement (redistribution of mesh points maintaining the structure) along each index direction of a structured mesh block similar to the work of [DKS80]. However, regularization concepts [And87, CAA87] need to be introduced, in order to avoid skewed meshes and discontinuities of the mesh lines at block interfaces in the first derivatives.

5 Solver Modules

According to the multi-disciplinarity of the considered problem, diverse computational modules are included in the environment, e.g., finite element, finite volume, particle transport, characteristic method, each of them requiring a different type of communication. The cell-centered finite volume flow solver is based on a data structure of overlapping cells, whereas a master/slave DS appears to be better suited for nodal based schemes [Ger94]. The communication matrices are derived by a separate module and saved in the database for the sake of modularity. Alternatively they can be produced by calls to a specific library.

One example is the finite element solver module that, in the present application, is used for the computation of the electrical potential. The iterative conjugate gradient solver has been reimplemented recently to allow for an efficient usage of latest computer architectures. The system matrix is not completely assembled at the mortar element interfaces. However, each interface point is treated either as a master or as a slave and together with the corresponding communication structure the global matrix vector multiplication and scalar product can be computed properly. Those are implemented in form of a communication library that can be ported readily to different hardware architectures. Results on the NEC-SX4 indicate good performance on emerging vector based parallel architectures, namely 0.9 Gflop/s on one and up to 2.7 Gflop/s on four processing elements of the SX4 vector multiprocessor system, with a minor parallelization effort by means of directives.

6 Conclusion

A data driven environment based on DD has been presented. The current application uses structured mortar elements, yet completely unstructured environments already have been realized [GEG$^+$95]. Automatized parametric topology generation is rendered possible by an extension language-based mesh generation. The master/slave DS together with the concept of not completely assembled matrices has proved effective for vertex based schemes in DD with matching grid points, i.e. it leads to high modularity and provides identical convergence behavior compared to the non-decomposed domain. The data driven skeleton described has already been used for several applications such as laser optimisation, gyrotron, magnetohydrodynamics, etc. [GEG$^+$95]. The present article is mainly based on the ESP application, which is demanding because of the multidisciplinary nature of the physical problem. The unified data representation together with the common database allows for a modular approach enabling the transparent combination of knowledge in different disciplines and concurrent program development at different sites.

Acknowledgement

This work has been supported by ABB Corporate Research, CH-Baden/Dättwil. We are especially grateful for the opportunity to work on a complex industrially relevant design. The geometry is courtesy of ABB Fläkt, S-35187 Växjö.

REFERENCES

[And87] Anderson D. (1987) Equidistribution Schemes, Poisson Generators, and Adaptive Grids. *Applied Mathematics and Computation* 24: 211–227.

[Bon90] Bonomi E. et al. (1990) Astrid: A Programming Environment for Scientific Applications on Parallel Vector Computers. In Devreese J. and Camp P. V. (eds) *Scientific Computing on Supercomputers II*. Plenum Press, New York.

[CAA87] Connett W. C., Agarwal R., and Achwartz A. L. (1987) An Adaptive Grid-Generation Scheme for Flowfield Calculations. Technical Report AIAA-87-0199, AIAA 25th Aerospace Sciences Meeting, January 12-15 1987, Reno, Nevada.

[Con] Control Data, *ICEM CFD/CAE*.

[DKS80] Dwyer H., Kee R., and Sanders B. (1980) Adaptive Grid Method for Problems in Fluid Mechanics and Heat Transfer. *AIAA Journal* 18(10): 1205–1212.

[EKG97] Egli W., Kogelschatz U., and Gerteisen E. (1997) 3D Computation of Corona, Ion Induced Secondary Flows and Particle Motion in Technical ESP Configurations. 8th Int. Conf. on Electrostatics, 4th-6th June 1997. To appear in the Journal of Electrostatics.

[GEG+95] Gruber R., Egli W., Gerteisen E., Jost G., and Merazzi S. (1995) Problem-Solving Environments: Towards an Environment for Engineering Applications. *SPEEDUP Journal* 9(1).

[Ger94] Gerteisen E. A. (1994) A Generic Data Structure for the Communication of Arbitrary Domain Splitted Mesh Topologies. Technical Report CSCS-TR-94-10.

[Ger96] Gerteisen E. A. (1996) Automatized Generation of Block-Structured Meshes for a Parametric Geometry. Technical Report CSCS/SCSC-TR-96-10.

[MBF+90] Merazzi S., Bonomi E., Flueck M., Gruber R., and Herbin R. (1990) *ASTRID User Manual Rapport GASOV No. 29.* EPFL.

[Mer95] Merazzi S. (1995) *The MEMCOM user manual (version 6.3), B2000 Data Access and Data Description Manual.* SMR Corporation, Bienne, Switzerland.

[Sei86] Seibert W. (1986) An Approach to the Interactive Generation of Blockstructured Volume Grids Using Computer Graphics Devices. In Wesseling P. (ed) *First Int. Conf. on Numerical Grid Generation in CFD*, volume 29, pages 333–342. Landshut, W. Germany, 14th-17th July, 1986.