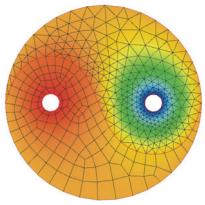
Publishing House
AKAPIT

COMPUTER METHODS IN MATERIALS SCIENCE

Informatyka w Technologii Materiałów

Vol. 13, 2013, No. 1



ModFem - A COMPUTATIONAL FRAMEWORK FOR PARALLEL ADAPTIVE FINITE ELEMENT SIMULATIONS

KAZIMIERZ MICHALIK*, KRZYSZTOF BANAŚ, PRZEMYSŁAW PŁASZEWSKI, PAWEŁ CYBUŁKA

AGH University of Science and Technology, Al. A. Mickiewicza 30, 30-059 Kraków
**Corresponding author: kamich@agh.edu.pl*

Abstract

We present the design and its' implementation for a flexible and robust modular finite element framework, called ModFem. The design is based on reusable modules which use narrow and well-defined interfaces to cooperate. At the top of the architecture there are problem dependent modules, with the main module being an incompressible flow solver. Problem dependent modules can be additionally grouped together by "super-modules", e.g. for the purpose of applying created codes for multi-physics and multi-scale problems. Additionally, the framework tries to provide suitable infrastructure for parallel computations, at the level of shared memory, as well as distributed memory systems.

Key words: fem, framework, parallelism, adaptivity, modules

1. INTRODUCTION

Most of commercial and open-source finite element codes' architectures are based on a large part, called a finite element core, that can be extended by writing special procedures, usually for new materials or physical phenomena. Because of that it is difficult to make deeper and extensive changes to codes. However, in certain situations, e.g. when trying to solve multi-physics and multi-scale problems, such changes may become indispensable. Our goal is to develop a framework for finite element computations based on a modular architecture, with modules smaller than in standard finite element codes (Banaś, 2004c). We believe that smaller modules are easier to manipulate and, hence, they could be modified for the purpose of solving special kinds of problems or adapting to new computing environments, like massively multi-core or hybrid processors.

The main idea and the novelty of the ModFem framework is the fact that there is no single code for solving different types of problems, but that parallel adaptive FEM applications are generated by the framework on demand. This approach can have advantages, especially for large-scale problems, that often enforce the use of distributed grids and message passing libraries. One of possible scenarios of the framework usage is that a group of scientists, who want to solve some new problem, writes one or more modules that suits their needs and create a large complex parallel code using their own modules and several modules provided by the framework.

2. CODE OVERVIEW

Modular architecture. The ModFem framework provides modular structure for FEM applications, presented in figure 1. Any created application is composed of several modules that interact only through strictly defined interfaces. In figure 1, the circles indicate defined interfaces, the rectangles

ISSN 1641-8581

show modules implementing and using the interfaces. The interfaces are generic and are the same for each generated application. There are several example modules already implemented and provided by the framework.

The most desirable situation would be to make it possible that every module can be exchanged with any other module with the same FEM functionality that uses and implements the same interfaces. However, for several kinds of problems or types of approximation there are constraints and limitations concerning the modules that can be used. Nevertheless we try to create modules as general as possible, in terms of the problems they can be used for, as well as in terms of computing platforms they will run on. All modules are implemented for cross platform usage and all mainstream compilers support. Process of application building is provided by cross platform compilation and linking framework.

by the problem module. The link between this module and other modules is not always direct. Problem module defines all procedures associated with the weak formulation of the problem. There are two small sets of callback routines, one for the interface with linear solvers and the second for numerical integration of terms from the weak FEM statement.

Problem utilities module (at the same level as problem dependent module, but not shown in figure 1, to keep the diagram simple). This module contains a variety of additional functions that are not linked to a given FEM formulation and are not used by modules different than problem modules. It provides certain input-output operations, auxiliary structures and procedures used in default adaptation procedures and many others.

Mesh manipulation module. Mesh module is the most basic unit. It does not depend on any other module. It determines some of the most important features

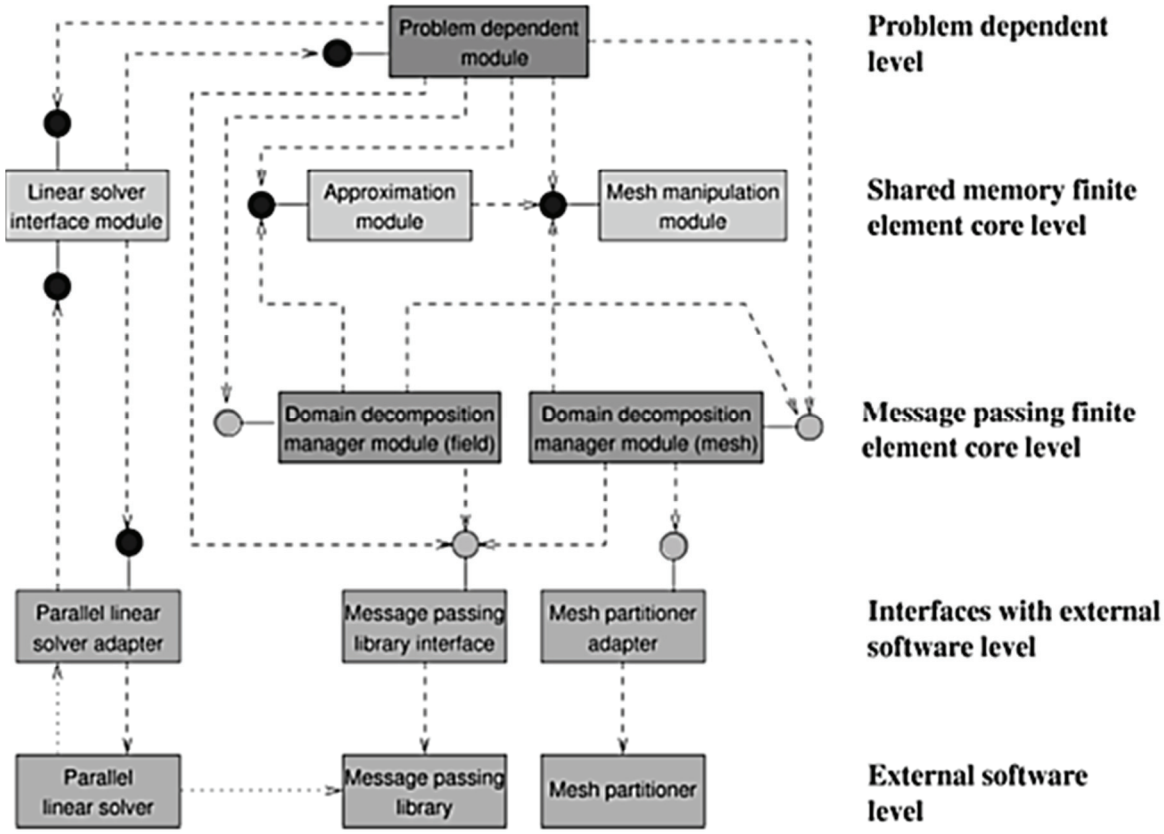


Fig. 1. *Modular structure of codes created by ModFEM framework.*

Below a short descriptions of the role of each module in created applications is provided.

Problem dependent module. Problem dependent module is the most important module. It contains a weak formulation of FEM for some selected problem. It also contains auxiliary procedures for time integration, error estimation, etc. All other modules are used

of FEM approximation. This module is responsible for all operations directly related to the geometric discretization of computational domain. All mesh entities are managed by this module. It provides iterators which allow access to the parts of the mesh. The inner logic of this module is also responsible for issues related to the orientation of the elements, faces

and edges. A very important part of the module are also a mesh adaptation and de-adaptation routines. In addition, detailed geometry sub-module is included inside this module. The geometry sub-module allows for improvement of boundary geometry during h-adaptation. Finally, mesh module implements procedures for validating a geometric mesh during input/output operations, and also supports the identification and correct location of the boundary conditions (Banaś & Michalik, 2010).

Domain decomposition manager module for mesh - distributed memory mesh module overlay. Distributed memory mesh module is an overlay for any implementation of the mesh module. The module extends the functionality of the mesh manipulation module. It adds functions responsible for domain decomposition, load balancing, transferring mesh data between computational nodes, parallel adaptation, inter-sub-domain boundary conditions and many other. It also expands the mesh entities identifiers assigning them unambiguous owner, to provide unique global identifiers. The module can handle meshes partitioned into subdomains with overlaps having arbitrary widths.

Approximation module. This module implements methods associated with the selected function space. FEM discretization requires to determine the space of functions used in approximation. Shape functions associated with elements, faces, edges and vertices are defined inside this module. Also, the calculation of the value of shape functions at a given point and their derivatives, integration over elements and faces is implemented in this module. The approximation module also includes a transformation between local and global coordinate systems.

Domain decomposition manager module for field - distributed memory approximation module overlay. Distributed memory approximation module overlay extends the approximation module. It adds functions responsible for exchanging dofs data between subdomains, synchronizing dofs in overlay managed by distributed memory mesh module, computing global (whole domain level) vector products, and transferring some additional information in distributed memory execution environments.

Linear solver interface module. The module is responsible for interfacing with external modules for solving systems of linear equations. However, all the routines related to FEM weak formulations and other problem specific routines required by the solvers, are delegated to appropriate FEM modules. This tiered approach allows for the separation of problem details and the selected solver strategy. Thanks to that, solver interface module is generic and independent from the problem being solved.

3. CAPABILITIES OF BASE MESH MODULES.

We present several capabilities of existing modules developed within the framework. We focus on mesh capabilities, since they are usually the most difficult to achieve in in-house codes. It is assumed that ModFEM users will seldom write their own mesh modules (although it is perfectly feasible), but instead, will choose between existing implementations.

Adaptation and remeshing. All mesh manipulation modules implemented in the code are designed to support different forms of adaptivity. At present there are three modules, one with prismatic elements, the second for hybrid meshes, composed of prisms and tetrahedrons (both modules support h-adaptivity) and the third, employing r-adaptivity with remeshing, designed to support moving meshes and fluid-structure interaction problems.

H-adaptation. H-adaptivity can use error estimation or some physical or geometrical criteria to determine which elements to break. Error estimation routines are implemented as problem dependent routines. Routines implementing adaptation are controlled by several parameters. They indicate a maximal level of refinement, the maximal difference between generations for neighbouring elements etc. Besides adaptation controlled by error estimation, there is also an option to manually control the process, if needed.

R-adaptation and remeshing. The main purpose of r-adaptivity in the developed module with tetrahedral elements is to account for moving boundaries of the computational domain. Remeshing algorithms are used when r-adaptivity leads to mesh degeneration. The overall flow diagram for remeshing is shown in figure 2:

1. Local domain is defined around the moving part of computational grid. This creates a new local computational domain that is a copy of the part of the global domain. Further calculations will be carried out on the new domain.
2. Each time step causes displacements of mesh vertices, which are associated with the shift of the moving part of the boundary.
3. After each time step, mesh smoothing algorithms are run in the local computational domain.

4. The quality of the mesh is checked after smoothing.
   a) If elements satisfy certain quality criteria, the position of nodes in the global computational domain is updated and the algorithm proceeds to the next time step (step 2),
   b) If the quality of elements is not satisfactory, the nodes are moved back to the position from prior time step. The local computational domain is rebuild using remeshing algorithms. During the reconstruction, the local domain can be enlarged. Time steps must be chosen in such a way, that there is at least one step with mesh smoothing only, between two subsequent remeshings.
5. Basic remeshing is done, forming the first stage of grid reconstruction. Laplace smoothing including weight points is performed to improve the distribution of element volumes and the change of the organization of faces and edges of neighboring elements (edge swapping, face swapping) takes place.
6. In the second stage of reconstruction advanced remeshing occurs. The following mechanisms are applied to improve the quality of the mesh: smoothing, point addition, element removal, change of boundary surface organization and the swapping of edges of adjacent nodes.
7. Part of the global domain is replaced by the local domain just improved by remeshing. Then algorithm returns to step 1.

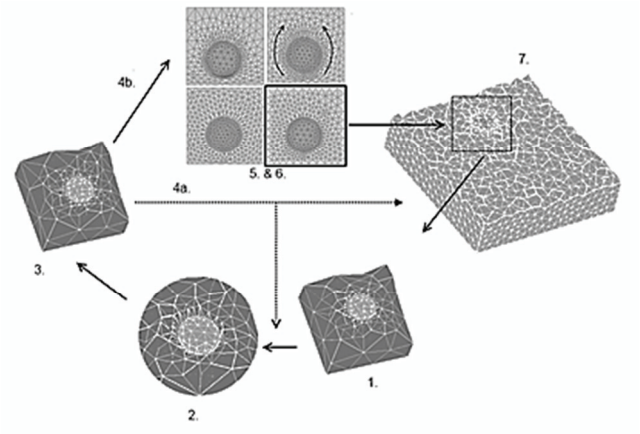


Fig. 2. *R-adaptation and remeshing scheme.*

Detailed geometry for hybrid meshes. Accurate modelling of flows requires good resolution of details of the boundary of computational domain. However, the necessary resolution may depend upon particular flow conditions, not known in advance. The mesh adaptation procedure produces regions with small elements and regions with relatively large elements. To minimize approximation error the vertices of elements lying on the boundary of the computational domain should be placed on the boundary of the modelled domain. There are two cases, the first in which there is no interaction between mesh modification procedures and geometrical modelling data and the second in which coordinates of new vertices produced during mesh refinement are supplied by the geometry module.

Modelling of boundary layers in flow simulations. Using prisms near boundary allows us to easily deal with errors related to high velocity gradient orthogonal to the boundary, imposed by no-slip boundary condition. In order to allow for proper resolution of boundary layers where highly elongated prismatic elements are well suited, in the module with hybrid prismatic-tetrahedral meshes we introduce two types of refinement for prismatic elements. First type of refinement brakes a prism into halves, which improves solution only in one direction. This refinement type is designed especially for modeling flow near boundary. Second type of prism refinement is a full refinement, which divides a prism into eight smaller similar prisms, decreasing the size of elements in all directions. Figure 3 presents an example mesh for resolving boundary layer.

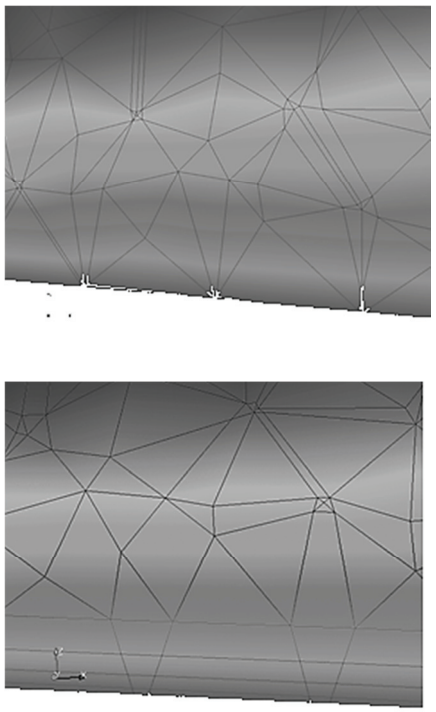


Fig. 3. *Boundary layer example. Left: cross-section of tetrahedral mesh, right: cross-section of hybrid mesh with prismatic layers near the boundary. The thickness of prismatic layers is increasing to achieve smooth transition from prismatic to tetrahedral elements.*

4. PARALLEL EXECUTION

Two levels of parallelism are exploited in the framework. The first, global level of parallelism with distributed memory and message passing, is implemented using parallel overlays for mesh and approximation modules. At the second level of parallelism we focus on a single computational node with shared memory. Both levels support multiple meshes. At the global level, domain decomposition manager module decomposes mesh into sub-meshes, which can be further merged, split or changed. At computational node level there can be distinct meshes for different fields (for multi-physics problems) or several approximation fields can share the same mesh.

Global level – computations on distributed memory systems. At this level the architecture makes use of parallel overlays designed to equip any mesh or approximation module that satisfy the specially designed for that purpose extensions to standard interfaces, with capabilities to run in distributed memory environments. There is currently only one mesh module implemented, with prismatic elements, that fully supports that level of parallel execution. The module allows for parallel h-adaptivity and load balancing (Banaś, 2004a), (Banaś, 2004b). Technical details of implementation include mechanisms for providing consistent global identification of mesh entities, especially in the case of transferring entities between subdomains during load balancing after parallel mesh adaptations.

Local level – computations on a single shared memory node. This is a parallel computation model for multi-threaded execution environments. By assumption each basic module exploits (when possible and necessary) parallel execution capabilities of the platform it is running on. Routines that take most of execution time are adapted for multi-threaded execution. If programming environment requires substantial changes to the code (like e.g. in the case of using kernel that run on GPUs or accelerators), special versions of procedures are created. The other procedures are created in such a way that thread safety is maintained. Still the fact that modules are small and independent allows for fast adaptations to changing execution platforms.

5. WEAK FORMULATION(S)

There are several problem dependent modules implemented in the framework. Below we describe the modules that are used in the multi-physics example presented later.

Fluid flow simulations. The provided problem module is capable of solving incompressible Navier-Stokes equations for viscous flows using well established SUPG-type stabilization. Problem is defined in input files covering meshes, fields, control parameters, materials, boundary conditions and linear solver parameters. The same problem module can be used for stand-alone fluid flow simulations, as well as for coupled problems. To this end the module is broken into two parts: procedures that are related to the particular weak formulation (in essence these are procedures responsible for creating element stiffness matrices and load vectors and taking into account boundary conditions) and procedures that control the whole solution process: time integration, adaptations as well as input/output operations and interfacing with linear solver.

Heat transfer. The module for heat transfer solves a single equation for time dependent temperature distribution. The equation takes into account conduction, convection in a flow field, as well as such phenomena as recrystallization. The associated boundary conditions include constant temperature, specified heat flux or radiation and convection relations. Again for the purpose of multi-physics calculations the module has a separate sub-module for creating entries to the global stiffness matrix related to the weak formulation.

Multi-physics capabilities. Modeling of many physical phenomena simultaneously is achieved by grouping problem modules together into "super-modules". This approach enables the reuse of the basic problem modules in more sophisticated context. The advantage is also the splitting of a complex physical phenomena into sequence of simpler ones. Following the principle of "divide et impera" we achieve high flexibility along with the performance and quality of coupled problem solution.

Coupled fluid flow with heat transfer problems. For the purpose of approximating this kind of problems two problem modules – fluid flow and heat transfer – are coupled using a problem "super-module". The interfaces of problem module procedures are specially designed in order to allow for such a multi-physics coupling. The coupled fluid flow-heat transfer super-module provide procedures for controlling simulations (reading input data, performing time integration, interfacing with linear solvers, writing output data) as well as performing mesh adaptations. Mesh adaptations can use error

estimates combined in an arbitrary way from separate estimates for fluid flow and heat transfer.

6. COMPUTATIONAL EXAMPLE

One of practical problems that can be modelled using the application generated by ModFem for coupled fluid flow and heat transfer, is TIG welding process. In materials being joined, due to high temperature, the so called weld pool appears. The flow in such a pool is governed by several phenomena including Boussinesq forces and surface tension. Figures 4a and 4b present illustration of an example of welding two steel plates along a line. Temperature distribution on the surface and fluid velocities at the cross-section are presented.

In the current paper, devoted to framework architecture, we briefly enumerate the modules used for the simulation. More details about this example: the mathematical model, weak formulation as well as approximation and linear solver configuration are provided in the companion paper (Siwek et al., 2013).

The selected ModFem modules were:

- fluid flow and heat transfer problem modules as well as the super-module for coupling them, all described above,
- standard linear approximation module with vertex shape functions and hanging nodes (dealt with by constrained approximation),
- prismatic mesh with regular adaptations
- direct linear solver for multi-core shared memory systems.

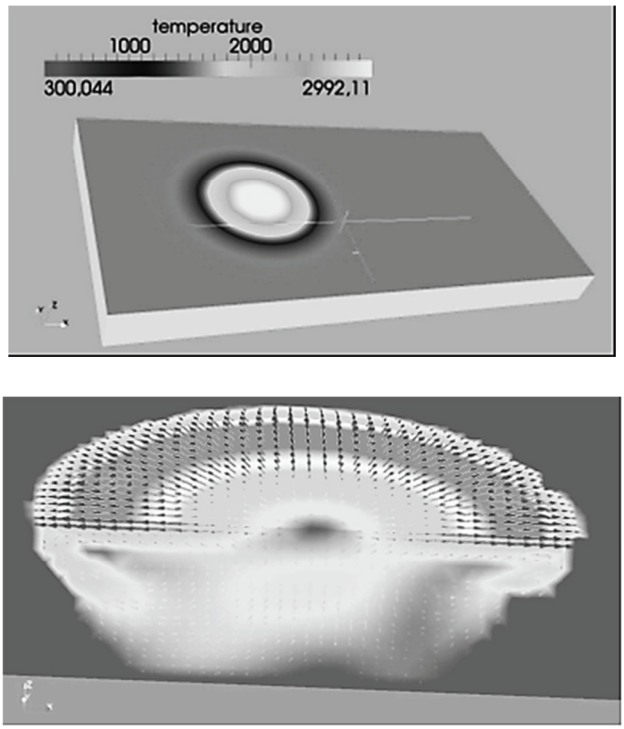


Fig. 4. *a) Calculated temperature distribution for welding pool simulations. b) Calculated liquid steel velocity distribution for welding pool simulations.*

6. FURTHER WORK

The design philosophy of ModFEM framework allows for constant improvements and extensions. At this point, we are working on extending parallel execution capabilities to all developed modules and emerging hardware and software environments, as well as enlarging the set of ready-to-use modules. The framework will be used to create codes for grid computing and complex coupled problems involving elasto-plastic deformations, heat transfer and fluid flow.

REFERENCES

Banaś, K., 2004a, Parallelization of large scale adaptive finite element computations, *Lecture Notes in Computer Science*, 3019, 431-438.

Banaś, K., 2004b, A model for parallel adaptive finite element software, *Lecture Notes in Computational Science and Engineering*, 40, 159-166.

Banaś, K., 2004c, A modular design for parallel adaptive finite element computational kernels, *Lecture Notes in Computer Science*, 3037, 155-162.

Banaś, K., Michalik, K., 2010, Design and development of an adaptive mesh manipulation module for detailed FEM simulation of flows, *Procedia Computer Science*, 1, 2037-2045.

Siwek, A., Rońda, J., Banaś, K., Cybułka, P., Michalik, K., Płaszewski, P., 2013, Modeling of Inconel 625 TIG welding process, submitted for KomPlasTech 2013 conference.

MODFEM - SZKIELET OBLICZENIOWY DO RÓWNOLEGŁYCH ADAPTACYJNYCH SYMULACJI METODĄ ELEMENTÓW SKOŃCZONYCH

Streszczenie

Autorzy prezentują koncepcję i implementację szkieletu obliczeniowego do równoległych adaptacyjnych symulacji, metodą elementów skończonych (MES), o nazwie ModFem. Głównym założeniem projektowym był podział całego szkieletu na moduły, połączone poprzez precyzyjnie zdefiniowane wąskie interfejsy. Na szczycie architektury modularnej znajduję się moduły odpowiedzialne za modelowanie konkretnych zjawisk fizycznych, w szeczególności przepływów nieściśliwych. Ponadto moduły być łączone z innymi modułami problemowymi w super-moduły, min.: w celu użycia istniejących rozwiązań podczas modelowania problemów ze sprzężeniem wielu różnych zjawisk fizycznych oraz modelowania wieloskalowego. Dodatkowo, szkielet wspiera wykorzystywanie równoległości zarówno na poziomie pamięci współdzielonej jak i rozproszonej.

Received: September 20, 2012
Received in a revised form: November 16, 2012
Accepted: December 5, 2012