OBJECT ORIENTED FRAMEWORK FOR MULTIPHYSICS ANALYSIS

G. G. Botelho [1], H. A. S. Monteiro [2], R. G. Peixoto [3], R. L. S. Pitangueira [4]

[1] Universidade Federal de Minas Gerais - UFMG, guilhermebotelho@ufmg.br
[2] Universidade Federal de Minas Gerais - UFMG, hmonteiro@ufmg.br
[3] Universidade Federal de Minas Gerais - UFMG, rodrigo.peixoto@dees.ufmg.br
[4] Universidade Federal de Minas Gerais - UFMG, roque@dees.ufmg.br

Abstract

The present work shares the new features included in the open source software INSANE (Interactive Structural Analysis Environment) for managing and solving multiphysics engineering problems. The implementation described in this paper focuses on the thermomechanical problem when the material's physical properties are temperature dependent and the problem is analyzed accordingly to the Finite Element Method (FEM). The suggested framework is based on a new entity, called Simulation Manager, responsible for setting up the analysis parameters for each physics, for controlling the solution process and for managing the data to be transferred from one analysis to the other. Both analysis are carried out using the same discrete model. Since the software is written according to the Object Oriented Programming, the existing resources for solving single-physics problems could be combined to minimize code repetition. The implementation was tested and validated by solving problems whose analytical solutions are available in the literature.

Keywords: Multiphysics, Themomechanical analysis, Finite Element Method, INSANE System

1. INTRODUCTION

The analysis and the design of industrial equipment require an understanding of the response to the actions in which the machinery will be exposed during its operating cycle. Thus, it is required for the analyst or designer to build a mathematical model simplified enough to be solved using the available methods and complex enough to provide a solution with the desired accuracy. In some situations the mathematical modeling leads to a system of partial differential equations whose solution by analytical methods is not feasible or sometimes impossible [1].

To overcome the limitations of analytical methods, numerical techniques have been developed such as the Finite Element Method (FEM) [1], the Generalized Finite Element Method (GFEM) [2], the Boundary Element Method (BEM) [3], among others. The numerical solution allows the analyst or designer to build models whose representation comes closer to the real physical phenomena. However, it is often necessary to solve a large system of algebraic equations to obtain reliable results (within suitable error bounds), a characteristic which justifies the computational implementation of proper algorithms.

Over the years, these algorithms were implemented into software packages, some of them proprietary and some open-source. These softwares for computational simulations of engineering problems usually contain a graphical interface for modeling the problem, a numerical core for mounting and solving the set of mathematical equations, and a graphical post-processing interface to display the results.

The present work refers to the inclusion of new features into the software INSANE (Interactive Structural Analysis Environment) for managing and solving multiphysics engineering problems. INSANE is a multiplatform free software, written in Java following the Object-Oriented Programming (OOP) paradigm, intended to be used as a didactic and research tool, among other purposes. This software has been developed by the Structural Engineering Department of the Federal University of

Minas Gerais (UFMG) and is available at http://www.insane.dees.ufmg.br. It has a segmented and generic numerical core independent of the pre- and post-processing graphical interfaces. The data persistence between the numerical core and the graphical interfaces is done by Extensible Markup Language (XML) files.

INSANE's numerical core is designed abstractly so that the same basic structure, which is a composition of abstract entities, applies for modeling and solving problems of different fields of engineering by different methodologies. The OOP inheritance mechanism is employed in each abstract entity to describe specific behaviors associated to the studied physical phenomena and to the adopted mathematical technique.

Tools for analyzing problems of solid mechanics by means of the FEM, GFEM, BEM and other discretization methods were already available in INSANE before the present work [4], [5], [6]. The software was also able to solve geometrically and physically nonlinear problems on the steady state and on the time-dependent regime. Resources for analyzing heat conduction problems on solids by the FEM, as well as resources for solving solid mechanics problems considering the thermal strains associated to the variation of the temperature field in the body were also already implemented in the software [7].

This paper describes a new entity introduced into INSANE's numerical core, called Simulation Manager, which acts controlling the solution processes and performing the necessary changes of the analysis parameters when necessary. This new set of classes were abstractly written, as the other entities of the numerical core, and then specialized for the thermomechanical particular case.

The numerical core description, as well its expansion, will be presented in this paper using Unified Modeling Language (UML) diagrams.

2. INSANE'S NUMERICAL CORE OVERVIEW

As described in detail by Fonseca [4], the numerical core is responsible for obtaining the results of different discrete models. The most important abstract classes and interfaces of the numerical core are Assembler, Model, Persistence and Solution.

The assembler interface is responsible for mounting the matricial system of equations, the abstract class Solution contains algorithms for solving different physical problems, the abstract class Model contains information related to the discrete model and the Persistence interface is responsible for parsing the data input and for persisting the analysis results.

Since this work refers to the management of the solution process in multiphysics simulations by the FEM, special attention will be given to the abstract class FemModel, that inherit from Model. This class contains lists of the entities that describe a finite element model. Some of these entities and its corresponding implementation will be briefly explained bellow.

In the FEM, the domain of a body is subdivided into interconnected regions, called finite elements and the solution is approximated by a set of predefined mathematical functions, which interpolate the values of the element vertices [1]. In the software, these components are represented by the Element class, the Shape function class and the Node class, respectively.

The constitutive law represents the material's response to the external actions. The constitutive laws are implemented by the ConstitutiveModel classes and the materials description are implemented by classes descending from Material [8].

The external actions are implemented by classes descending from the Loading class, and the combinations of loadings are described by the class LoadCombination.

When the body geometry is modeled in a non tridimensional simplified form, for example when a bar is modeled as a line, it is said that the geometry is degenerated. The information related to non-modeled part of the body, for example the cross sectional area of a bar or the thickness of a plate, is described by classes descending from Degeneration.

The physics being analyzed is described by the AnalysisModel class. This class informs, for example, the number of degrees of freedom contained by a node in the mesh.

The classes derived from ProblemDriver perform the calculations relative to the FEM at the element level, i.e., they calculate, for example, the element's stiffness matrix and the equivalent nodal loading associated to the external actions acting over the element domain.

It should be noted that some of the classes that compose the Model are physics dependent, specially the ProblemDriver, the AnalysisModel and the ConstitutiveModel classes. This dependence implies the need of a manager of the solution process to configure the FemModel to the current physics in a multiphysics analysis.

The solution process begins after the user call by the Graphical User Interface (GUI). This request dispatches an event to an auxiliary class responsible for initiating the Persistence and the Model and

for solving the problem for each Load Combination.

The abovementioned auxiliary class, called SolverClass, works as a basic manager of the solution process since it performs basic configurations and then manipulates the execution flow for calculating the results for each combination. However, with the SimulationManager some limitations could be overcome and the multiphysics problem readily solved. For instance:

- In multiphysics problems, each physics might be analyzed using a different Solution class. It would be possible, for example, a linear thermal solution (and thus performed by the SteadyState class) and a nonlinear mechanical solution (by the StaticEquilibriumPath class). The proposed SimulationManager can instantiate and point to the desired solution class in each physics, solving the problems in independent computational ways.
- There is a need to reconfigure the model for each physics. This configuration consists in changing the current AnalysisModel, ProblemDriver and ConstitutiveModel classes. The adoption of a manager of the solution processes allows the code to be fragmented in order to keep the configuration steps related to multiphysics only in the classes which are targeted to this purpose.
- When the thermal analysis finishes, it is necessary to create a set of temperature variation loads and add them to the remaining mechanical loads before the beginning of the mechanical analysis. The SimulationManager can handle this task in a very straightforward manner.

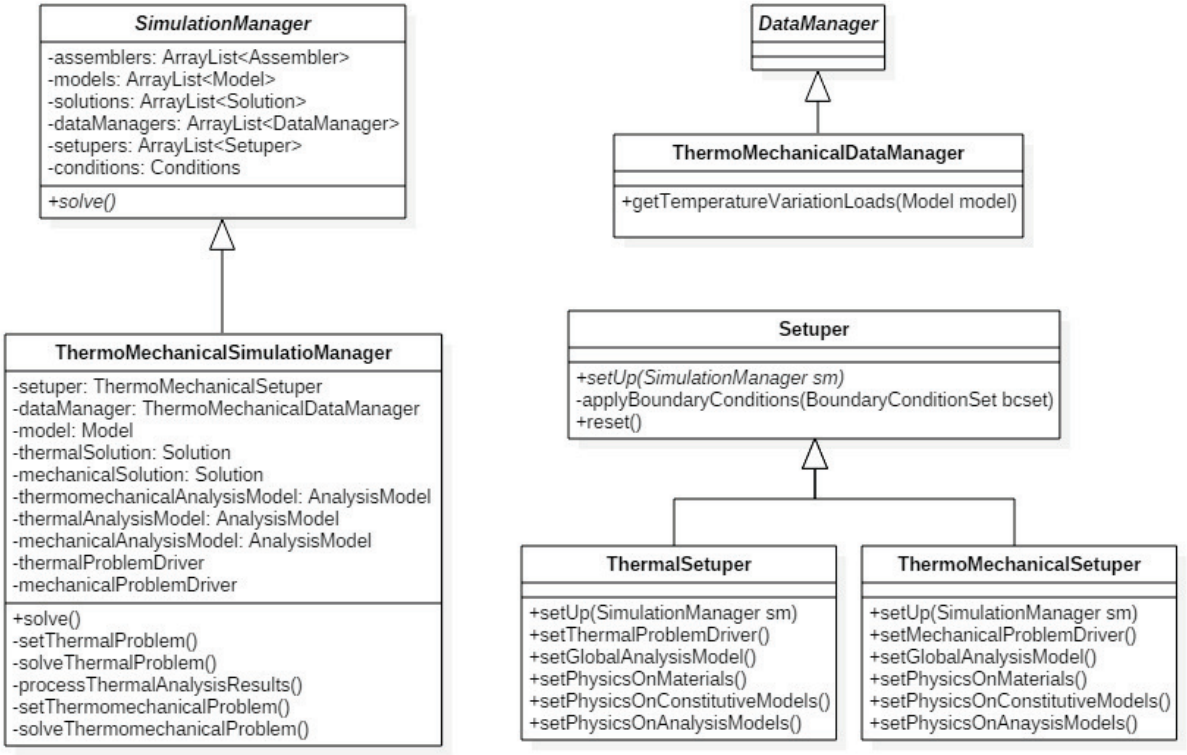

Figure 1: SimulationManager, DataManager and Setuper classes diagrams.

3. INSANE'S NUMERICAL CORE EXPANSION

The SimulationManager class is depicted on Fig. 1. This class has the public method *solve()* which is now the starting point of the solution process. This mehod is invoked by the SolverClass, previously mentioned in this paper. It is implemented in the inherited class ThermoMechanicalSimulationManager, and will iterate over the MultiphysicsCombinations. For each combination, it will call the private methods of the ThermoMechanicalSimulationManager class for:

1. setting up the thermal problem;
2. solving the thermal problem;

3. processing the thermal analysis results in order to create the temperature variation loads;
4. setting up the thermomechanical problem;
5. solving the thermomechanical problem.

After the solution of the thermal and of the thermomechanical problems an output file is generated containing the corresponding analysis results. These files are read by the GUI post-processor when the user intends to visualize them.

4. CONCLUSIONS

The proposed entity included in the INSANE's numerical core, responsible for managing the solution process, was implemented by the class SimulationManager and its child ThermoMechanicalSimulationManager. After the expansion, multiphysics analysis are being entirely performed by INSANE. The new classes were validated by solving a thermomechanical problem, in which the material's physical properties are temperature dependent and by comparing the numerical results with the analytical solution. These results will be presented in the final paper.

The new organization of the numerical core provided more flexibility for future expansions in areas where the solution process contains more than one stage, as in the case of multiscale problems or in the case of the simultaneous use of different discretization methodologies in the same numerical model.

5. ACKNOWLEDGEMENTS

The authors gratefully acknowledge the important financial support of the brazilian research agencies CAPES (in portuguese "Coordenação de Aperfeiçoamento de Pessoal de Nível Superior"), CNPq (in portuguese "Conselho Nacional de Desenvolvimento Científico e Tecnológico") and FAPEMIG (in portuguese "Fundação de Amparo à Pesquisa do Estado de Minas Gerais").

References

[1] O. C. Zienkiewicz, R. L. Taylor. *The Finite Element Method*, Vol. 1, Butterworth Heinemann, 2000.

[2] C. A. Duarte, I. Babuska, J. T. Oden, *Generalized finite element methods for three-dimensional structural mechanics problems*, Computers and Structures, 215-232, 2000.

[3] C. A. Brebbia. *The Boundary Element Method for Engineers*, Pentech Press, London; Halstead Press, New York, 1978.

[4] F. T. Fonseca, R. L. S. Pitangueira, *An object oriented class organization for dynamic geometrically non-linear FEM analysis*, XXVIII CILAMCE - Iberian Latin American Congress on Computational Methods in Engineering , Porto , 2007

[5] P. D. Alves, F. B. Barros, R. L.S. Pitangueira, *An object-oriented approach to the Generalized Finite Element Method*, Advances in Engineering Software, 1-18, 2013.

[6] R. G. Peixoto, F. E. S. Anacleto, G. O. Ribeiro, R. L. S. Pitangueira, S. S. Penna, *A solution strategy for non-linear implicit BEM formulation using a unified constitutive modelling framework*, Engineering Analysis with Boundary Elements, 295-310, 2016.

[7] G. G. Botelho, R. L. S. Pitangueira, F. T. Fonseca, *Sistema computacional orientado a objetos para análises acopladas termo-estruturais pelo método dos elementos finitos*, XXXVI CILAMCE - Iberian Latin American Congress on Computational Methods in Engineering , Rio de Janeiro , 2015.

[8] Gori, L., Penna, S.S. and Pitangueira, R.L.S., 2017. *A computational framework for constitutive modeling*. Computers and Structures, Vol. 187, pp. 1–23.