ECCOMAS Congress 2016
VII European Congress on Computational Methods in Applied Sciences and Engineering
M. Papadrakakis, V. Papadopoulos, G. Stefanou, V. Plevris (eds.)
Crete Island, Greece, 5–10 June 2016

# THE GEMA FRAMEWORK – AN INNOVATIVE FRAMEWORK FOR THE DEVELOPMENT OF MULTIPHYSICS AND MULTISCALE SIMULATIONS

**Carlos A. T. Mendes**[1], **Marcelo Gattass**[1], **and Deane Roehl**[1]

[1]Tecgraf / PUC-Rio
Rua Marques de Sao Vicente, 225. Rio de Janeiro-RJ. Brazil.
e-mail: {cmendes, mgattass, deane}@tecgraf.puc-rio.br

**Keywords:** Multiphysics Simulation, Configurable orchestration, Extensible environments, Framework, Basin modeling

**Abstract.** *This paper summarizes the GeMA (Geo Modelling Analysis) framework, a library intended to support the development of new multiphysics simulators and its integration with existing ones. GeMA uses software engineering techniques to allow engineers to focus on the programming of the physical simulation and letting the framework take care of data management and other necessary support functions to develop efficient, professional programs. These programs are capable of simulating complex problems that may involve multiple physics that interact in different spatial and time scales. GeMA architecture supports multiple simulation and coupling paradigms, with particular emphasis given to finite element methods. GeMA support includes the coupling of different physics, each one with possibly different spatial domain discretizations (meshes). It has functions to support efficient transfer of state variable values from one discretization to another. The framework also implements some important concepts of extensibility, through the combined use of plugins and abstract interfaces, configurable orchestration and fast prototyping through the use of the Lua language. In this paper, we also present some results of a 2D basin modeling test case that couples FEM non-linear temperature calculations, compaction and kinetic oil maturation and generation algorithms. This scenario includes a time evolving mesh and different time scales among physics.*

## 1 INTRODUCTION

The building of a new physical simulator comprises three major steps. In the first one, the problem at hand must be studied so that a physical representation can be created for it. This representation is then mapped to a set of equations that define the mathematical model of the problem, usually composed of ordinary or partial differential equations. In the second step, these equations are are then discretized to allow the problem to be solved. Finally, the third stage represents the construction of a simulation system that implements the models defined in the previous steps [1].

Clearly, the knowledge and skills necessary to determine the appropriate physical representation and to create the mathematical model of the problem, are different from those needed to transform this same model into a computer system capable of simulating the desired phenomena. This dichotomy is even more pronounced when the objective is not the simulation of a particular scenario, but the simulation of a class of scenarios supporting user given parameters, in an efficient and robust way.

Ideally, simulators should be developed by multidisciplinary teams. Nevertheless, it is common that these are implemented by researchers with extensive knowledge of the problem physics, but limited software engineering skills, resulting in systems that are more error-prone and difficult to maintain/extend, especially if the resulting system extrapolates the research environment and is used in a production environment by other users.

Physics and software engineering are different worlds. However, a framework supporting the development of multiphysics simulations can create a "bridge" between these two worlds, allowing the engineer to focus on the construction of the physical representation and on the mathematical model, being supported and guided by the framework to follow best programming practices. As a result, new simulators can be created in less time and with better quality, compatible with production environments.

The idea behind the GeMA (Geo Modelling Analysis) framework is to bring state of the art software engineering techniques such as extensibility, reusability, modularity and portability to engineering physical modeling, leaving engineers free to focus on the mathematical formulation of the physics of the problem. The framework takes care of all the data management and required support functions, speeding up code development. A central point is that the framework does not dictate how physics are simulated or how multiple physics are coupled. Those decisions can be made by engineers according to the scenario at hand.

## 2 THE GEMA FRAMEWORK

Executing a simulation in the GeMA environment is a two-step process. In the first step, the model data and the solution method description are loaded, defining, respectively, what will be simulated and how. In the second step, an orchestration script is executed to do the required calculations (Figure 1).

The orchestration script is the central object of the solution method. Its role is to allow the user to describe the sequence of processes that should be applied to the model, so that the desired results are calculated, providing the main simulation loop. This script is written by the user using the Lua [2] language.

The Lua language is an interpreted language specially built to be embedded into applications, allowing them to dynamically run user provided programs. It is considered a simple, easy to learn language that includes some powerful concepts, such as dynamic typing, garbage collection, functions as first class objects and the use of associative maps for building data structures.

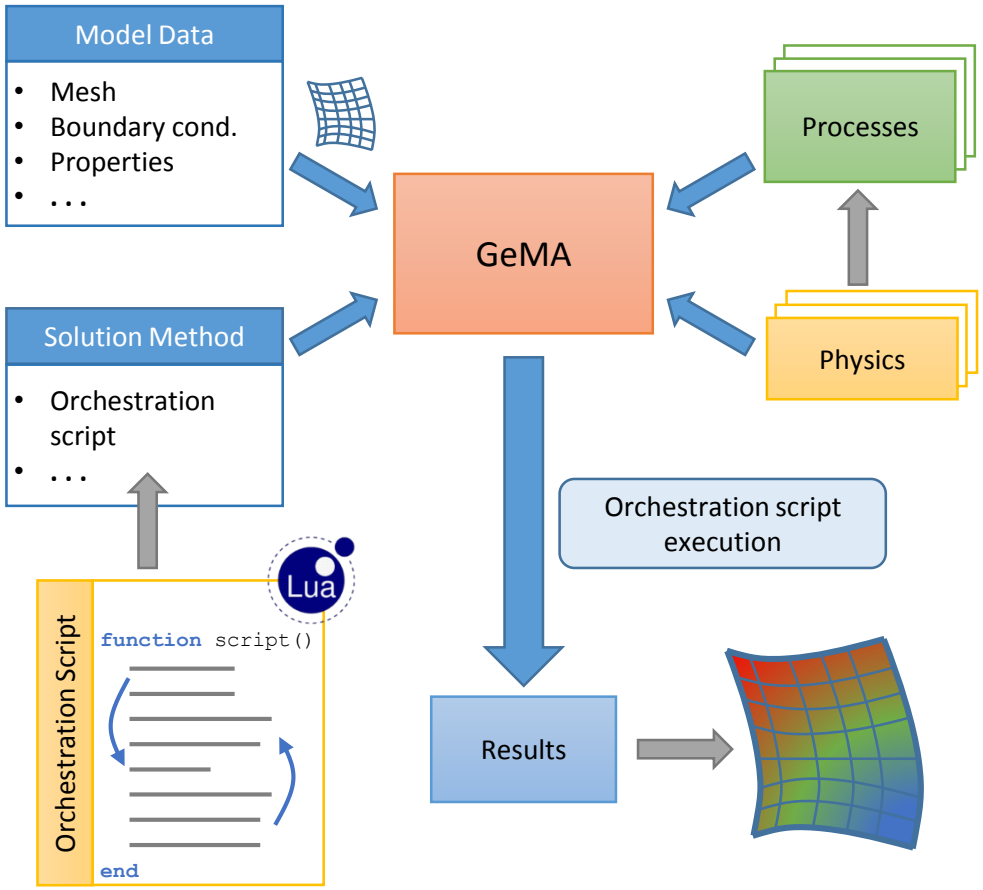


Figure 1: Main steps in a GeMA simulation.

It is an extensible language that allows the creation of domain-specific languages. Widely used as a language for integrating and extending applications, it is also considered as one of the fastest available interpreted languages[3].

By adopting a complete programming language which includes, among others, flow control, functions and support for data structures, instead of a simplified solution that only identifies the process execution order, the framework gives the orchestration script the freedom to run complex operations as needed. The GeMA orchestrator has a similar role as the orchestrator used by the Rocstar framework[4], but the flexibility given by the use of a script, instead of a C++ API as in Rocstar, allows for users with minimum computer language skills to be able to create their own orchestration models.

Processes are the basic unit used to describe the solution method and can be written in C++ or in the Lua language. In general, they are high-level primitives that describe a complete action, such as running a finite element analysis, transferring data between meshes, adaptively refining a mesh or saving a set of results.

Inside the framework, processes are abstract interfaces having concrete implementations given by plugins. Meshes, linear system solvers and other relevant entities are treated the same way. The use of abstract interfaces to model the main entities of the framework promotes its extensibility. The use of plugins, among other advantages, forces the existence of a clear separation of concepts and interdependencies, ensuring code modularity.

In this way, the combination of extensibility through abstract interfaces and plugins, with

the flexibility introduced by the orchestration script, provides the modeler with all the needed freedom to define how the simulation will be structured.

The possibility of implementing processes and physics in Lua, instead of in C++, allows for the framework to be used for rapid prototyping of new ideas, which, once tested and validated, can be converted into C++ code for greater efficiency, if at all needed.

The current framework implementation includes processes to support solutions based on the finite element method. These processes, in turn, define abstract physics interfaces, also implemented by plugins, which are responsible for providing the FEM process with the specific discrete mathematical formulation for a problem. Other discretization methods are supported by creating new processes. The framework also supports processes for integration with pre-existing simulators.

For several reasons, multiphysics models can contain more than one spatial domain discretization. To support those situations, where there is need to work with multiple meshes, at multiple scales, including heterogeneous types of elements and possibly representing different partitions of the spatial domain, the GeMA framework allows the simulation model to contain a set of meshes and includes processes for implementing data transfer between them, supported by spatial index data structures.

## 3 TEST CASES

Several test simulations were implemented to assess the framework correctness and expressiveness, including basic tests using the finite element method for stress and temperature calculation. Stress simulations were carried out with linear and non-linear trusses and with elements under plane stress state. Temperature simulations were based on heat conduction in steady and transient states, with several types of boundary conditions and the possibility of using Lua user defined functions to create temperature dependent material properties, such as thermal conductivity, making the problem nonlinear. Phase change scenarios were explored by the effective heat capacity method [5]. Results were compared with analytical models and/or with literature results.

Multiphysics simulations were tested by coupled stress-temperature models and by a considerably more complex scenario of a 2D sedimentary basin model, described below. This model includes treatment for several physical phenomena, such as geological layer sedimentation with mechanical compaction, thermal history and hydrocarbon maturation and generation. The basin evolution over time requires the use of a dynamic mesh to follow the deposition of sedimentary layers and igneous intrusions, whose presence makes it necessary to use adaptive time steps in the simulation. Temperature calculations were made based on the finite element method process, implemented in C++. Other calculations, including the mesh evolution over time, were implemented in Lua to evaluate the environment potential.

Basin modeling consists of a set of techniques designed to study the formation and the evolution of sedimentary basins. Through the use of physical simulation, basin models are used to characterize the petroleum system and to quantify potential hydrocarbon accumulations, clarifying the risks involved in exploration processes. More details can be found in [6].

The basin model under study is provided to the simulator through a mesh representing the current layer geometry, associated with a table that provides, for each layer, ages for its deposition start and end. Figure 2 shows the mesh used in the example simulation. Due to confidentiality issues, this model is not a real case, having been built to illustrate the simulator features. It contains two intrusive layers represented by the “diabase” layers.

Layer compaction and decompaction calculations are done in 1D through equation 1, given

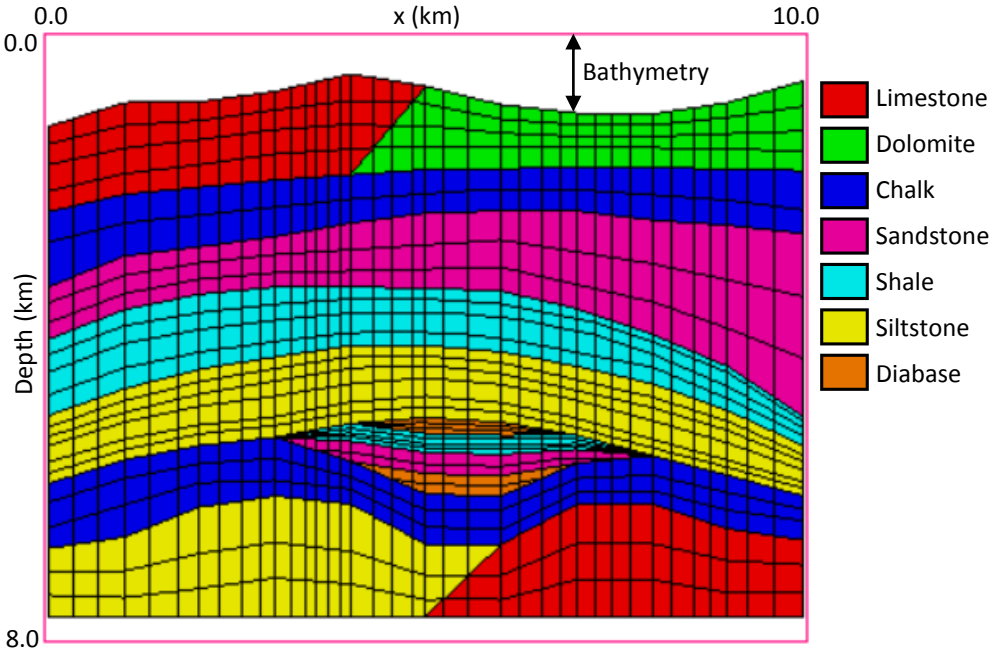


Figure 2: Sample basin model for the test simulation.

by [7], where $\phi_0$ and $c$ represent, respectively, the material initial porosity and compression rate, $z_1$ and $z_2$ are today's layer top and bottom depths and $z'_1$ and $z'_2$ are the layer top and bottom depths at the compaction/decompaction moment.

$$z'_2 = z'_1 + z_2 - z_1 - \frac{\phi_0}{c}(e^{-cz_1} - e^{-cz_2}) + \frac{\phi_0}{c}(e^{-cz'_1} - e^{-cz'_2}) \tag{1}$$

Some restrictions were imposed on the current mesh layout to ease simulation mesh evolution over time and compaction calculations. The mesh should be structured and composed of strips of vertically aligned quadrilaterals and/or triangles. Each layer should also consist of a single mesh row. If a material change is necessary within a layer (facies change), that must occur along a triangle diagonal, so that any vertical mesh edge always have the same material on both sides, as proposed in [8]. This allows for 1D compaction calculations along vertical mesh lines, without concerns about which material parameters should be used.

Temperature calculations over time are based on equation 2, where $\rho$ is the material density, $c_p$ is its specific heat capacity at constant pressure, $T$ is the temperature, $\lambda$ is the thermal conductivity and $G$ is the rate of internal heat generation. This equation is discretized in space with the finite element method and in time by an implicit method based on finite differences. Applied boundary conditions are given by the surface temperature, $T_s(x,t)$, and by the heat flow at the base of the basin, $q(x,t)$, both varying over geological time.

$$\rho c_p \frac{\partial T}{\partial t} = \nabla \cdot (\lambda \nabla T) + G \tag{2}$$

Sedimentary layers are porous media, and, therefore, material properties must reflect the grain-fluid mixture in each layer. To do so, the model considers that density and internal heat generation rates are porosity functions, $\rho(\phi)$ and $G(\phi)$. Specific heat capacity and thermal conductivity are also functions of the temperature, $c_p(\phi, T)$ and $\lambda(\phi, T)$, turning the problem into a non-linear one. If desired, the model can also consider that the thermal conductivity

is a function of the oil and gas saturation in the layer $\lambda(\phi, T, S_g, S_o)$. Mixture models and temperature dependency models used in the simulator can be found in [6].

Kinetic models are used to quantify the conversion rate of a compound into its derivatives, and can be used to quantify how the organic matter present in the source rock reacts to changes in temperature over time, reducing the initial quantity of kerogen and forming hydrocarbons. They can also be used to predict vitrinite reflectance (Ro), an important maturation indicator. Equation 3 calculates the fraction of the converted material, depending on the thermal history $T(t)$ and on the organic matter kinetic properties, given by its activation energy $E$ and frequency factor $A$. Assuming that the basins thermal history can be decomposed into multiple periods with constant heating rate, equation 3 can be solved analytically as shown in [9].

$$F(t) = 1 - e^{-\int_0^t A e^{-\frac{E}{RT(t)}}\, dt} \tag{3}$$

Figure 3 presents a schematic view of the simulation orchestration script, presenting the coupling model between the required physics and the main processes called in each time step.

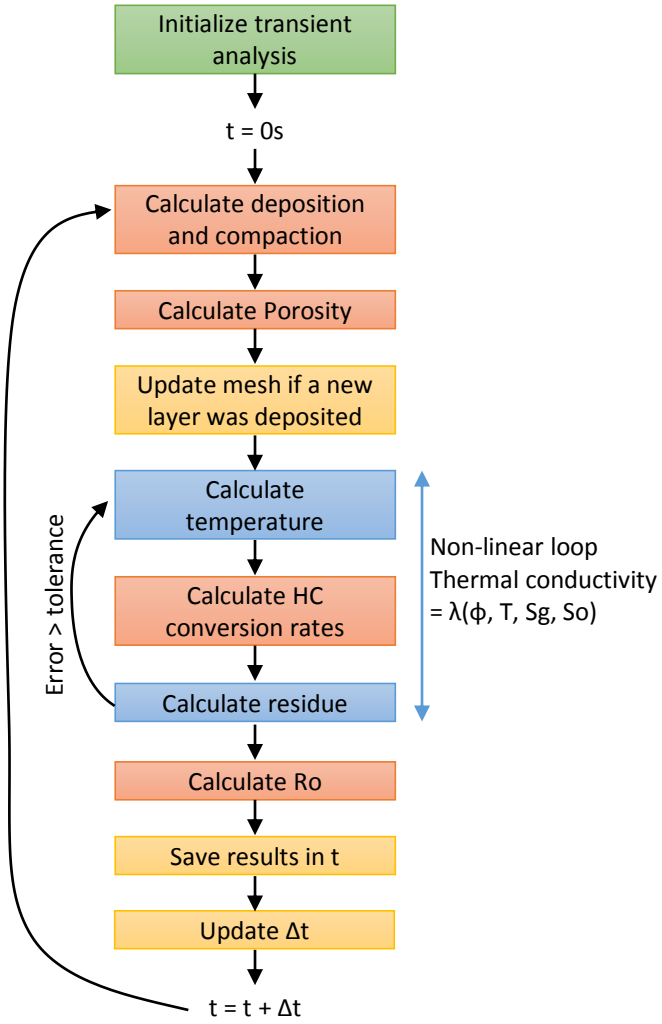


Figure 3: Schematic view of the basin simulation orchestration script.

The simulation uses a second mesh, initially empty, that is constructed and updated as layers are deposited to follow the progress of layer deposition and compaction. At each timestep, the

simulator calculates the fraction of the layer that will be deposited, ensuring a constant layer material deposition rate through time. This fraction is decompacted based on present thickness to calculate the size of the deposited layer. After deposition, older layers are compacted due to the weight of the new sediments and mesh node depths are adjusted accordingly.

If a time step involves the beginning of the deposition of a new layer, new elements are included in the mesh, and their initial temperature adjusted according to the surface temperature. If the new layer is presently over an intrusion that has not yet occurred so far in the simulation history, the new layer is "sewn" over the layer below the intrusion. Intrusion layers are always included instantaneously in a time step, being inserted between two layers, forcing the reorganization of the mesh elements in the upper one.

After compaction and porosity determination, temperature and hydrocarbon generation calculations are executed and repeated in a non-linear loop until convergence is achieved. The last process step consists in determining the time step to be used in the next iteration. After an intrusion, adaptive time steps are used to capture the cooling of the intruded body.

Figure 4 shows porosity, temperature, Ro and converted fraction values calculated at selected time steps. It is interesting to notice the influence of intrusions in Ro and converted fraction results. Temperature evolution in time steps following an intrusion event can be observed in Figure 5, which shows how the influence of the intrusion in the basin temperature history is localized and dissipates relatively quickly. In 100,000 years the effect has nearly vanished. Its impact on the maturation and generation of hydrocarbons in nearby rocks are, however, permanent.

## 4 CONCLUSIONS

The GeMA framework is, naturally, an extensible environment. The use of an architecture based on abstract interfaces and plugins allows new functionality to be easily added by third party users, without the need to modify or recompile existing code. It also promotes a separation of concepts and the decoupling between system modules, easing their development and maintenance.

Despite its current focus on simulations using the finite element method, the entire GeMA environment is prepared to include new process types, implementing support for other discretization methods. In particular, the given basin modeling example includes new processes based on analytical models for compaction, kinetic analysis and mesh evolution that were integrated with the finite element method used for temperature calculations.

The use of an orchestration script based on the Lua language as a central simulation component is largely responsible for the flexibility offered by the framework in the creation of multiphysics simulations, allowing the user to easily define coupling strategies and the actions to be taken at each time step.

The presented 2D basin modeling example illustrates some of the GeMA framework potential and flexibility. In its first version, the framework is currently being explored for the integration of hydro-mechanical simulators and their coupling with new chemical processes physics. It is being actively developed and will be further extended to deal with parallel simulations.

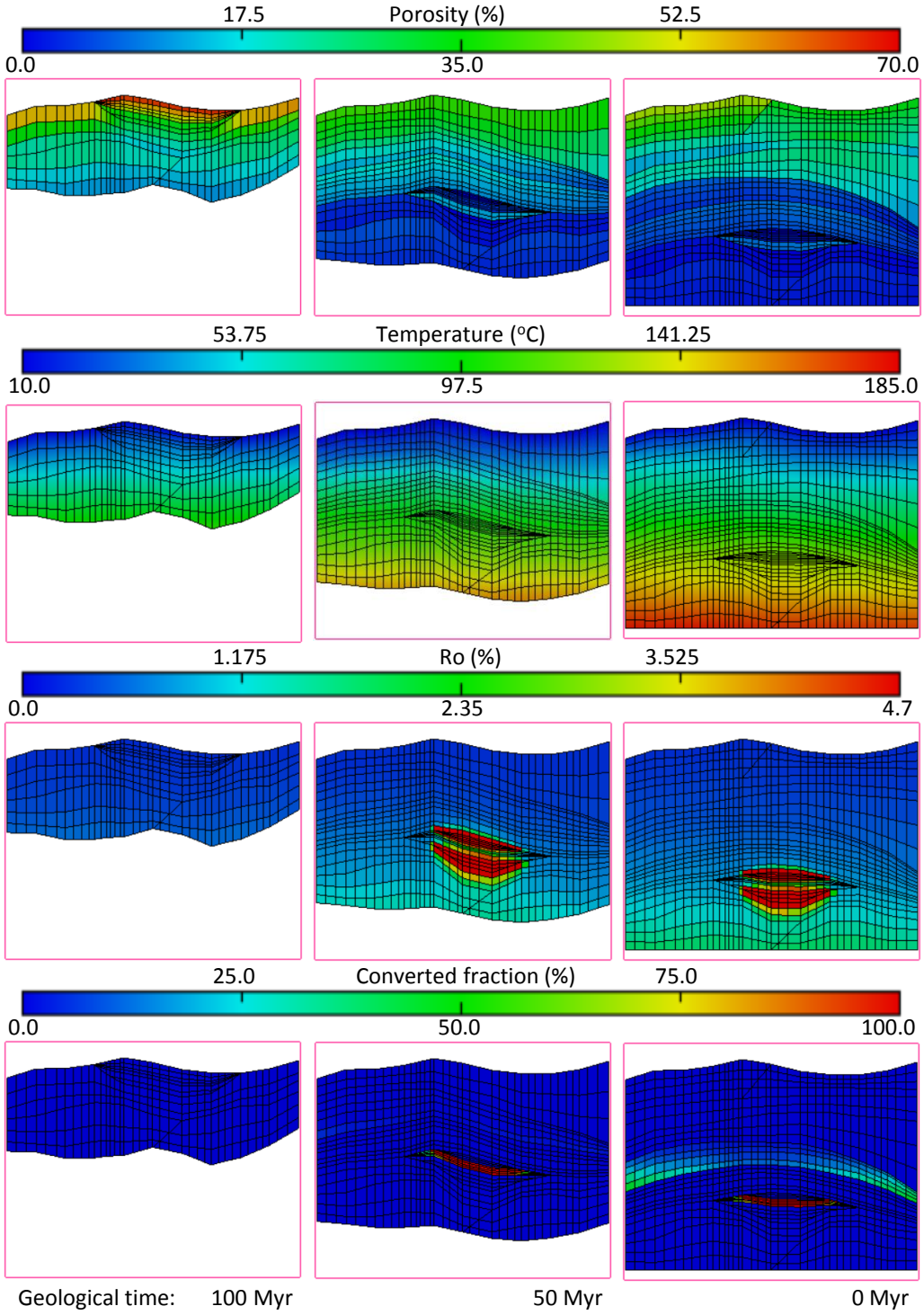


Figure 4: Basin analysis results on selected time steps.

## ACKNOWLEDGMENT

The authors gratefully acknowledge support from BG E&P Brasil through the "Coupled Geomechanics" project at TecGraf Institute (PUC-Rio) and the strategic importance of the support given by ANP (Brazils National Oil, Natural Gas and Biofuels Agency) through the R&D levy regulation.

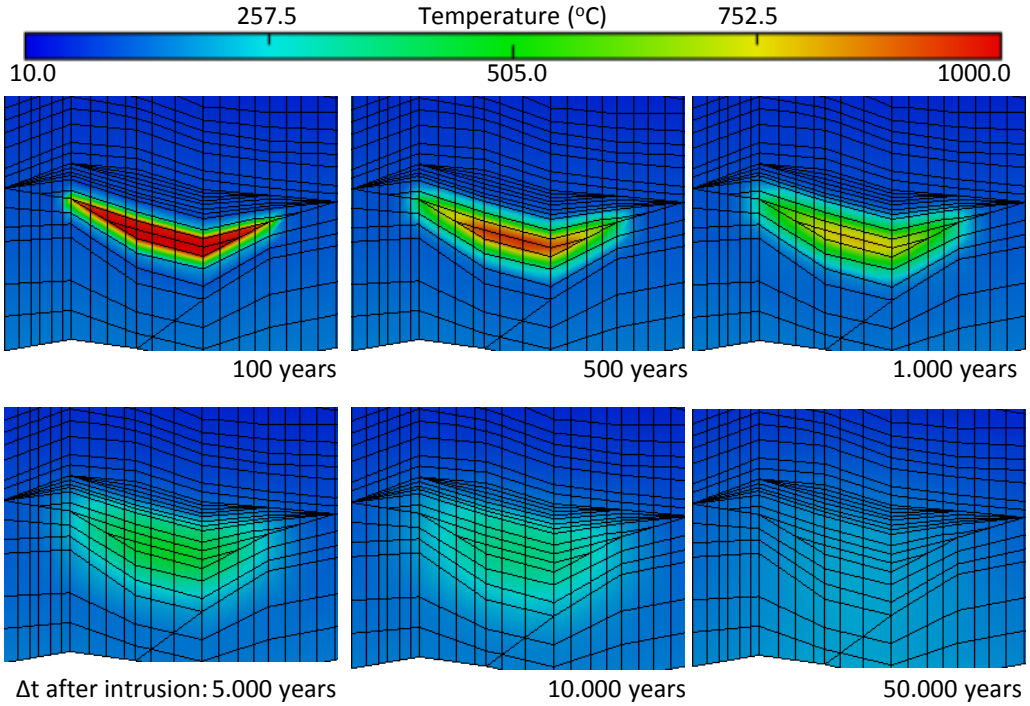


Figure 5: Temperature evolution in time steps following an intrusion event.

## REFERENCES

[1] J. Peir, S. Sherwin, Finite Difference, Finite Element and Finite Volume Methods for Partial Differential Equations. *Handbook of Materials Modeling*, 2415–2446, 2005.

[2] R. Ierusalimschy, L. Figueiredo, W. Celes, LuaAn Extensible Extension Language. *Software: Practice and Experience*, Vol. 26, Num. 6, 635–652, 1996.

[3] R. Ierusalimschy, *Programming in Lua*. 2003.

[4] X. Jiao, G. Zheng, P. A. Alexander, M. T. Campbell, O. S. Lawlor, J. Norris, A. Haselbacher, M. T. Heath, A system integration framework for coupled multiphysics simulations. *Engineering with Computers*, Vol. 23, Num. 3-4, 293–309, 2006.

[5] R. W. Lewis, P. Nithiarasu, K. N. Seetharamu, *Fundamentals of the Finite Element Method for Heat and Fluid Flow*. John Wiley & Sons, 2004.

[6] T. Hantschel, A. I. Kauerauf, *Fundamentals of Basin and Petroleum Systems Modeling*. Springer, 2009.

[7] J. G. Sclater, P. A. F. Christie, Continental stretching: An explanation of the post-mid-cretaceous subsidence of the central north sea basin. *Journal of Geophysical Research*, Vol. 85, Num. B7, 3711–3739, 1980.

[8] M. Wangen, Modelling heat and fluid flow in sedimentary basins by the finite element method. *Int. J. Numer. Anal. Methods Geomech.*, Vol. 15, Num. 10, 705–733, 1991.

[9] R. L. Braun, A. K. Burnham, Analysis of chemical reaction kinetics using a distribution of activation energies and simpler models. *Energy & Fuels*, Vol. 1, Num. 2, 153–161, 1987.