PROOST, AN OPEN SOURCE FRAMEWORK FOR GEOHYDROLOGICAL RESEARCH AND MODELING

Luit Jan Slooten, IDAEA, 934095410, luitjan.slooten@gmail.com

1. Luit Jan Slooten, Spanish Reasearch Council CSIC, IDAEA environmental institute, Barcelona, Spain
2. Jesus Carrera, Spanish Reasearch Council CSIC, IDAEA environmental institute, Barcelona, Spain
3. Fransisco Batlle, Hydrological Model Hosting company, Barcelona, Spain

Our PRocess Oriented Optimization and Simulation Tool (Proost) is a C++ framework for geohydrological modelling with a focus on coupled phenomena such as reactive transport, hydromechanical coupling and multiphase flow. From the outset, it was designed with a double purpose: to be a research tool (a sandbox environment for new algorithms and numerical methods), and to be a modeling tool capable of dealing with real world problems. The combination of these two purposes will, hopefully, put state of the art methods faster in the hands of end-users. It is soon to be released as a open source project and hence we welcome users and developers from the hydrological community to use it, test and contribute to it.
For open source projects, it is especially important that the code is well structured, clear and easy to expand upon. For end users on the other hand it is important that the code has a good performance and is sufficiently flexible.
In Proost, we attempted to achieve a clear structure with a classical Object Oriented framework design, which means that the overall program flow is defined in terms of abstract classes. Specific applications can be built by writing specialization classes deriving from these abstract classes. As the program logic is contained in the abstract classes, is not necessary to know much details about the program flow when writing a specialization class. In the case of Proost, the abstract classes forming the framework represent mathematical, physical and hydrological concepts, such as a Fields class (for storing and manipulating spatial distributions of quantities such as temperature or permeability), a Phenomenon class (representing balance equations) , a Solver class (representing systems of coupled balance equations), a Process class (representing physical or chemical processes that contribute to a balance equation, such as diffusion) and more.
In terms of performance, we have found that the Object Oriented methodology is a double edged sword. Object instantiation is expensive, and the use of classes for basic data structures such as arrays (even when using the highly optimized STL containers) can increase running time significantly if used careless. On the other hand, the flexibility of the framework allows creating taylor-made and optimal solutions for specific cases at low development cost (for example, an explicit solver for linear flow problems)
The flexibility of Proost for end users lies in the fact that he or she can construct models by freely defining and combining instances of the classes mentioned earlier (fields, processes, solvers, etcetera) in input files. Testimony to this is the way the mathematical model formulation can be made to change in time and space. Within a natural system, it is common to find different physical and chemical processes in different parts of the domain. Using Proost's classes it is possible to assigning spatial definitions to the equations being solved. A class called “state variable and equation manager” takes care of assembling the different equations in a single matrix system to simultaneously solve them. This reduces calculation time and leads to enhanced convergence.