Highly parallel implementations of a DFN flow simulation code

Stefano Berrone, Politecnico di Torino, stefano.berrone@polito.it,  
Alessandro D’Auria, Politecnico di Torino, alessandro.dauria@polito.it,  
Stefano Scialò, Politecnico di Torino, stefano.scialo@polito.it,  
Fabio Vicini, Politecnico di Torino, fabio.vicini@polito.it

Key words: DFN, Parallel Codes, MPI-CUDA

Abstract

The focus of the work is on efficient flow simulations in fractured media following the Discrete Fracture Network (DFN) framework. The medium is modeled as a rock matrix crossed by a network of polygonal fractures. Fractures intersect each other along segments called traces. The Darcy law is used to model the flow on each fracture: flux exchange among fractures occurs through the traces, whereas the surrounding rock matrix is assumed to be impervious. At fractures intersections, suitable matching conditions are imposed, ensuring the hydraulic head continuity and the flux balance.

Known probabilistic distributions are used to generate the DFNs, sampling from such distributions fracture hydro-geological and geometrical parameters like position, dimensions, aspect ratio and orientation in space. The networks generated are likely to display a complex geometry, with several critical features, such as multi-scale distribution in trace lengths and traces on fractures forming very narrow angles, Figure 2. These geometrical complexities may cause severe problems in building a good quality mesh conforming to the traces. A second challenge in DFN simulations is related to the size of the computational domains. Networks of interest for practical applications might count up to several thousands of fractures: in these cases a parallel method with an efficient handling of computational resources is mandatory.

Using an optimization based approach proposed in [1], which has proven to be very robust with respect to severe geometrical configurations, many of the geometrical problems could be avoided, allowing for non-conforming meshes on the fractures, Figure 1. Moreover, with the mentioned optimization based formulation, the discrete problem can be tackled using an iterative gradient based method; the core of each iteration can be decoupled in several local small problems, whose solution requires few data on the trace segments of the intersecting fracture to be communicated.

We propose two parallel approaches, the first one based on general-purpose computing on Graphics Processing Units (GPU), the second approach based on distributed computing using the Message Passing Interface (MPI) library. Both parallel algorithms are written in C++ language, with a standard Finite Element (FE) implementation. Other different discretization strategies, such as eXtended Finite Elements (XFE) and Virtual Elements (VE), can be used as well as a basis for the discretization; their implementation is under development.

In the GPU approach all the local systems are gathered together to form a single large system in order to optimize the use of all cores in the graphic unit. The cuSparse and cuBlas (NVIDIA® CUDA) libraries are used to perform all the arithmetic operations. In the distributed memory approach, fractures are split among the available processes according to a balanced partitioning. A Master-Slaves topology is used, which minimizes the number of communications, since only a moderate number of degrees of freedom, related to the traces, need to be communicated at each iteration.

Special care is devoted to shadowing the communication time in both the approaches, obtaining good scalability performances. A mixed approach MPI-CUDA is under construction.

References


Figure 1: Non conforming mesh on fracture intersections

Figure 2: DFN with 8000 fractures and 430000 traces