Multi-parametric intensive stochastic simulations for hydrogeology on a computational grid

J. Erhel, J-R. de Dreuzy, E. Bresciani

Parallel CFD conference, Lyon, May 2008

Contents

Stochastic models

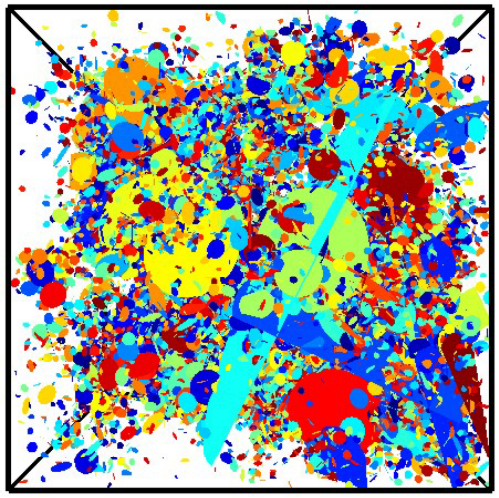
Porous geological media
fractured geological media

Spatial heterogeneity

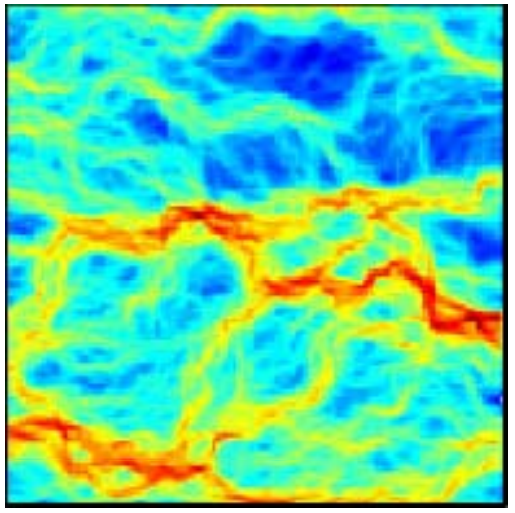
Lack of observations

Stochastic models of flow and solute transport

-random velocity field
-random solute transfer time and dispersivity



3D Discrete Fracture Network



Flow in highly heterogeneous porous medium

Uncertainty Quantification methods

Objective: statistic description of the outputs

Two types: non intrusive or intrusive UQ method

A non intrusive method : Monte Carlo simulations

=> Realization of N random simulations

=> Estimations of the first statistical moments (mean and standard deviation) of the output results by the empirical estimators

=> Generic method for any application

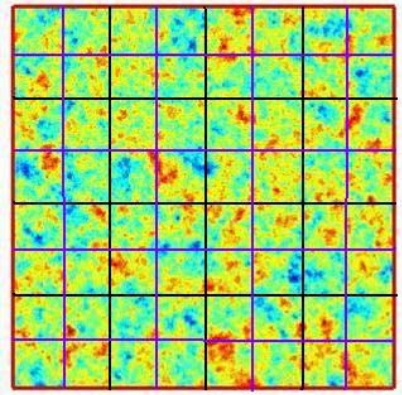
=> Easy to parallelize

=> But a large number of simulations

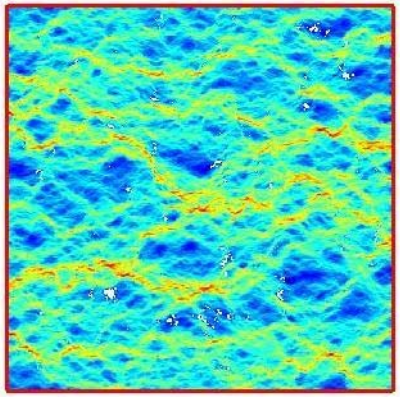
Monte-Carlo simulations

For $j=1,\ldots,N_s$

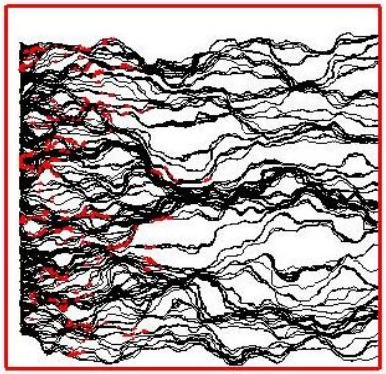
generate permeability field $K(\omega_j,x)$ using a regular mesh



Compute $V(\omega_j,x)$using a finite volume method



Compute $D(\omega_j,t)$using a random walker method



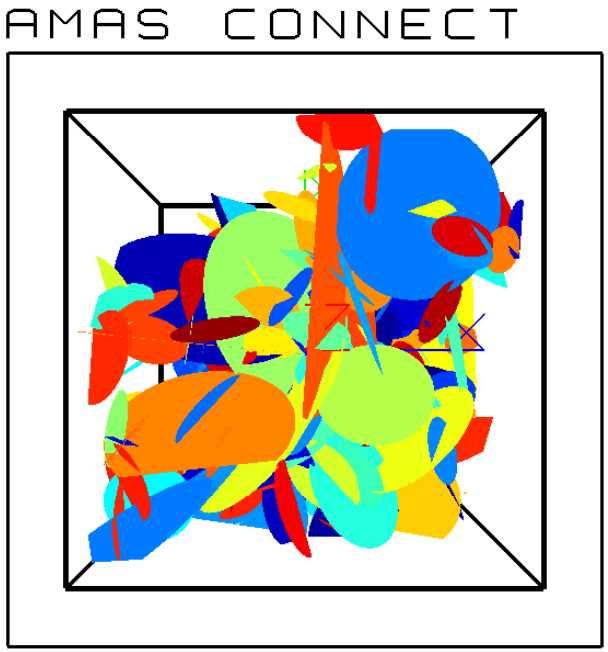
End For

$$D(t) \simeq 1/N_s \sum_j D(\omega_j,t)$$

Monte-Carlo simulations

For $j=1,\ldots,N_s$

generate random
discrete fracture network
and permeability field



Compute $V(\omega_j,x)$using
A mixed finite element
method

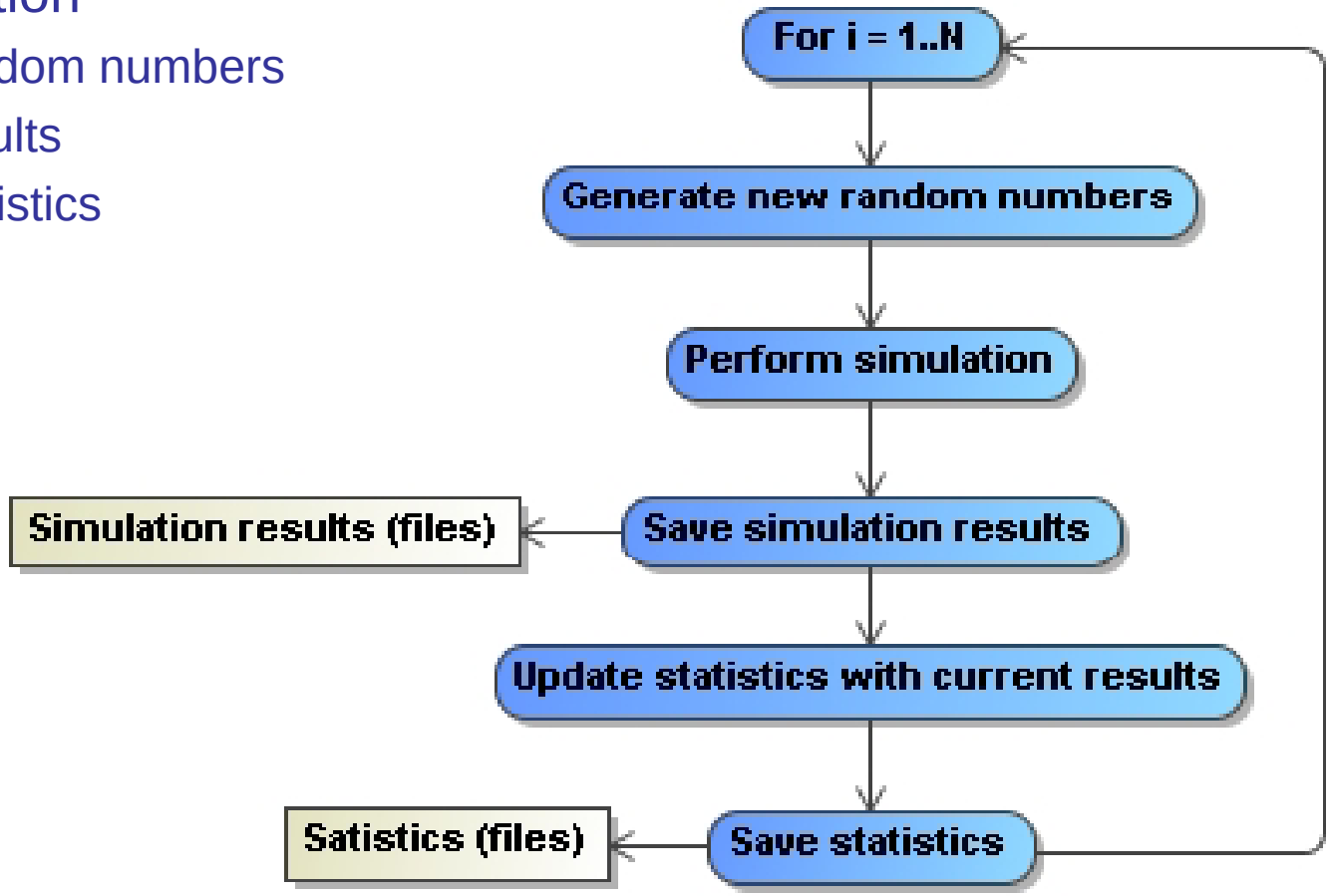
End For

$$V(x) \simeq 1/N_s \sum_j V(\omega_j,x)$$

A generic implementation

Generic implementation

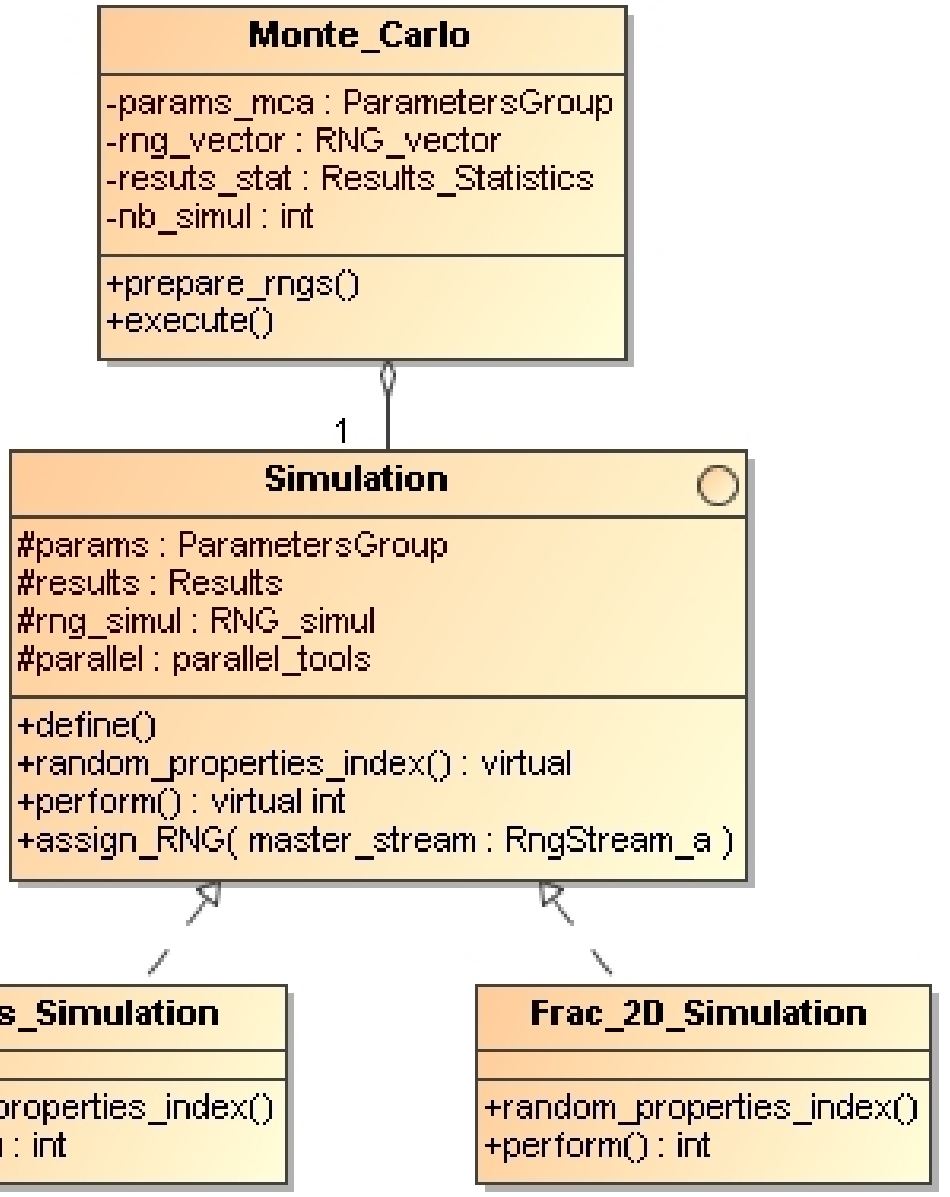
- To generate random numbers
- To structure results
- To compute statistics
- To store outputs



A generic interface

Simulation virtual class

- Used by Monte Carlo
- Implemented by the application
- Generic parameters structure
- Generic results structure
- Only two functions to overload



Specific random properties

Specific simulation

Continuing a Monte Carlo loop

Possibility of adding simulations to existent statistics

- Fault-tolerant feature
- Better usage of resources

Implementation

- Parameters
- Reload of statistics
  - Read the existing statistics
  - Read the number of already performed simulations
- Manage random number generators

Two-level parallelism

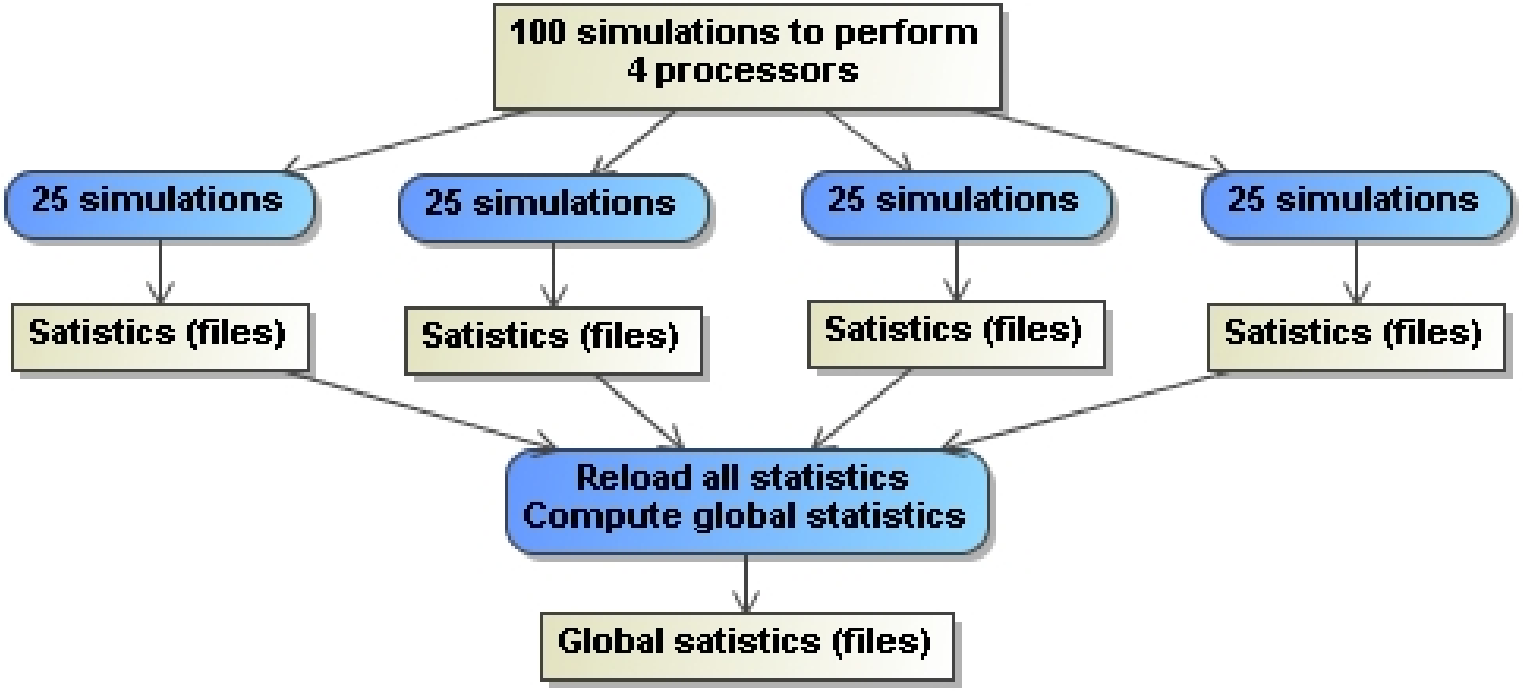
Parallel simulations

- Subdomain decomposition
- Parallel sparse linear solver for flux computation
- Parallel random walker for transport computation
- Programming model based on C++ and MPI

Parallel Monte-Carlo run

- Independent simulations
- Manage random number generation
- Programming model based on C++ and MPI

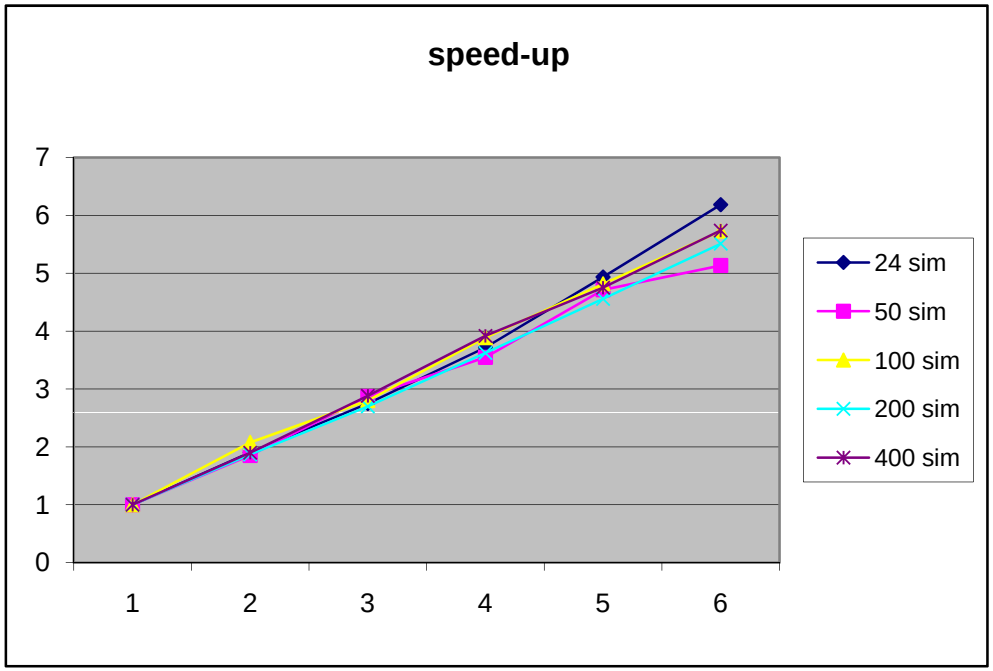
Two-level parallelism



Example: 1 domain and 1 processor for one simulation

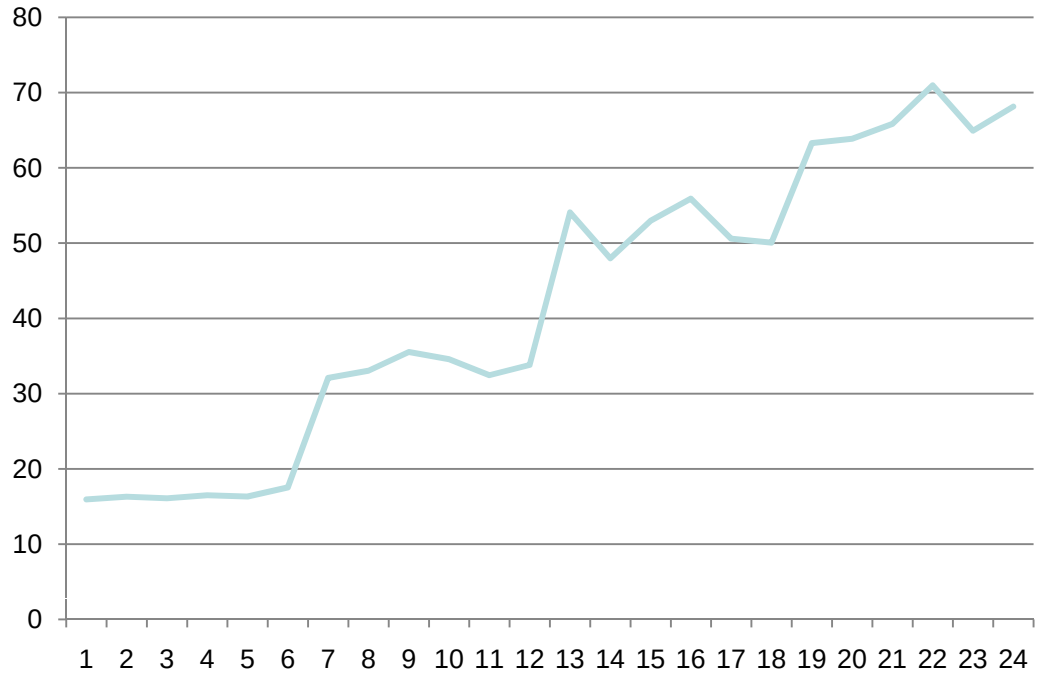
Parallel Monte-Carlo results

- Cluster of nodes with a Myrinet network
- Each node is one-core bi-processor, with 2Go memory
- Monte-Carlo run of flow and transport simulations
- Computational domain of size 1024x1024



Parallel Monte-Carlo results

Parallel simulations on 6 nodes
CPU time for each simulation



Two-level parallelism results

Global parallelism on a cluster of 4 nodes (one-core bi-processors)

CPU time for three configurations

- 2 subdomains and 4 parallel runs: 129 s
- 4 subdomains and 2 parallel runs: 199 s
- 8 subdomains and 1 run: 299 s

Better performances of parallel Monte-Carlo than parallel simulation

The critical resource is the memory

Use the minimal number of subdomains fitting in the memory

Then distribute several simulations

Conclusion

A generic parallel Monte Carlo method

- Usable by any application
- A convenient and easy to implement interface
- Continue facilities
- Parallel run of parallel simulations

Perspectives

- Distribution of Monte Carlo on a grid
- Improvement of Monte Carlo method
- More powerful non intrusive UQ methods