# An Efficient Parallel Particle Tracker for Advection-Diffusion Simulations in Heterogeneous Porous Media

Anthony Beaudoin[1], Jean-Raynald de Dreuzy[2], and Jocelyne Erhel[3]

[1] University of Le Havre, France
anthony.beaudoin@univ-lehavre.fr
[2] Geosciences Rennes, UMR CNRS 6118, France
Jean-Raynald.de-Dreuzy@univ-rennes1.fr
[3] Inria Rennes, France
Jocelyne.Erhel@irisa.fr
http://hydrolab.univ-rennes1.fr/

**Abstract.** The heterogeneity of natural geological formations has a major impact in the contamination of groundwater by migration of pollutants. In order to get an asymptotic behavior of the solute dispersion, numerical simultations require large scale computations. We have developed a fully parallel software, where the transport model is an original parallel particke tracker. Our performance results on a distributed memory parallel architecture show the efficiency of our algorithm, for the whole range of geological parameters studied.

**Keywords:** Heterogeneous porous media, advection, diffusion, particle tracker, cluster, distributed memory, speed-up.

## 1 Introduction

Numerical modeling is an important key for the management and remediation of groundwater resources. Several field experiments have showed that the natural geological formations are highly heterogeneous, leading to preferential flow paths and stagnant regions. The contaminant migration is strongly influenced by these irregular velocity distributions.

In order to account for the limited knowledge of the geological characteristics and for the natural heterogeneity, stochastic approaches have been developed [1]. Here we study the case of a lognormal permeability field with an exponential correlation function. The objective is to compute the dispersion coefficients as functions of time and to find out the asymptotic values of the dispersion coefficients [2]. We use numerical models to compute the velocity field over large spatial domains and to simulate solute transport over large temporal scales [1]. This approach must overcome two main difficulties, memory size and runtime, in order to solve very large linear systems and to simulate over a large number of timesteps. High performance computing is thus necessary to carry out these large scale simulations [1].

A.-M. Kermarrec, L. Bougé, and T. Priol (Eds.): Euro-Par 2007, LNCS 4641, pp. 705–714, 2007.

We have developed a fully parallel software, which is object-oriented, with a modular approach and well-defined interfaces [3]. The permeability field is generated by means of parallel Fast Fourier Transforms and the velocity field is computed by solving a large sparse linear system, using a parallel multigrid solver of the numerical library Hypre [4]. In this paper, we describe a parallel algorithm for advection-diffusion modeling and its performances on clusters of processors. Transport is simulated by a particle tracker algorithm [5], well suited for pure advection and advection-dominated transport processes, because it does not introduce spurious numerical diffusions. Our parallel algorithm takes advantage of a subdomain decomposition and of the non interaction between the particles. This hybrid parallel strategy is quite original, since most of previous work uses either one or the other [6],[7],[8]. We perform many numerical experiments, in order to study the effects on parallel performances of the hydraulic parameters, mainly the size of the computational domain, the heterogeneity of the permeability field and the ratio of advection to diffusion. We also analyze scalability issues by varying the number of processors used. Our results show that our parallel particle tracker is quite robust and efficient in the whole range of interesting values. Using a cluster with 32 processors, we could run simulations with spatial and temporal scales never achieved so far. This allowed us to determine with no ambiguity the asymptotic dispersion coefficients [9].

## 2 Numerical Model for Advection-Diffusion

### 2.1 Physical Model

In this paper, we consider a 2D problem where the porous medium defined by a rectangle with dimensions $L_x$ and $L_y$. As the mean flow direction is the $x$ axis, $L_x$ and $L_y$ are respectively longitudinal and transversal to the mean flow direction. The porous medium, assumed isotropic, is characterized by a random hydraulic conductivity field $K$, which follows a stationary log-normal probability distribution $Y = ln(K)$, defined by a mean $m$ and a covariance function $C$ given by

$$C(r) = \sigma^2 \exp(-\frac{|r|}{\lambda}),$$

where $\sigma^2$ is the variance of the log hydraulic conductivity, $|r|$ represents the separation distance between two points and $\lambda$ denotes the correlation length scale. The length $\lambda$ is typically in the range $[0.1m, 100m]$ and the variance $\sigma^2$ is in the interval $[0, 7]$. These two ranges encompass most of the generally studied values.

Classical laws governing the steady flow in a porous medium are mass conservation and Darcy law

$$\epsilon V = -K\nabla h, \nabla.V = 0$$

where $\epsilon$ is the porosity, $V$ is the Darcy velocity and $h$ is the hydraulic head. Boundary conditions are homogeneous Neumann on upper and lower sides and

Dirichlet $h = 0$ on left side, Dirichlet $h = 1$ on right side. The system should not be too much elongated in order to avoid border effects. We denote by $U$ the mean velocity norm.

An inert solute is injected in the porous medium and transported by advection and dispersion. This type of solute migration is described by the advection - dispersion equation

$$\frac{\partial(\epsilon c)}{\partial t} + \nabla.(\epsilon c V) - \nabla.(\epsilon D \nabla c) = 0$$

where $c$ is the solute concentration and $D$ is the dynamic dispersion tensor. Here, we assume a constant porosity $\epsilon = 1$ and we consider only molecular diffusion, assumed homogeneous and isotropic, with a tensor

$$D = D_m I$$

where $D_m$ is the molecular diffusion coefficient and $I$ denotes the unity tensor.

Boundary conditions and an initial condition complete the transport equation. The initial condition at $t = 0$ is the injection of the solute. In order to overcome the border effects, the inert solute is injected at a given distance of the left side. The ratio of diffusion to advection is measured by the Peclet number defined by

$$Pe = \frac{\lambda U}{D_m}$$

with a typical range of $[100, \infty]$, where $\infty$ means pure advection. This range covers advection-dominated transport models, whereas diffusion-dominated models require smaller values.

### 2.2 Numerical Flow

The flow equations are discretized on a regular grid using a classical Finite Volume scheme. Let $\Delta_x$ and $\Delta_y$ the mesh sizes in each direction. The number of cells in each direction is given by $n_x = L_x/\Delta_x$ and $n_y = L_y/\Delta_y$, with a total number $N = n_x n_y$. The linear system $Ax = b$, obtained from the discretization of flow equations, is characterized by a sparse symmetric positive definite matrix $A$ of order $N$, with a pentadiagonal structure. The solution $x$ is the discrete hydraulic head and the right-hand side $b$ represents Dirichlet boundary conditions. We solve this sparse linear system by an algebraic multigrid method, with the procedure Boomer-AMG of the Hypre library [4] and we get very good parallel performances.

### 2.3 Numerical Transport: Particle Tracker

The inert solute transport is simulated by a particle tracker algorithm whose key advantages for this study are the absence of numerical diffusion and the good performances. In this Lagrangian framework, the solution of transport

equation consists in evaluating the trajectory of particles [10],[2]. These paths are represented by the Stochastic Differential Equation

$$dX = Vdt + \sqrt{2D_m}dW$$

where $X$ is the position of the particle and $dW$ is a Brownian motion process.

Using a first-order explicit scheme for the discretization of transport equation, between a time $t$ and a time $t + \Delta t$, a particle moves at each timestep according to the equation

$$X(t + \Delta t) = X(t) + V\Delta t + \sqrt{2D_m}Zw\sqrt{\Delta t}$$

where $Z$ is a random number drawn from a Gaussian distribution of mean 0 and variance 1 and $w$ is a unitary vector with uniformly distributed orientation. The timestep $\Delta t$ evolves along the particle path according to the velocity magnitude of the crossed cells. More precisely, the timestep is either proportional to the diffusion time necessary to cross the cell or to the local advection time, which is equal to the cell size divided by the minimum of the velocities computed on the cell borders in both directions. Within each cell, particle velocities $V$ are obtained by a bilinear interpolation of the boundary velocities, because it is the sole interpolation method that respects mass conservation. No-flux boundaries are maintained by bouncing particles off of the boundary. At the free outflow boundary (the right side), particles are allowed to naturally flow out of the domain. The algorithm finishes when all the particles have quit.

### 2.4 Macro-dispersion Analysis

The main objective of our study is to evaluate the temporal macro-dispersions $D_x(t)$ and $D_y(t)$ in both directions and their asymptotic value when $t$ becomes infinite [1]. Let $(x_m(t), y_m(t))$ the coordinates of the computed trajectory of a particle and let $N_p$ the number of particles. As long as the particles are in the domain, the total mass is constant. The center of mass of the solute distribution in the longitudinal direction is approximated by the first moment

$$\overline{x}(t) = \frac{1}{N_p}\sum_{m=1}^{N_p} x_m(t)$$

and the spread of mass around $\overline{x}$ is approximated by the second moment

$$S_x(t) = \frac{1}{N_p}\sum_{m=1}^{N_p} x_m(t)^2 - \overline{x}(t)^2.$$

Then the temporal longitudinal macro-dispersion is defined by

$$D_x(t) = \frac{dS_x}{dt}(t),$$

with time derivatives approximated by a first-order Taylor expansion. The transversal macro-dipsersion is defined in a similar way.

Asymptotic behavior can be obtained only at very large times. In order to keep all the particles in the domain during long simulated times, the dimensions $L_x$ and $L_y$ must be very large. Therefore, high performance computing is required to deal with these computationally intensive simulations. Also, memory requirements are very high and data must be spread from the beginning in a distributed memory architecture.

## 3 Parallel Particle Tracker

We have designed a parallel algorithm for parallel distributed memory architectures. We use a SPMD model of programmation, where all processors execute the same program (with conditional statements using the processor number). Communications are based on the message-passing interface provided by the MPI library. In the particle tracker currently used, the particles do not interact so each trajectory can be implemented in a parallel independent process. This is a classical efficient approach, see for example [6]. However, our objective is to run simulations on very large computational domains so we need to distribute data in order to use all memory resources. Therefore, we choose to use a domain decomposition approach, where the velocity field is distributed among processors. Actually, we use the domain decomposition and data distribution arising from permeability generation and flow computation [3]. This methodology is also classical, see for example [7]. Each subdomain is assigned to one processor, which computes the trajectory of the particles inside the subdomain owned by it. Communications with processors owning neighboring subdomains must occur when particles exit or enter the subdomain. This is achieved by means of SEND and RECV operations. We choose non blocking SEND operations so that each processor can first proceed with sending then with receiving. The algorithm must ensure a correct ending, when all particles have left the computational domain [8]. It must also guarantee that the sending buffers are empty when new SEND operations are executed. For these reasons, we choose to add a global synchronisation point after the exchanges of particles. This point allows defining a global state where each processor knows the total number of particles still in the domain. By the way, it guarantees that all RECV operations have been successful and that the algorithm may execute new SEND operations. Thus neither deadlock nor starvation may occur and the algorithm terminates safely.

A difficulty is to balance the computational load between the processors. A specificity of our application is the initial localization of the particles. Indeed, at their initial stage, all particles are on the left part of the domain. If the master processors owning the subdomains with the initial particles launched all particles, then all other processors would be idle, waiting for particles entering their domain. For example, if we divide the domain into slices, most of the processors have two neighbors, on the left and on the right. If the master processor launched all particles together, parallelism would not be efficient at all since only one processor would be active while others would be idle.

Therefore, we have designed a parallel algorithm well-suited for our particular case. We take advantage of the independence of the particles and launch

them by bunches. The idea is to initiate a pipeline of particles from the left to the right. At each global synchronisation point, the master processors inject a new bunch of particles, until all particles have been injected into the domain. All processors track the particles in their subdomain, then exchange particles on their boundaries, finally synchronize globally to define the total number of particles in activity. Let us consider as an example the case of a constant velocity field and a transport by pure advection, with a slice decomposition. Then after initiating the pipeline, each processor owns a bunch of particles between two synchronisation points, until draining the pipeline. Thus computations are well-balanced. In the general case, it is not so easy to predict the computational load. Two main factors will impact the parallel algorithm: the heterogeneity and the diffusion. Heterogeneity is measured by the variance of the permeability field, whereas diffusion is measured by the Peclet number.

## 4 Numerical Results

### 4.1 Numerical Experiments

We have developed a fully parallel object-oriented software which provides a generic platform to run Monte-Carlo numerical simulations of flow and transport in highly heterogeneous porous media. The software is divided into three main modules, respectively dedicated to the generation of the permeability field, the computation of flow and velocity, the computation of transport and dispersion coefficents. This modularity allows a great flexibility and portability. The software is written in C++ and can be implemented on machines with Unix, Linux or Windows systems. Graphical functions are written with OpenGL and parallel programming relies on the MPI library. The software integrates open-source components such as sparse linear solvers for flow computation.

In this paper, we show the results of various experiments with the parallel particle tracker. All tests are performed on a SUN cluster composed of two nodes of 32 computers each. Each computer is a 2.2 Ghz AMD Opteron bi-processor with 2 Go of RAM. Inside each node, computers are interconnected by a Gigabit Ethernet Network Interface, and the two nodes are interconnected by a Gigabit Ethernet switch (CISCO 3750). This cluster is a component of the Grid'5000 computing resource installed at INRIA in Rennes. Experiments are done with a variable number of processors $P$.

The different parameters and their values are summarized below:

$$\begin{cases} \Delta x = \Delta y = 1, \lambda = 10, N_p = 1000, \\ L_x = L_y \in \{512, 1024, 4096, 8192\}, \\ \sigma \in \{0.5, 1, 2, 2.5, 3\}, \\ Pe \in \{10, 100, 1000, \infty\}, \\ P \in \{1, 2, 4, 8, 16, 32, 64\}. \end{cases}$$

We have done other experiments with different values of $\lambda$ and the choice $\lambda = 10$ is a good tradeoff between the accuracy of the numerical model and the

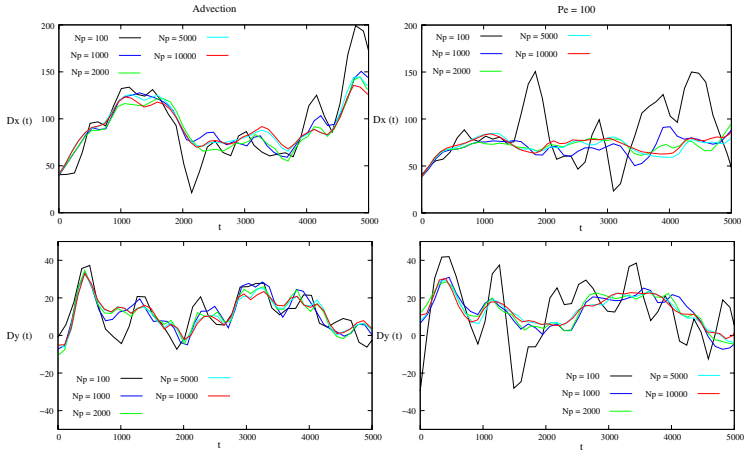


**Fig. 1.** Convergence analysis: longitudinal and transversal dispersions for varying number of particles $N_p$. Pure advection $Pe = \infty$ (left) and advection-diffusion $Pe = 100$ (right); heterogeneous case $\sigma = 2.5$. Large domain size $L_x = L_y = 8192$.

computational cost. We analyse and comment below with more details the choice of the number of particles $N_p$. The values $\sigma = 3$ and $Pe = 10$ are out of scope in geological surveys but we include them in order to check the behavior of our algorithms with extreme values of the parameters. All other values have a strong interest for geological study. The dimension $L_x = 8192$ of the domain is the target value, in order to get with no ambiguity the asymptotic values of the longitudinal and transversal dispersion coefficients. Other dimensions are useful to analyze the complexity and the scalability of our parallel particle tracker.

### 4.2 Convergence Analysis

Clearly, the CPU time of transport computations is proportional to the number of particles $N_p$ so that this number must be kept low, still ensuring a good convergence of the dispersion coefficients. Therefore, we study the convergence of the dispersion coefficients with the number of particles, in order to choose an optimal number. In Figure 1, we plot the temporal dispersions $D_x(t)$ and $D_y(t)$ for various values of $N_p$. We find out that the value $N_p = 1000$ is a good tradeoff between convergence and performance. We show only the cases $Pe = \infty$ and $Pe = 100$ with $\sigma = 2.5$ and with $L_x = L_y = 8192$, but other results are similar.

### 4.3 Performance Analysis

In Figure 2, we plot the CPU time versus the number of processors for two Peclet numbers $Pe = \infty, Pe = 100$ and for two domain sizes $L_x = 1024, L_x = 4096$. In the pure advection case, we can observe good performances for any value of the heterogeneity parameter $\sigma$. The CPU time increases slightly with $\sigma$ and

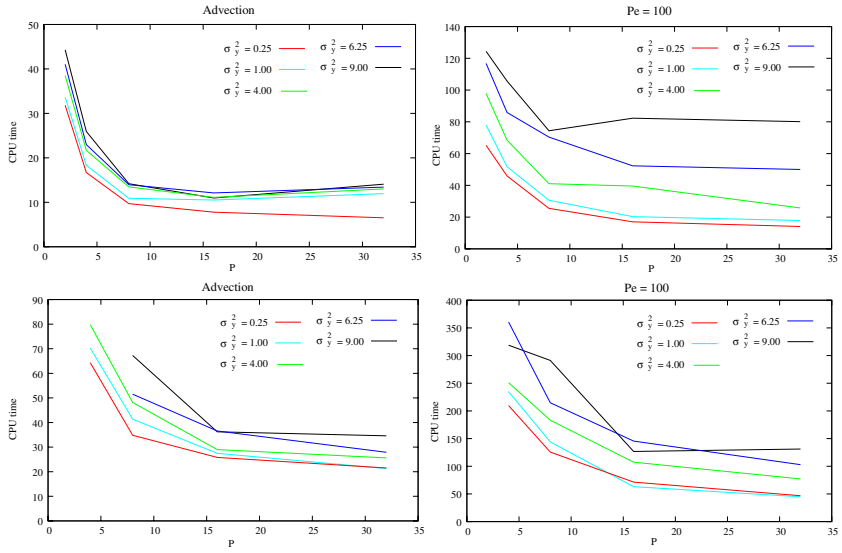


**Fig. 2.** Performance analysis: CPU time versus the number of processors $P$. Pure advection $Pe = \infty$ (left) and advection-diffusion $Pe = 100$ (right); small size $L_x = L_y = 1024$ (top) and large size $L_x = L_y = 4096$ (bottom).

performances slightly deteriorate for large $\sigma$. From the small to large domain size, the performances improve clearly. The CPU time is significantly reduced from 8 to 16 processors with the large size $L_x = 4096$. In the advection-diffusion case with $Pe = 100$, the trends are similar. However, CPU times are much higher, because the particle tracker includes a random diffusion motion, with generation of random numbers at each timestep. Moreover, the simulated time is larger because the particles stay longer in the domain before reaching the exit at the right side. Performances are not so good with large values of the heterogeneity and the small domain size. A possible reason is some imbalance of work in the parallel algorithm, inducing idle time for some processors. This effect almost disappears for the large domain size and the CPU time decreases from 16 to 32 processors for any kind of heterogeneity.

In order to study scalability issues, in Figure 3, we plot the speed-up versus the number of processors. Because large domain sizes cannot fit in the memory of 2 processors, we compute the speed-up from $P = 4$ with a speed-up arbitrarily equal to 4 for $P = 4$. We conclude that our parallel particle tracker is quite efficient and is reasonably scalable.

### 4.4 Results for Large-Scale Simulations

The dispersion coefficients and their asymptotic values are computed on a large domain size, in order to avoid border effects and to get large simulated times. All the simulations are now done with this large size $L_x = L_y = 8192$ and with

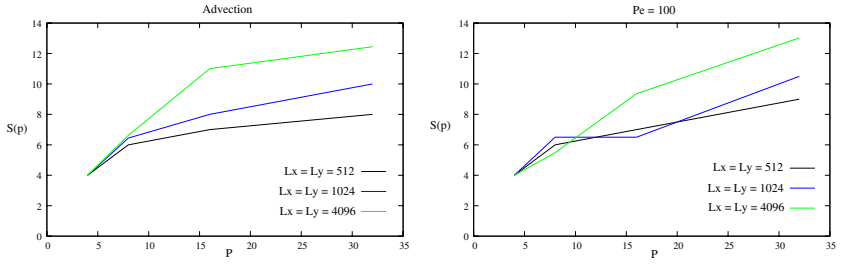


**Fig. 3.** Scalability analysis: speed-up versus the number of processors $P$. Pure advection $Pe = \infty$ (left) and advection-diffusion $Pe = 100$ (right); heterogeneous case $\sigma = 2$.

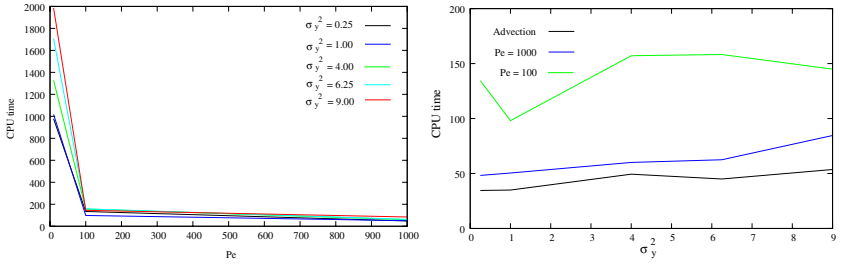


**Fig. 4.** Impact of diffusion (left) and of heterogeneity (right). Dimension $L_x = L_y = 8192$. Number of processors $P = 32$.

$P = 32$ processors. Figure 4, left, plots the CPU time versus the Peclet number for the different kinds of the heterogeneity. Down to $Pe = 100$, CPU times are not too high. Clearly, the CPU time blows up for the small value $Pe = 10$, indicating an exponential-like behavior. Thus, another numerical model should probably be used for diffusion-dominated transport equations. Indeed, a method such as a Finite Volume method could be relevant since numerical diffusion would not be such an issue. In the range of interest of advection-dominated equations, the particle tracker remains a good choice, efficient for both cost and accuracy. Figure 4, right, plots the CPU time versus the heterogeneity coefficient $\sigma$ for the different Peclet numbers. The CPU time increases slightly but the largest time is less than twice the smallest time. Therefore, our particle tracker is very well-suited and efficient for all values of interest for the heterogeneity.

## 5 Conclusion

We have developed a fully parallel software for simulating flow and solute transport in highly heterogeneous porous media. In this paper, we describe an original parallel particle tracker, which relies on both the independence of particles and a subdomain decomposition. The performances for a whole range of parameters

show that our algorithm is efficient and robust. We could run simulations at very large spatial and temporal scales and get with no ambiguity the asymptotic dispersion coefficients. We plan to extend this work to 3D simulations.

**Acknowledgments.** This work was supported by Grid'5000 grants for executing simulations on the grid at Irisa in Rennes.

## References

1. Gelhar, L.: Stochastic Subsurface Hydrology, Engelwood Cliffs, New Jersey (1993)
2. Tompson, A., Gelhar, L.: Numerical simulation of solute transport in three-dimensional, randomly heterogeneous porous media. Water Resources Research 26(10), 2541–2562 (1990)
3. Beaudoin, A., Erhel, J., de Dreuzy, J.R.: A comparison between a direct and a multigrid sparse linear solvers for highly heterogeneous flux computations. In: Eccomas CFD 2006, vol. CD (2006)
4. Falgout, R.D., Jones, J.E., Yang, U.M.: Pursuing scalability for hypre's conceptual interfaces. ACM Trans. Math. Softw. 31(3), 326–350 (2005)
5. Delay, F., Ackerer, P., Danquigny, C.: Solution of solute transport in porous or fractured formations by random walk particle tracking: a review. Vadose Zone Journal 4, 360–379 (2005)
6. Huber, E., Spivakovskaya, D., Lin, H., Heemink, A.: The parallel implementation of forward-reverse estimator. In: Wesseling, P., Onate, E., Periaux, J. (eds.) ECCOMAS CFD, TU Delft, The Netherlands (2006)
7. Cheng, J.R., Plassmann, P.: The accuracy and performance of parallel in-element particle tracking methods. In: Proceedings of the Tenth SIAM Conference on Parallel Processing for Scientific Computing, Portsmouth, VA, pp. 252–261 (2001)
8. Kaludercic, B.: Parallelisation of the lagrangian model in a mixed eulerian-lagrangian cfd algorithm. Journal of Parallel and Distributed Computing 64, 277–284 (2004)
9. de Dreuzy, J.R., Beaudoin, A., Erhel, J.: Asymptotic dispersion in 2D heterogeneous porous media determined by parallel numerical simulations. In: INRIA (submitted to Water Resource Research, in revision) (preprint 2007)
10. LaBolle, E., Quastel, J., Fogg, G., Gravner, J.: Diffusion processes in composite porous media and their numerical integration by random walks: generalized stochastic differential equations with discontinuous coefficients. Water Resources Research 36(3), 651–662 (2000)