

# Effectiveness of Pump–and–Treat Remediation in Heterogeneous Groundwater Aquifers: A computational investigation using the Intel Paragon

G. Mahinthakumar

*Center for Computational Sciences  
Oak Ridge National Laboratory  
Oak Ridge, TN 37931  
e-mail: kumar@ornl.gov*

## Abstract

In this paper we investigate the importance of fine scale aquifer heterogeneity in pump–and–treat remediation through a series of large three–dimensional high resolution numerical simulations performed on the Intel Paragon. Parallelization strategies and performance issues are also discussed. The groundwater flow problem is solved using the Galerkin finite–element method using eight–node brick elements and a conjugate gradient matrix solver. The solute transport problem is solved using a particle tracking scheme. Most of the practical simulations consisting of 800,000 finite elements were performed on 32–node partitions of the Intel Paragon XPS/35. We report single–node performances of 10–15 Mflops for the groundwater flow problem with communication overheads less than 5% for most simulations. For the solute transport problem the communication overheads were much higher resulting in poorer performances. The major scientific conclusion from this work is that geologic heterogeneity has a profound impact on remediation times. In the extreme cases of heterogeneity remediation was almost impossible under practical pumping durations.

## 1 Introduction

Groundwater contamination from industrial, municipal, and agricultural sources is widely recognized as a major environmental problem. Although there is ongoing debate regarding the extent and severity of contamination, there is no disagreement over the staggering economic costs incurred in monitoring and rehabilitating polluted groundwater supplies. One of the most common technologies for aquifer remediation is to pump contaminated water to the surface for treatment [1]. This is commonly referred to as pump–and–treat (PAT) remediation. Under ideal conditions, contami-

nants will be in the dissolved phase distributed throughout the pore space of a relatively homogeneous and highly permeable aquifer; in this case, pumping will effectively remove the pollutants. Although the PAT option is often selected as the technology to achieve aquifer restoration, there is an increasingly large body of evidence suggesting that this technology is not always effective. This is primarily because ideal conditions are rarely approached in realistic aquifers. One of the principal nonidealities affecting the performance of PAT systems is geologic heterogeneity. In this paper we investigate the effect of this nonideality through a series of high resolution numerical simulations performed on the Intel Paragon supercomputer.

## 2 Governing Equations

The principal equations involved in groundwater solute transport problems can be divided into two: the groundwater flow equation, and the solute transport equation.

### 2.1 Groundwater Flow Equation

The three–dimensional form of the steady–state saturated groundwater flow equation used in this work is given by [2]

$$\frac{\partial}{\partial x} \left( K(x, y, z) \frac{\partial h}{\partial x} \right) + \frac{\partial}{\partial y} \left( K(x, y, z) \frac{\partial h}{\partial y} \right) + \frac{\partial}{\partial z} \left( K(x, y, z) \frac{\partial h}{\partial z} \right) = q \quad (1)$$

where  $K(x, y, z)$  is the hydraulic conductivity field,  $h$  is the head field, and  $q$  represents the source/sink terms coming from injection/pumping wells. Once the hydraulic head field  $h$  is computed from equation (1), we compute the velocity field  $\mathbf{v}$  from Darcy’s law:

$$\mathbf{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = -\frac{K(x,y,z)}{\theta} \begin{bmatrix} \frac{\partial h}{\partial x} \\ \frac{\partial h}{\partial y} \\ \frac{\partial h}{\partial z} \end{bmatrix} \quad (2)$$

where  $v_x$ ,  $v_y$ ,  $v_z$  are the velocity components and  $\theta$  is the constant porosity.

## 2.2 Solute Transport Equation

The three-dimensional form of the convection dispersion equation used in this work is described by [2]

$$\begin{aligned} \frac{\partial c}{\partial t} = & D_{xx} \frac{\partial^2 c}{\partial x^2} + D_{xy} \frac{\partial^2 c}{\partial x \partial y} + D_{xz} \frac{\partial^2 c}{\partial x \partial z} \\ & + D_{yx} \frac{\partial^2 c}{\partial y \partial x} + D_{yy} \frac{\partial^2 c}{\partial y^2} + D_{yz} \frac{\partial^2 c}{\partial y \partial z} \\ & + D_{zx} \frac{\partial^2 c}{\partial z \partial x} + D_{zy} \frac{\partial^2 c}{\partial z \partial y} + D_{zz} \frac{\partial^2 c}{\partial z^2} \\ & -v_x \frac{\partial c}{\partial x} -v_y \frac{\partial c}{\partial y} -v_z \frac{\partial c}{\partial z} \end{aligned} \quad (3)$$

where  $c(x,y,z,t)$  is the contaminant concentration at location  $(x,y,z)$  at time  $t$ ,  $v_x$ ,  $v_y$ , and  $v_z$  are the components of the velocity vector  $\mathbf{v}$  at location  $(x,y,z)$  obtained from equation (1), and  $(D_{xx}, D_{yy}, D_{zz}, D_{xy}, D_{yz}, D_{zx})$  are the components of the dispersion tensor  $\mathbf{D}(\mathbf{v})$ .

## 3 Solution Technique

We employ two distinct approaches in solving these two equations. For the flow equation we use a field approach using finite elements, and for the transport equation we use a particle approach.

### 3.1 Groundwater Flow Problem

The flow equation is discretized using the Galerkin finite-element approach with eight node brick elements. The influence coefficient technique is used to evaluate the element matrices which gives us an option to chose either the finite-element or finite-difference approach using the same finite-element mesh data. However, in the results that are discussed in this paper we use only the finite element option. The resulting matrix equation is solved using a conjugate gradient matrix

solver using either diagonal or block-diagonal preconditioning. The global conductance matrix is a banded 27-diagonal matrix for the finite-element option since a logically rectangular grid is assumed. The pumping/extraction well used in this study in the remediation stage is modelled as a line source. The relative flux distribution along the nodes of the well line source is computed using an iterative procedure described in [3]. This procedure is based on the assumption that the drawdown at the well is proportional to the pumping rate. Each iteration in this procedure involves a solution of the flow equation although convergence is usually achieved within 2–3 iterations when single wells are involved. In other words we had to perform at least two matrix solves when the flow solution involved wells.

### 3.2 Solute Transport Problem

The solute transport equation is solved using the randomwalk particle tracking approach described in [4]. A particle tracking analog is derived for equation (3) as described in [5]. The resulting equation can be written as

$$\mathbf{X}_p(t + \Delta t) = \mathbf{X}_p(t) + \mathbf{v}(\mathbf{X}_p) \Delta t + \mathbf{B} \cdot \mathbf{Z} \sqrt{\Delta t} \quad (4)$$

where  $\mathbf{X}_p(t)$  is a 3D vector  $(x,y,z)$  coordinates) denoting the  $p$ th particle position at time  $t$ ,  $\Delta t$  is the time step,  $\mathbf{B}\mathbf{B}^T = 2\mathbf{D}$ ,  $\mathbf{Z}$  is a vector of three independent random numbers from normal distributions with mean zero and variance one, and  $\mathbf{v}(\mathbf{X}_p)$  is the velocity vector. An 8-point trilinear scheme is used for the velocity interpolation which takes up most of the computation time. Other major computational routines are for the generation of normally distributed random numbers  $\mathbf{Z}$  and for computing components of the tensor  $\mathbf{B}$  during each time step.

## 4 Simulation methodology

The simulations consist of the following basic tasks that are described in the following sections.

#### 4.1 Generation of a randomly heterogeneous hydraulic conductivity field

This is accomplished by a parallelized version of the serial turning bands code originally developed at the Lawrence Livermore National Laboratory by A.F. Tompson [6]. Turning bands relies on generating a large number randomly oriented one-dimensional line processes (called bands or rays) using spectral transforms which are then projected onto a three-dimensional field. The one-dimensional processes are generated such that the resulting 3D field is a lognormal field with a specified mean, variance, and correlation scales. The degree of heterogeneity is measured by the variance of the logarithmic of the field (denoted commonly by  $\sigma^2_Y$ ). A very simple loop decomposition was performed to parallelize this code on the Intel Paragon. Each processor will compute a portion of the field which are then written into a file. The only communication phase was in the final statistics computation routines which was handled by the Paragon system routines such as "gdsum", "gisum", "gcolx", and "gland". Since this parallelization is fairly trivial we do not go into any further details in this paper.

#### 4.2 Generation of the contaminant plume

The contaminant plume is formed by allowing the contaminant to leach from a single rectangular source into a naturally flowing groundwater aquifer as shown in Fig 1. To generate this plume numerically we first use the groundwater flow code which produces a velocity field corresponding to this scenario. Note that the hydraulic conductivity field that is used in the flow problem is a fully heterogeneous field produced by the turning bands code. The velocity field output from the flow code is then fed into the transport code which produces the concentration field for the final plume.

#### 4.3 Remediation of the contaminant plume

Remediation of the generated plume is performed by placing an extraction well at the centroid of the plume as shown in Fig 1. We assume that the contaminant source has been plugged and there is no further leaching during this stage. First we use the flow code to

produce a velocity field that corresponds to this scenario with the extraction well. This velocity field and the previously generated concentration field (contaminant plume) is then fed into the transport code which is used to track the changes in the plume as remediation progresses.

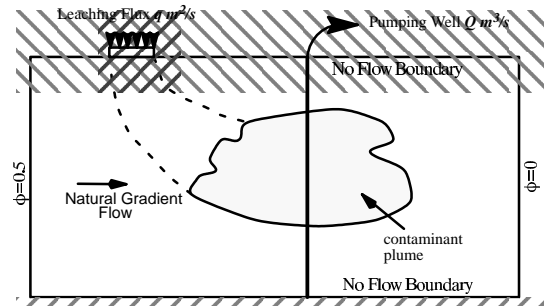


Fig 1: Vertical cross-section of the basic problem setup

## 5 Parallelization Strategy

### 5.1 Groundwater Flow Problem

We tried three different parallel decomposition techniques for the flow problem: block node decomposition, one-dimensional domain decomposition, and two-dimensional domain decomposition. Each has its own merits and shortcomings depending on the problem as discussed later in section 6.1. The word "nodes" refers to the grid points of the computational domain and not processor nodes in this paper. All three decompositions assumed a regular numbering scheme for the nodes and elements based on a logically rectangular grid. For the Intel Paragon machines, explicit message passing using the *nx* library was used to handle processor domain interface data communication.

In all three decompositions, each processor is responsible for a defined set of nodes. Although there is no overlap in these nodes, the boundary cells (or elements) overlap as shown in Fig 2. We overlap the boundary cells to avoid additional communication at the end of assembly. Each processor assembles its own submatrix containing the rows of the nodes that is assigned to it. Explicit message passing is performed mainly at the following stages of the algorithm: before processor submatrix computation and assembly, during the well node submatrix update (only if the nodes of a well reside on the boundary of a processor domain), and in the matrix

vector multiplication stage of the conjugate gradient algorithm.

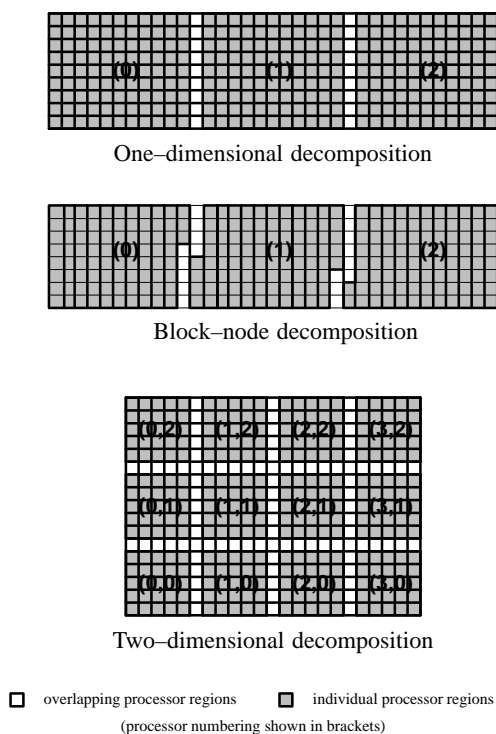


Fig 2: The three domain decomposition strategies

In the conjugate gradient algorithm we used either a simple diagonal or a block diagonal preconditioning. Block diagonal preconditioning is simply incomplete Cholesky preconditioning applied to each processor submatrix by neglecting the coupling terms. For the two-dimensional decomposition we had to perform a local numbering of the nodes for each processor subdomain so that the 27-diagonal band structure is preserved within each processor submatrix. This results in non contiguous rows being allocated to each processor in the global sense. But for all computational purposes this does not matter since each processor is responsible only for its portion of the rows which are locally contiguous. However, when the solution output is written to a file we had to make sure that the proper order is preserved in the global sense. This resulted in non-contiguous writes to a file which was very slow using the paragon  $nx$  calls when a large number of processors were involved. We circumvented this problem by having only

a few processors perform the I/O using temporary buffers.

## 5.2 Solute Transport Problem

A major difficulty in the parallelization of the particle tracking code is that a static domain decomposition has to be performed on the large velocity (input to the code from the flow solver) and concentration (output) fields so that memory is distributed across all processors. However, the particle decomposition cannot correspond to the velocity and concentration domain decomposition due to load balancing difficulties. To ensure good load balancing a simple static block particle decomposition is required. However, this results in the difficulty that arbitrary data communication may be required to map the velocities on to the particles at each time step of the simulation as the velocities do not generally reside on the same processor as the particle. This difficulty was resolved using the DOLIB library provided by [7] for the Intel Paragon machines. Using the DOLIB library we were able to emulate global shared memory for the velocity and concentration fields. That is, after an initial domain decomposition of the velocity and concentration fields we scatter the local portions of these arrays into the global shared memory using DOLIB. The velocity mapping to the particle arrays are done by the "dodgather" primitive of DOLIB at each time step. The concentration computation is performed using the DOLIB primitive "dodaxpy".

## 6 Performance and parallelization issues

### 6.1 Groundwater flow problem

The block node decomposition gave the best load balancing and was the fastest scheme for aquifer geometries with one dimension much larger than the other two. The one dimensional domain decomposition gave the least communication overhead for similar geometries. In both cases the communication time was only about 5% of the total cost for such geometries. However, for domains with almost equal dimensions for all three sides, the block node decomposition and one-dimensional domain decompositions entailed large volume of communication and high memory requirements. There is also a restriction that the number of processors

has to be smaller than at least one side of the domain for these two decomposition schemes to be effective. But for common groundwater aquifer geometries the first two decompositions are generally sufficient since the vertical dimension is usually much smaller than the other two dimensions.

The two-dimensional decomposition was the most robust since it does not have restrictions on the aspect ratio of the domain or the number of processors to be effective. However, the two dimensional decomposition required communication from a maximum of 8 neighboring processors (as opposed to only 2 in the other two cases) resulting in a large number of shorter messages. This in turn resulted in added latency overhead for communication. Generally the communication overhead was about 15–20% of the total time for this decomposition. Also, the two-dimensional decomposition required many temporary buffers to gather messages into contiguous blocks prior to sending. Therefore we found the two-dimensional decomposition to be effective only for very large problems requiring a large number of processors where the other two schemes have many restrictions.

With regard to preconditioning we found that diagonal preconditioning was sufficient and efficient for most problems. Although the block diagonal preconditioning converged within 3–5 times fewer iterations than diagonal preconditioning the solution times were generally higher. This is primarily due to the non-vectorizable operations such as forward and backward triangular solves involved in this preconditioning step. Using diagonal preconditioning and a two dimensional decomposition we were able to solve groundwater flow problems of size 1001x501x41 (approximately 20 million nodes) with relative ease (about 2 minutes per matrix solve) on the 1024-node XPS/150 at ORNL. However for practical purposes most of the simulations were performed using a 201x101x41 grid (800,000 elements) on a 32-node partition of the Intel Paragon XP/S 35. Performance for double precision problems was in the order of 10–15 Mflops per processor on the Paragon XP/35. In Fig 3, we show the solution timings for this

800K problem and a 9.6M problem on the Intel Paragons XPS/35 and XPS/150.

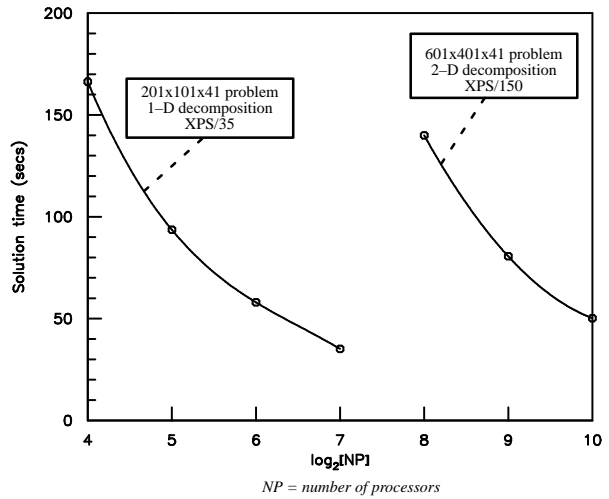


Fig 3: Flow code solution timings on the Intel Paragons

For the 1–D decomposition, the time spent in explicit message passing remained approximately constant with increasing number of processors. This is primarily because, the volume of communication remains the same for this decomposition regardless of the number of processors. However, for the 2–D decomposition, a decreasing trend in the communication time was observed with increasing number of processors. This is primarily due to the reduction in the volume of communication as the domain is split into smaller subregions.

### 6.2 Solute Transport Problem

The transport code performance was not great but this is to be expected with the simplistic approach we took with the DOLIB shared memory emulation. In Fig 4, we show the timings for 160K particles through 100 time steps of the contamination simulation on the Paragon XPS/35. The velocity field for this problem was obtained from a flow simulation consisting of 800K elements. As seen from this figure, the DOLIB overhead in most cases is in the 75–80% range. Although this would be considered very high overhead by many, we consider this to be fairly satisfactory due to highly arbitrary communication patterns occurring in the velocity mapping stage. As a measure of comparison, we note that DOLIB performed slightly better than the CM–2 arbitrary array indexing used in another version of the code [8]. As seen

also from Fig 4, increasing the number of processors for the same problem decreased the DOLIB communication time. This is primarily because the number of particles per processor decreased with increasing number of processors hence requiring less data to be transferred.

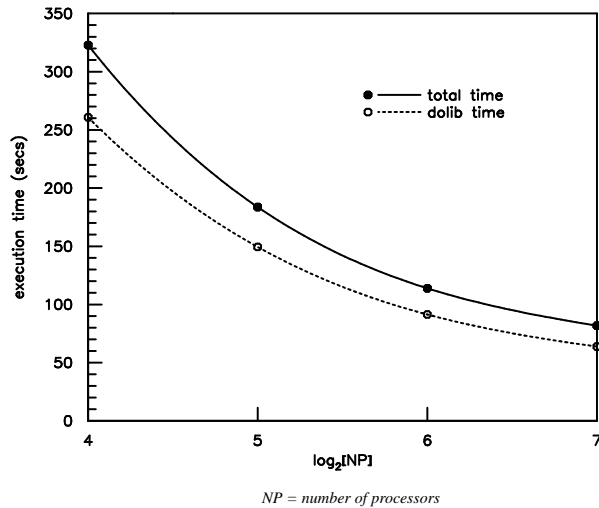


Fig 4: Transport code timings on the Intel Paragon XPS/35

## 7 Conclusions

Impact of geologic heterogeneity upon the effectiveness of pump-and-treat remediation was analyzed in this work using a series of high resolution numerical simulations performed on the Intel Paragon machines. We found that heterogeneity has a profound impact on remediation times. For example, an aquifer with a degree of heterogeneity of  $\sigma^2_Y=1.5$  required 4 times more pumping time to achieve 95% cleanup than a homogeneous aquifer ( $\sigma^2_Y=0.0$ ). For aquifers with  $\sigma^2_Y \geq 2.0$  it was almost impossible to achieve more than 80% cleanup under practical pumping durations.

For the groundwater flow problem, three types of domain decomposition strategies were tried out. Although 1-D and/or block-node decompositions were sufficient for most of our practical simulations, for very large problems requiring large number of processors, a 2-D decomposition was necessary. For the transport problem, we used the DOLIB shared memory library to simplify the parallelization at the expense of high communication overheads.

## Acknowledgements

This work was sponsored by the Center for Computational Sciences of the Oak Ridge National Laboratory under contract DE-AC05-84OR21400 with Martin Marietta Energy Systems, Inc. Ed D'Azevedo and Chuck Romine of the Computing and Mathematics Division at ORNL provided the DOLIB library for the transport simulations.

## References

- [1] Mercer, J.W., D.C. Skipp, and D. Giffin, "Basics of pump-and-treat groundwater remediation technology", Robert S. Kerr Environmental Research Laboratory, Ada, Oklahoma, *U.S. EPA report*, EPA-600/8-90/003, 1990.
- [2] Istok, J.D., Groundwater modeling by the finite element method, *Water Resources Monograph 13*, American Geophysical Union, 1989.
- [3] Huyakorn, P.S., and G.F. Pinder, *Computational Methods in Subsurface Flow*, Academic Press, New York, 1983.
- [4] Tompson, A.F.B., E.G. Vomvoris, and L.W. Gelhar, "Numerical simulation of solute transport in randomly heterogeneous porous media: motivation, model development, and application", *Rep. UCID-21281*, Lawrence Livermore Natl. Lab, Livermore, Calif., 1987.
- [5] Tompson, A.F.B., and L.W. Gelhar, Numerical simulation of solute transport in three-dimensional, randomly heterogeneous porous media, *Water Resour. Res.*, 26(10), 2541-2562, 1990.
- [6] Tompson, A.F.B., R. Aboubu, and L.W. Gelhar, "Implementation of the three-dimensional turning bands random field generator", *Water Resour. Res.*, 25(10), 2227-2243, 1989.
- [7] D'Azevedo, E.F., and C. H. Romine, "DOLIB: Distributed Object Library", *ORNL/TM-12743*, Oak Ridge National Laboratory, August 1994.
- [8] Mahinthakumar, G., and Valocchi, A.J., Application of the Connection Machine to flow and transport problems in three-dimensional heterogeneous aquifers, *Advances in Water Resources*, 15, 289-302, 1992.