# A parallel software for a saltwater intrusion problem *

É. Canot[a], C. de Dieuleveult[b], J. Erhel[b]

[a]CNRS – IRISA, Campus de Beaulieu, 35042 Rennes

[b]INRIA – IRISA, Campus de Beaulieu, 35042 Rennes

**Abstract**

In this paper we present a parallel implementation of a 2D numerical model for the solution of a transient density driven flow in porous media. The physical processes involved are multi-scale, therefore computation time are usually long, thus a special effort has been made to speed-up computations by using parallel architectures.

As most of the CPU time is spent in solving large sparse linear systems, it is important to choose an efficient linear solver which can deal with symmetric and non-symmetric sparse matrices. Accordingly, the parallel direct linear solver MUMPS is used for solving both transport and flow. However, scalability is not completely achieved. Therefore, parallelism is generalised to all computations. Matrices and data are distributed thanks to the partitioning of the METIS package instead of centralising on one processor. Actually, each processor treats its own spatial sub-domain and transfers data to the other ones when necessary.

The resulting software is tested on a standard benchmark : the Henry test case. We use a homogeneous parallel cluster of PCs interconnected by a fast network. Investigation on the network and on the MUMPS options are carried out in order to obtain good performances.

## 1. Introduction

Many areas of the world use groundwater as their main source of freshwater supply. In the particular case of coastal aquifers, one of the major concerns is the seawater intrusion. Moreover, effective management demands a thorough understanding of the variable density groundwater flow system. Numerical simulations help in this sustainable management but they require high performance computing resources.

Generally, transport of solute by groundwater flow does not affect fluid properties, but in the particular case of seawater intrusion it does. The resulting problem becomes difficult to solve because it is highly nonlinear. Indeed transport of saltwater modifies the basic flow by density variations whereas the Darcy velocity calculated by flow, is required to solve advection.

Many numerical models, adapted for this particular case, have been developed : for example, FEFLOW[7], HST3D [13], MOCDENSE [14], SUTRA [17], SWIFT [15] or UG [11].

We present here a parallel program dealing with this topic. The original sequential software TVDV-2D (Transport with Variable Density and Viscosity [2]) has been developed at IMFS in Strasbourg, using robust numerical methods well adapted to density driven flow [18].

Our goal has been to get good performances thanks to parallelism and to allow large scale simulations. Validations of our software are mainly based on a test case, the Henry problem [10] (saltwater front in a confined aquifer initially charged with freshwater).

*This work has been supported by a french government grant, the ACI GRID project called HYDROGRID.

## 2. Model of saltwater intrusion problem

The model of the TVD2D software is first presented. It is described in [2,18,1]. The governing equations of variable density groundwater flow and solute transport are described in detail by Bear and Bachmat [6].

These equations include, classically, fluid and solute mass balances, generalised Darcy's law and equations of state for the liquid density and viscosity.

### 2.1. Mathematical model

#### 2.1.1. Fluid equations

The generalised Darcy law can be written as a function of $h$, the hydraulic head defined by $h = P/\rho_0 g + z$, where $P$ is the pressure, $\rho_0$ is the density of pure water, $g$ is the gravity acceleration and $z$ is the vertical coordinate. This leads to the following equation :

$$\varepsilon v = -\frac{k\rho_0 g}{\mu}\left(\nabla h + \frac{\rho-\rho_0}{\rho_0}\nabla z\right).$$

where $\varepsilon$ is the porosity, $v$ is the fluid velocity, $k$ is the permeability tensor of the porous medium, $\mu$ is the dynamic viscosity of the fluid, and $\rho$ is the mass density of the fluid.

Then, conservation of mass gives:

$$\frac{\partial(\varepsilon\rho)}{\partial t} + \nabla.(\rho\varepsilon v) = \rho Q,$$

where $Q$ is a source term.

Most models assume that the effect of temperature can be neglected, porosity is only a function of pressure whereas density and viscosity are functions of pressure and solute mass fraction. Moreover, in our case, we assume that $\varepsilon$ and $k$ are constants like fluid viscosity and fluid density is independent of pressure but a linear function of the solute mass fraction.

By defining

$$\alpha = \frac{1}{1-\varepsilon}\frac{\partial\varepsilon}{\partial P},\ \beta = \frac{1}{\rho}\frac{\partial\rho}{\partial P},\ S = \alpha(1-\varepsilon) + \varepsilon\beta,$$

with $\alpha$ the coefficient of compressibility of the porous medium, $\beta$ the coefficient of compressibility of the fluid and $S$ the specific storativity of the porous medium, the mass conservation law can be written as

$$\rho_0\rho g S\frac{\partial h}{\partial t} + \varepsilon\frac{\partial\rho}{\partial C}\frac{\partial C}{\partial t} + \nabla.(\rho\varepsilon v) = \rho Q.$$

with $C$ is the solute mass fraction.

In most cases, the storage term $S$ is very small or null, therefore the discrete mass matrix can be singular.

#### 2.1.2. Transport equations

The solute transport is governed by a convection-diffusion process. Assuming that $\varepsilon\rho$ is almost constant, the solute mass conservation equation can be written as :

$$\frac{\partial C}{\partial t} + v\nabla C = \nabla.(D\nabla C),$$

where $D$ is the dispersion tensor defined by

$$D = D_c + D_m I,$$
$$D_c = \|v\|(\alpha_L E(v) + \alpha_T(I - E(v))), E_{i,j}(v) = \frac{v_i v_j}{\|v\|^2},$$

where $D_m$ is the molecular diffusion coefficient, $\alpha_L$ is the longitudinal dispersivity, and $\alpha_T$ is the transverse dispersivity.

### 2.2. Numerical discretisation

The global system is discretised both in space and time. For the transport equation, operator splitting is applied, thus allowing adaptive numerical schemes. The convective part of the transport is spatially discretised by a Discontinuous Finite Element scheme (DFE) stabilised with a slope limiter, whereas the dispersive term in the transport equation is spatially discretised by a Mixed Finite Element scheme (MFE). A MFE scheme is also used in the flow equations, in order to get an accurate fluid velocity.

After space discretisation, a fully coupled stiff system of differential algebraic equations is obtained, which is discretised in time by an implicit Euler scheme, excepted in the convective part where the mass fraction is explicit in time.

Finally, at each time step, a system of nonlinear equations is solved by using a fixed-point scheme, more precisely a nonlinear Gauss-Seidel iterative method. This allows to separate transport equations from flow equations. The stopping criterion is based on the maximum differences between the heat and the concentration on the edges at iteration $k$ and $k+1$. As far as we know, there is no proof of convergence for this specific nonlinear system but in practice, we get convergence by reducing the time step.

Roughly, each time step can be written :

$$\begin{cases} \rho^{n+1} = Density(C^{n+1}) \\ A_{flow}(\rho^{n+1})h^{n+1} = b_{flow}(\rho^{n+1}, C^{n+1}, h^n) \\ v^{n+1} = Velocity(h^{n+1}) \\ C^* = b_{convection}(v^{n+1}, C^n) \\ D^{n+1} = Dispersion\_Tensor(v^{n+1}) \\ A_{dispersion}(D^{n+1})C^{n+1} = b_{dispersion}(C^*, D^{n+1}) \end{cases}$$

The most time-consuming parts are the linear solvers, involving large sparse matrices $A_{dispersion}$ and $A_{flow}$. Other parts involve computation of the density, the velocity, the dispersion tensor but also the convection scheme and the matrices calculation. These matrices are computed at each nonlinear iteration since they depend respectively on the velocity and on the density. The matrix $A_{dispersion}$ is symmetric positive definite whereas the matrix $A_{flow}$ is non-symmetric but with a symmetric structure (the non-symmetry comes from the density in the mass balance equation).

## 3. Parallel implementation

Numerical simulations for saltwater intrusion must deal with a very large number of time steps. Moreover, 3D geometries imply a large number of cells and large linear systems. Therefore, a special effort has been made to accelerate computations.

### 3.1. Parallel linear solvers

Most of the CPU time is spent in solving large sparse linear systems. Therefore, it is important to choose an efficient linear solver which can deal with symmetric definite positive or non-symmetric sparse matrices in order to reduce execution time.

The choice of the appropriate method is a hard task because of the variety of packages and possible options of each solver [4]. We choose to use a direct method implemented in a parallel library. Our choice is the direct linear solver MUMPS [5] for solving transport and flow.

MUMPS (Multifrontal Massively Parallel Solver) is a package using a multifrontal technique for solving linear systems of equations of type AX=b, where the matrix A is sparse and can be either unsymmetric like in the flow part, symmetric positive definite like in the transport part, or general symmetric. We choose this package because it is free and is known to provide efficient results [9].

The MUMPS solver is decomposed into three steps : symbolic factorisation, numerical factorisation and triangular solvings. The symbolic factorisation is executed only once, in the initialisation step for the flow and in the first iteration step for the transport, because the structure of the matrices is fixed during the whole simulation.

Different options for the pivot order are possible. The pivot order consists in reordering the unknowns of the matrix to reduce fill-in during factorisation.

1. Approximate Minimum Degree (AMD)[3],
2. user-defined ordering,
3. Approximate Minimum Fill (AMF),
4. PORD [16],
5. METIS [12],
6. AMD with automatic quasi-dense row detection (QAMD).

### 3.2. Data distribution and parallel matrix generation

Scalability is not completely achieved by the choice of a parallel linear solver, because of the other computations.

Therefore, parallelism is generalised to all computations in order to speed up again the execution. This also reduces memory requirements. Indeed, matrices and data are distributed instead of centralising them on one processor. Data are partitioned by the free METIS [12] package according to the elements of the mesh. Indeed, METIS is a well-known package for partitioning large irregular graphs and large meshes, and computing fill-reducing orderings of sparse matrices.

Each subdomain contains the data corresponding to the edges and nodes belonging to these elements. Each processor treats its own sub-domain and transfers data to other interfaces when necessary.

Currently, the matrix needed by MUMPS is distributed on processors but neither the right-hand side nor the solution which are centralised on processor 0. It is one restriction of MUMPS, but it is planned to be changed in the next version of MUMPS. Otherwise, only the I/O operations are centralised on a single processor.

## 4. Results

### 4.1. Test case

In order to validate our modifications, a classical test case is used : the Henry [10] problem.

This seawater intrusion problem describes the advance of a saltwater front in a confined aquifer initially charged with freshwater.

This test case admits a steady state which is represented in Figure 1. The water is considered saturated in salt ($c = 1$) when the density is $1200\ kg/m^{-3}$ whereas the density of fresh water is equal to $1000\ kg/m^{-3}$.

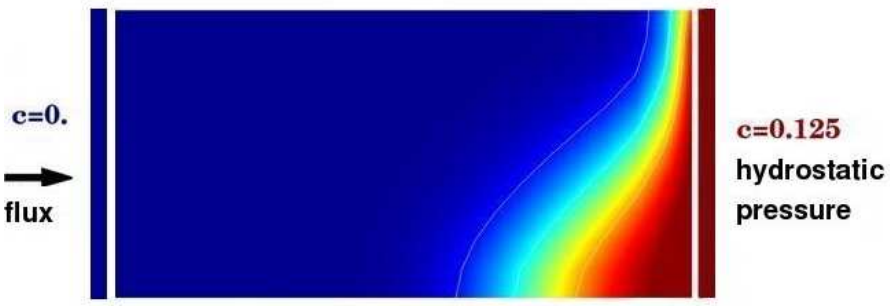


Figure 1. Advance of a saltwater front in a confined aquifer initially charged with fresh water : Henry test case at the steady state and boundary conditions of the problem.

Table 1
Parameters for the Henry problem.

| | |
|---|---|
| Permeability | $kx = ky = 1.0204 \times 10^{-9}\ m^2$ |
| Porosity | $\epsilon = 0.35$ |
| Dispersivity | $\alpha_L = \alpha_T = 0\ m$ |
| Molecular diffusion coefficient | $D_m = 18.86 \times 10^{-6}\ m^2 s^{-1}$ |
| Flux | $Q = 6.6 \times 10^{-5}\ kg/s$ |
| State equations | $\rho = \rho_0 + 200 C_m$ ($C_m$ is the mass fraction) |
| | $\mu = 10^{-3}\ Pa.s$ |
| Domain | $2 \times 1\ m$ |

Here, the density variations are small, but this test case is the first stage of the model validation because of the existence of a semi-analytic steady-state solution. It has become a classic test of variable-density flow. The steady state appears after 120 minutes. With this problem, the results obtained with two regular meshes, one with 254 horizontals elements by 126 verticals elements and the other with 510 by 254 elements, are presented.

We have also done experiments with the Elder test case [8], which deals with tens of years but is more unstable.

### 4.2. Choice of the network

Parallel experiments have been firstly run using MPI on a Fast Ethernet (100 Mb/s) network with bi-processors Intel Xeon (CPU 2.4GHz, cache 512KB). But the results are not so good. In fact, the matrix concerned are relatively small with little time of calculation and a very sparse structure. The use of a high latency network in connection with relatively small volume of data exchanged explains these poor results.

Thus, similar machines are used but with a faster network, Myrinet (2Gb/s) to obtain efficient results.

### 4.3. Choice of the options

As the meshes used are regular, we investigated if a nested dissection on a regular grid could improve performances. We have defined an ordering based on the following idea :

It consists in partitioning recursively in two parts the original mesh in the largest dimension, and then, renumbering the edges accordingly to this partition. The mesh ordering is then optimized as

we can see on Figure 2 with the representation of the sparsity structure of the matrix.

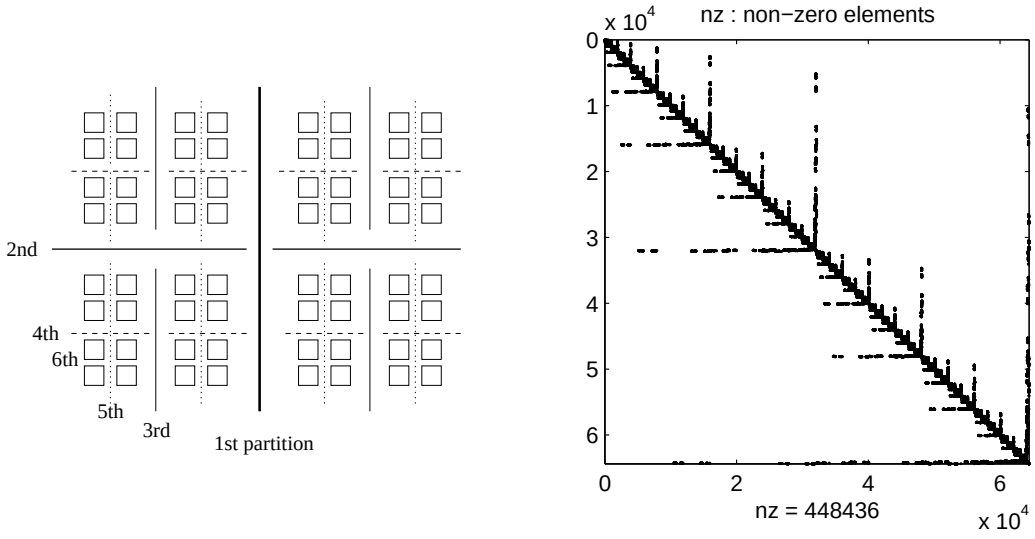


Figure 2. Renumbering of the edges, the partitioning and matrix sparsity structure for a mesh of 254x126 elements.

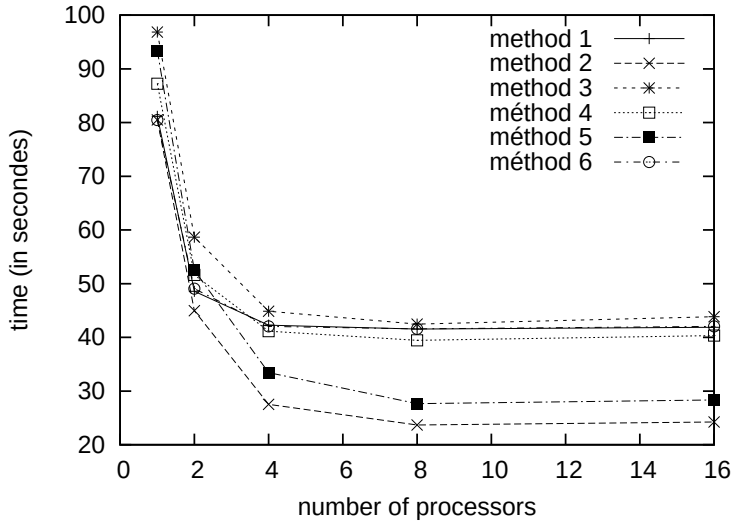


Figure 3. Time spent in MUMPS for different methods of pivot order with mesh 254x126 elements for the Henry test case and 10 time steps.

We see on Figure 3 that the best pivot order is with our user-defined ordering (method 2) just before METIS one (method 5). Actually, these two methods share the same principle, nested dissection, but whereas method 2 is applied only on regular meshes, method 5 is applied on any mesh. Because of this difference, method 5 is used in the following studies.

### 4.4. Parallel results

Time measurements are reported in Figure 4. In this figure, three different parts of the computation are plotted :

- the MUMPS parallel solving timing,
- the initialisation, the storage and the visualisation timings, which are sequential and are not representative of the calculation. Indeed, the initialisation only occurs one time whereas the storage and the visualisation are optional,
- the other parts of the calculation which are parallelised thanks to the METIS partitioning.

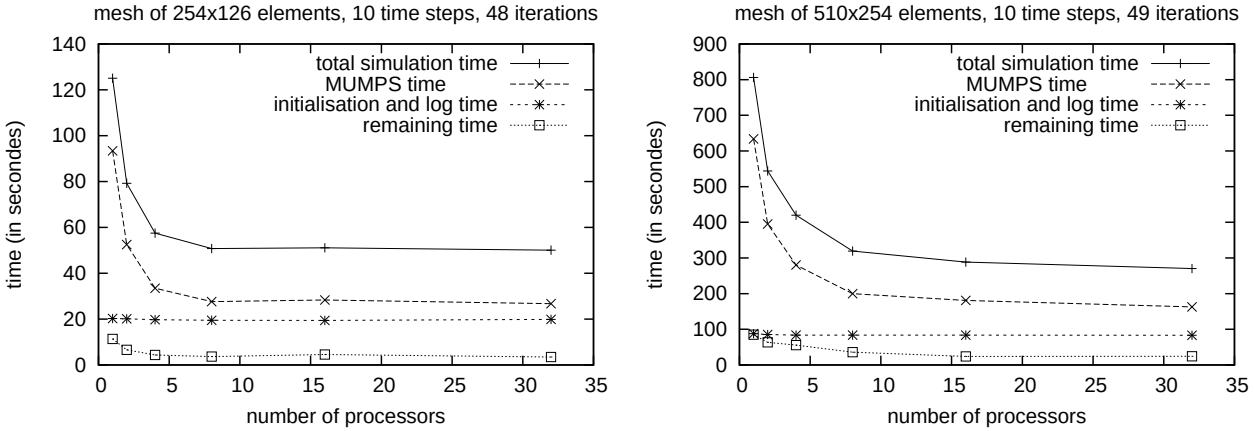


Figure 4. time results with MUMPS and the Henry problem on 2D meshes.

As expected, solving the linear systems takes a large part of the total simulation time. For the smallest mesh (system size N= 64262), parallelism is very efficient up to 4 processors. For the largest mesh (system size N=259590) performances are still good with 8 and 16 processors.

## 5. Conclusion

Numerical simulations in hydrogeology are quite often based on coupled models like the saltwater intrusion problem involving strongly flow and transport coupling.

Our parallel software allows to speed up the execution especially by the use of the parallel linear solver but also by data partitioning.

Moreover, simulations with 3D geometry should show better performances because the matrices would be much larger. Besides, parallelisation of the visualisation could also be investigated as well as the comparison with other linear solvers. Another direction of work will be to improve the coupling of the flow and the transport.

## References

[1] P. Ackerer, A. Younes, S.E. Oswald, and W. Kinzelbach. On modelling of density driven fbw. *Calibration and Reliability in Groundwater Modelling : Coping with Uncertainty (Red book of the ModelCARE 99 Conference), IAHS Publication*, 265:377–384, 2000.

[2] Ph. Ackerer, A. Younes, and R. Mosé. Modeling variable density flow and solute transport in porous medium : 1. numerical model and verification. *Transport in Porous Media*, 35:345–373, 1999.

[3] P. Amestoy, T. A. Davis, and I. S. Duff. An approximate minimum degree ordering algorithm. *SIAM Journal on Matrix Analysis and Applications*, 17:886–905, 1996.

[4] P. Amestoy, I.S. Duff, J.-Y. L'Excellent, and X. S. Li. Analysis and comparison of two general sparse solvers for distributed memory computers. *ACM Transactions on Mathematical Software*, 27(4):388–421, 2001.

[5] P. R. Amestroy, I. S. Duff, J.-Y. L'Excellent, and J. Koster. *MUltifrontal Massively Parallel Solver (MUMPS Version 4.3) Users' guide*, July 2003.

[6] J. Bear and Y. Bachmat. *Introduction to Modeling of Transport Phenomena in Porous Media*. Kluwer Academic Publishers, Dordrecht, 1991.

[7] H. J. Diersch and O. Kolditz. Coupled groundwater flow and transport: 2. thermohaline and 3d convection systems. *Advances Water Ressource*, 21(5):401–425, 1998.

[8] J. W. Elder. Numerical experiments with a free convection in a vertical slot. *Journal of Fluid Mechanics*, 24:823–843, 1966.

[9] A. Gupta. Recent advances in direct methods for solving unsymmetric sparse systems of linear equations. *ACM Transactions on Mathematical Software*, 28(3):301–324, 2002.

[10] H. R. Henry. Interfaces between salt water and fresh water in coastal aquifers. Technical Report 1613-C, U.S. Geological Survey, Water Supply Paper, 1964.

[11] K. Johannsen, W. Kinzelbach, S.E. Oswald, and G. Wittum. The saltpool benchmark problem - numerical simulation of saltwater upconing in a porous medium. *Advances in Water Resources*, 25(3):335–348, 2002.

[12] George Karypis and Vipin Kumar. Metis a software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing ordering of sparse matrices version 4.0. Technical Report MN 55455, University of Minnesota, Department of Computer Science/Army HPC Research Center, September 1998.

[13] K. L. Kipp. Guide to the revised heat and solute transport simulator : Hst3d - version 2. Technical Report 97-4157, US Geological Survey, Water Resources Investigations, 1997.

[14] L. F. Konikow, P.J. Campbell, and W. E. Sanford. Modelling brine transport in a porous medium : a re-evaluation of the hydrocoin level 1, case 5 problem. In K. Kodar and P. Van Der Heijde, editors, *Calibration and Reliability in Groundwater Modeling*, 237, pages 363–372. IAHS Press IAHS Publication, 1996.

[15] M. Reeves, D. S. Ward, N. D. Johns, and R. M. Cranwell. Theory and implementation of SWIFT II, the sandia waste-isolation flow and transport model for fractured media. Technical Report SAND83-1159, Sandia National Laboratory, 1986.

[16] J. Schulze. Towards a tighter coupling of bottom-up and top-down sparse matrix ordering methods. *BIT*, 4(800-841):2001, 41.

[17] C. I. Voss. A finite element simulation model for saturated-unsaturated fluid-density dependent groundwater flow with energy transport or chemically-reactive single species solute transport. Technical Report 84-4369, US Geological Survey, Water Resources Investigations, 1984.

[18] A. Younès, Ph. Ackerer, and R. Mosé. Modelling variable density flow and solute transport in porous medium : 2. re-evaluation of the salt dome flow problem. *Transport in Porous Media*, 35:375–394, 1999.