DETERMINISTIC SENSITIVITY ANALYSIS FOR THE NUMERICAL SIMULATION OF THE TRANSPORT OF CONTAMINANTS

ESTELLE MARCHAND[1], FRANÇOIS CLÉMENT[2]

[1]ANDRA, 1-7 rue Jean Monnet, Parc de la Croix-Blanche, F-92290 Châtenay-Malabry & INRIA Rocquencourt, B.P. 105, F-78150 Le Chesnay [2]INRIA Rocquencourt

Abstract

This work is devoted to the deterministic sensitivity analysis for the numerical simulation of the transport of contaminants. The method is based on the singular value decomposition of the Jacobian matrix of the mathematical model of the problem. We compare different ways of computing the Jacobian matrix: automatic differentiation and differentiation "by hand" from analytic formulas, in both direct and reverse modes. Here we consider only the first stage and the sensitivity analysis for the flow computation from Darcy's equation.

1. INTRODUCTION

The questions of safety and uncertainties are central to feasibility studies for an underground waste storage site. One of the important points to be considered is the problem of the evaluation of concentration uncertainties which are due to input parameter uncertainties. These concentration uncertainties can be obtained by probabilistic methods. These methods give good results and are relatively easy to implement, but they are expensive because they require a large number of simulations. The deterministic methods investigated here are much less demanding in computing time but they give only a local information. Thus the two approaches are complementary and deserve to be developed.

2. MATHEMATICAL MODEL

2.1. **The full mathematical model.** The full mathematical model constists of a flow equation based on Darcy's law, where the Darcy velocity is assumed stationary, a transport equation, ie mass balance equation, for each radionuclide, and a law of exchange between liquid and solid phases for each radionuclide.

As the flow equation is independent of time and of the concentrations, we can solve it first and then solve separately the mass balance equations for the individual radionuclides. Here we present the sensitivity analysis for the flow equation. We will be interested in the sensitivity of the Darcy velocity to the distribution of permeabilities.

2.2. **Darcy's equation.** Saturated flows are described by the stationary system of equations

$$\left\{\begin{array}{rcll} \text{div } \vec{u} & = & q & \text{in } \Omega \\ \vec{u} & = & -K\vec{\nabla}p & \text{in } \Omega \\ p & = & p_D & \text{on } \Gamma_D \\ \vec{u}\cdot\vec{n} & = & g_N & \text{on } \Gamma_N \end{array}\right.$$

where p is the pressure, $\vec{u}$ is the water flow velocity ($\vec{u}$ will be the convection velocity in the transport equation), q is a source term and K is the ratio between the permeability of the porous medium and the water viscosity. The gravity is not taken into account here.

It will be important to differentiate a dicretized problem, and not to discretize the differentiation, because the operations of differentiation and discretization do not commute.

The problem is discretized using a mixed hybrid finite element method. So we have to consider a triangulation of the domain Ω: $\tau_h = \{T_i\}_i$. The set of the edges or faces (in the interior or on the boundary) of the triangulation is $\{F_j\}_j$. With this formulation, the pressure is considered constant per cell and we introduce supplementary unknowns which are traces of pressure (one value for each face) and flows through each face (one value per boundary face, two values per interior face, since the continuity of the flow is imposed by a supplementary equation and not by the choice of the unknows).

We have decomposed the computation domain into zones of uniform permeability and we suppose that the tensor of permeability K is symmetric everywhere. Hence, the application to differentiate takes as input a vector containing, for each zone i, either the value of the permeability coefficient in this zone, if permeability is scalar in zone i, or the 6 coefficients of the lower part of the symmetric permeability tensor and it returns either the vector containing all the unknowns representing velocity fluxes, or a vector containing a few of the velocity flux components.

3. DIRECT AND REVERSE MODES

We can write the discretized Darcy's equation as follows:

$$\mathcal{C}(\mathcal{U}, K) = 0, \tag{1}$$

where

$$\mathcal{U} = \begin{pmatrix} U \\ P \\ L \end{pmatrix} \; ; \tag{2}$$

$$\mathcal{C}(\mathcal{U}, K) = \begin{pmatrix} A\,(K^{-1})\,U + BP + C\tilde{L} - F_1 \\ {}^tBU - F_2 \\ {}^tCU - F_3 \\ L^D - P_{lim} \end{pmatrix}. \tag{3}$$

The components of U are the fluxes of $\vec{u}$ through the faces of the mesh (one unknown for each boundary face and two unknowns for each interior face), P contains a value of pressure per cell and L contains the traces of pressure, ie one value of pressure per face. We have decomposed L into $\tilde{L}$ containing the components of L either internal to the domain or on the part of the boundary where we have imposed Neumann boundary conditions, and L^D containing the components of L on the part of the boundary where we have imposed Dirichlet boundary conditions.

The first equation corresponds to Darcy's law where F_1 depends on the Dirichlet boundary conditions. The second equation corresponds to the conservation law where F_2 contains the source term. The third equation represents the continuity of the velocity flux where F_3 contains Neumann boundary conditions.

In direct mode, in order to obtain the Jacobian matrix of a function giving some components of U according to the distribution of K, we have to solve for each component of the input of the function an equation of the form

$$\text{find } \mathrm{d}\mathcal{U} \text{ such that } \frac{\partial \mathcal{C}}{\partial K}\mathrm{d}K + \frac{\partial \mathcal{C}}{\partial \mathcal{U}}\mathrm{d}\mathcal{U} = 0. \tag{4}$$

The solution of this equation is one column of the sought Jacobian matrix. Thus, if the input length is N_i and the length of the vector $\mathcal{U}$ is N_o, there are, independently on the number of velocity components we are interested in, N_i linear systems of size $N_o \times N_o$ to solve.

In reverse mode, to differentiate the component number l of the velocity, we first have to determine the corresponding adjoint state λ^l by solving

$$\left[\frac{\partial C}{\partial \mathcal{U}}\right]^T \lambda^l = -g_U^l, \tag{5}$$

where the l^{th} component of g_U^l is equal to 1 and all other components are equal to 0. Then the l^{th} line of the Jacobian matrix is given by

$$g_K^l = \left[\frac{\partial \mathcal{C}}{\partial K}\right]^T \lambda^l. \tag{6}$$

So there is one linear system of size $N_o \times N_o$ to solve for each component of U that we are interested in. The number and size of systems to solve is independant of the irregularity of the permeability.

Thus it is advantageous to be able to differentiate the resolution of Darcy's equation both in direct and reverse modes since we may want to modify the choice of input or output parameters.

We use the `C++` finite element library `LifeV`, see [LifeV].

4. AUTOMATIC DIFFERENTIATION

4.1. **Motivation.** A simple way to compute derivatives is using finite differences, but this is not always accurate because it is difficult to choose the appropriate step length. An accurate and efficient way is to compute all the derivatives "by hand", following precisely what is written on the paper. Unfortunately this way is quite error prone, and it can take a very long time to validate such an implementation of the adjoint state. Automatic differentiation provides an alternative way which is partially "automatic". We have also used automatic differentiation to help in validating the "by hand" implementation.

4.2. **Principle.** There are two kinds of automatic differentiation:

(1) automatic differentiation by source transformation: a differentiated source code is directly computed from the source code defining the function. As of yet, this method does not make it possible to differentiate in reverse mode source codes written in `C` or `C++`.

(2) automatic differentiation by operator overloading. The user has to slightly modify his/her source codes to call tools of the automatic differentiation library and the differentiation is obtained at run time (it is not necessary to compile a derivated code). This way is used by the library `adolC` (see [Griewank et al, 2004]) that we have used for our problem.

4.3. **Performance comparison.** We have compared performances of "manual" differentiation and automatic differentiation using the example of a cylindrical domain meshed by 3200 hexahedra in which the permeability is supposed to be both scalar and uniform. We have imposed a monodimensional flow along the principal axis of the cylinder: we have imposed null flux conditions on the lateral face; on the bases we have imposed either Dirichlet conditions on both bases or Neumann conditions on both bases (in this second case the system of equations has an infinite number of solutions but we can still find one of them). The derivative is very simple in this case, so it is also a first test for the correctness of the differentiated codes:

- "Dirichlet-Dirichlet" test-case, $\delta\vec{u} = \vec{0}$ and $\delta\vec{\nabla}p = -\frac{\delta K}{K}\vec{\nabla}p$
- "Neumann-Neumann" test-case, $\delta\vec{u} = \frac{\delta K}{K}\vec{u}$ and $\delta p = 0$ and $\delta\vec{\nabla}p = \vec{0}$.

4.3.1. *Developement time.* Developement time is approximately twice as long for manual differentiation as for automatic differentiation. We all the same encountered important difficulties for automatic differentiation with the use of huge exterior libraries such as `Lapack` (direct methods for full matrices) and we had to give up using `AZTEC` (a library of iterative methods to solve linear systems with sparse matrices).

4.3.2. *Execution times.* To compare execution times, we have decomposed the executiom time for the manual mode into four parts:

(1) init: time necessary to "build" the problem (mainly mesh reading);
(2) values: computation of the solution of Darcy's equation;
(3) direct: computation of the derivative in direct mode;
(4) reverse: computation of one component of the derivative in reverse mode.

`AdolC` considers a function as a list of elementary operations. The first time it computes the function, it stores, in a file or in memory, a list of data (a "tape") representing this list of operations. It can reuse the tape several times to evaluate derivatives and values for different input points. We have decomposed the execution time for automatic differentiation into four parts:

(1) init: time necessary to "build" the problem;
(2) taping: first computation of the function;
(3) `forward`: reading the tape while computing the derivatives at the same time;
(4) `vec_jac`: reading a tape "backward" (preliminary operations are needed for this) to compute one component of the derivative in reverse mode.

Note that in this particular case, with just one input value (the scalar value of uniform permeability), it is not at all reasonable to use the reverse mode.

The results of this comparison are reported in Table 1.

So, for such a problem, execution time does not seem to be a problem for the use of automatic differentiation.

Neumann Neumann				Dirichlet Dirichlet			
manual		automatic		manual		automatic	
init	0.46s	1.03s	init	init	0.47s	1.04s	init
values	7.29s	12.68s	taping	values	7.33s	15.05s	taping
direct	2.76s	2.76s	`forward`	direct	2.80s	3.48s	`forward`
1×reverse	5.67s	6.15s	1×`vec_jac`	1×reverse	5.67s	7.40s	1×`vec_jac`

TABLE 1. Comparison of execution times for the two derivated codes.

4.3.3. *Memory management.* Automatic differentiation, in the way we use it here, creates several tape files of a few hundreds of Megaoctets. Some difficulties remain for us in the memory management for automatic differentiation. However, we should be able to overcome them by decomposing the problems into small enough subproblems, and using methods described in [Griewank and Walther, 2000].

4.4. **Discussion about automatic differentiation.** We would suggest using manual differentiation for portions of code using exterior libraries and for portions of code using iterative methods. It should be very interesting to use automatic differentiation for the case of programs for which the list of operations realized during the execution depends strongly, but not iteratively, on the parameters, for example for the computation of upwinded schemes which will be used for the transport equation.

Moreover, automatic differentiation can be a very interesting tool for checking a manual differentiation. It is indeed very easy to generalize: it was very fast to transform the program differentiated according to a scalar uniform permeability, to obtain a program differentiated according to the non-uniform non-scalar permeability and according to two parameters representing boundary conditions, and it took quite a long time to implement the same generalization for a finite differences checking routine.

5. SINGULAR VALUE DECOMPOSITION

Let A be any rectangular matrix. The singular value decomposition of A is given by

$$A = USV^T$$

where U and V are unitary matrices and S is a diagonal matrix of the same size as A. The columns of V are the singular vectors in the input space and the colums of U are the singular vectors in the output space. The diagonal terms of S are the singular values of A, that is to say the square roots of the eigenvalues of the square symmetric matrix A^TA. They are nonnegative numbers ordered decreasingly.

The rate of the decrease of the singular values provides a hierarchical classification of the sensitivities of the output values with respect to the input entries.

6. DETERMINISTIC SENSITIVITY ANALYSIS RESULTS

6.1. **Test case description.** The data for this test case were provided by ANDRA, see [Martin, 2004]. The computation domain, whose total dimensions are roughly 500 meters depth, 40 kilometers for horizontal dimensions, is divided into 14 zones and meshed with 300,000 cells. Permeability is scalar in 11 zones and tensorial in 3 zones, so it is represented

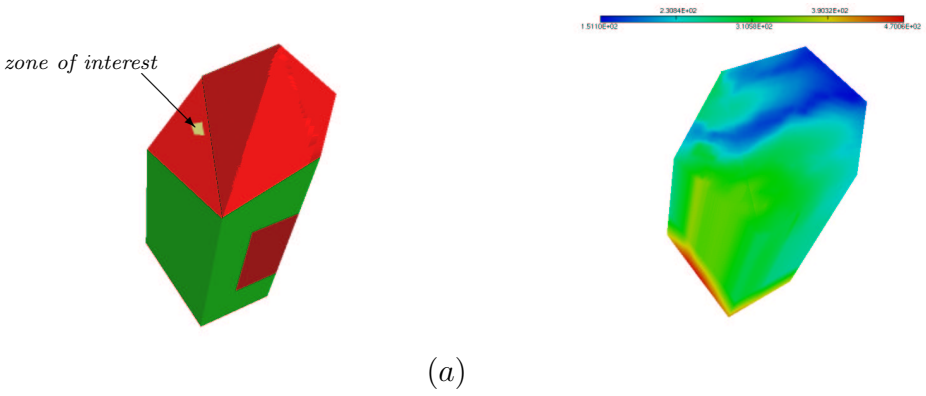
by a vector of 29 components. The tensor permeabilities are diagonal, but by supposing just symmetry we can test the sensitivity of the hypothesis of diagonality. Table 2 gives the values of permeability. For each zone it gives either a scalar or the lower half of the symmetric tensor.

In all the figures vertical dimensions have been increased 100 times. Figure 1-(a) shows the computation domain with boundary conditions: pressure is imposed on the red part and the green boundary is impermeable. We have chosen for sensitivity analysis a few components of fluxes. The location of these components is also shown in Figure 1-(a).

zone number	1, 2, 14	3	4	5	6	7	8	9	10	11	12	13
permeability $\mathrm{m.s^{-1}}$	$\begin{pmatrix} 10^{-12} & & \\ 0 & 10^{-12} & \\ 0 & 0 & 10^{-14} \end{pmatrix}$	10^{-12}	10^{-10}	8.10^{-9}	6.10^{-7}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}	3.10^{-5}

TABLE 2. Permeability distribution.

Figure 1-(b) shows the corresponding distribution of pressures. Water globally flows from high pressures (red in the figure) to the low pressures (blue in the figure).



(a) (b)

FIGURE 1. Computation domain (a) and distribution of pressure, in meters (b).

6.2. **Singular value decomposition.** We have differentiated 12 components of flux according to the distribution of permeabilities. Those 12 components are located on the surface of the domain, in zone number 11 which is the most permeable zone. They are represented in light brown in Figure 1-(a).

The singular value decomposition of the Jacobian matrix of the problem has been computed using the adjoint state method since outputs are less numerous than inputs, and using the manually differentiated code. Too much memory was necessary to use automatic differentiation. Figure 2 gives the singular values. The corresponding singular vectors are given in Figure 3.

The first singular vector of the input space mainly corresponds to the 12^{th} component in the input vector, that is to say to the vertical component of the permeability tensor in zone number 2. This zone is the deepest zone of the domain and covers all its base. It is associated with an output singular vector whose components are all of the same order:

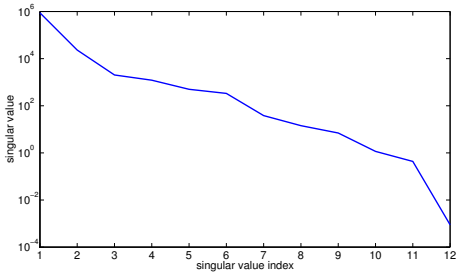
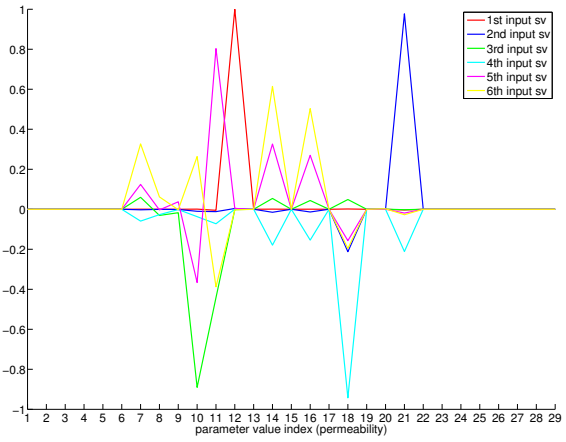
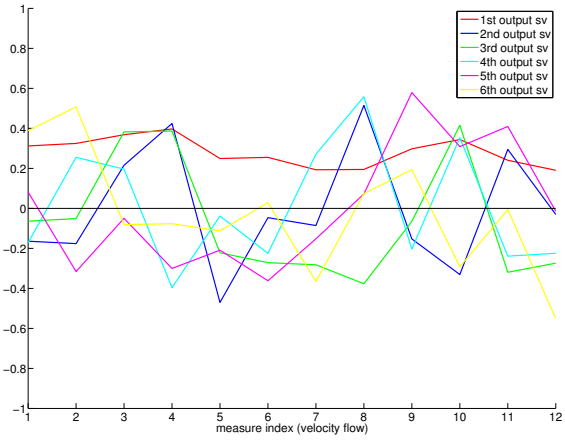


FIGURE 2. The 12 singular values. The first 6 are in a ratio of 10^4.



(a)



(b)

FIGURE 3. The 6 first singular vectors of the input (a) and output (b) spaces.

the vertical permeability in the base of the domain will have an influence on the mean flow in the studied zone.

The second singular vector corresponds mainly to the 21^{st} component in the input vector, that is to say to the scalar permeability in zone number 11, where the studied flow

components are located. The corresponding output singular vector has changing signs. The local permeability influences the corresponding combination of fluxes.

The sensitivities with respect to the permeabilities of the intermediary zones, much more permeable than the base zone, are much weaker. The output velocitites seem to be mainly determined by the imposed pressures and by the permeability of the less permeable zone.

We will compare results obtained with the present deterministic method with those obtained at ANDRA through statistical methods for the same problem.

7. CONCLUSION

This work deals with sensitivity analysis of the first stage for the simulation of the transport of contaminants in porous media: the Darcy solver that computes the velocities. In this case, the model is the function computing the velocity fluxes through portions of the boundary of the domain from the permeability parameters.

The first step consists in computing the Jacobian matrix of this model, either by columns in the direct mode, or by lines in the reverse mode. For such an elliptic problem, the manual implementation of the derivatives remains the most efficient way for the memory management, but automatic differentiation through operator overloading (here with `adolC`) is very useful to validate the manual approach, and moreover, it is competitive in terms of execution time when no memory problem occurs.

The second step is devoted to the sensitivity analysis of the model through the singular value decomposition of its Jacobian matrix. The result is a (weighted) hierarchical list of directions in both the parameter space and the velocity flux space. Then a truncated computation of the uncertainties on the outputs is possible. This first order information is only local but its computation is fast. Thus, it should be complementary to global, but more computationally demanding, probabilistic approaches.

Ongoing works concern a comparison between deterministic and stochastic approaches on this first stage, and then we will further develop the present deterministic method for the full model: parallel implementation of the Darcy solver, and transport equations, then precipitation laws for each contaminant.

Acknowledgments

This work is partially supported by ANDRA, the French National radioactive waste management agency, and is partially supported by the GDR MOMAS.

References

Griewank et al, 2004. A. Griewank, A. Kowarz, J. Utke, O. Vogel, and A. Walther, *Adol-c: A package for the automatic differentiation of algorithms written in c/c++*, documentation for ADOL-C, version 1.9.0, May 2004.

Griewank and Walther, 2000. A. Griewank and A. Walther, *Algorithm 799: Adol-c: An implementation of checkpointing for the reverse and adjoint mode of computational differentiation*, ACM Transactions on Mathematical Software **26** (2000), no. 1, 19–45.

LifeV. LifeV, *http://www.lifev.org*.

Martin, 2004. V. Martin, *Simulation multidomaine d'un écoulement autour d'un site de stockage de déchets*, Ph.D. thesis, Université de Paris 9 Dauphine, 2004.