Development of numerical methods for the reactive transport of chemical species in a porous media : a nonlinear conjugate gradient method

Nicolas Bouillard, Philippe Montarnal, Raphaele Herbin

Int. Conf. on Computational Methods for Coupled Problems in Science and Engineering
COUPLED PROBLEMS 2005
M. Papadrakakis, E. Oñate and B. Schrefler (Eds)

DEVELOPMENT OF NUMERICAL METHODS FOR THE REACTIVE TRANSPORT OF CHEMICAL SPECIES IN A POROUS MEDIA : A NONLINEAR CONJUGATE GRADIENT METHOD

Nicolas Bouillard*, Philippe Montarnal* and Raphaèle Herbin†

*CEA/Saclay,
DM2S/SFME/MTMS,
91191 Gif-sur-Yvette Cedex, France.
e-mail: nicolas.bouillard@cea.fr - philippe.montarnal@cea.fr

† Université de Provence
39 rue Joliot Curie,
13453 Marseille 13, France.
e-mail: Raphaele.Herbin@cmi.univ-mrs.fr - Web page: http://www.cmi.univ-mrs.fr/∼herbin

Key words: porous media, reactive transport, code coupling, nonlinear conjugate gradient method, preconditioning

Abstract. *In the framework of the evaluation of nuclear waste disposal safety, the French Atomic Energy Commission (CEA) is interested in modelling the reactive transport in porous media. At a given time step, the equation system of reactive-transport can be written as a system of nonlinear coupled equations* $\mathbf{F}(x) = 0$. *In the computational code which is presently used, this system is solved using classical sequential iterative algorithms (SIA) [2]. We are currently investigating nonlinear conjugate gradient methods to improve the resolution of the system* $\mathbf{F}(x) = 0$ *where* x *is the discrete unknown. Indeed, the handling of the coupling is improved by numerical derivation along the descent direction. The original feature of this method is the use of an explicit formula for the descent parameter. We choose an approach involving two distinct codes, that is one code for the chemistry and one code for the transport equations.*

1 Introduction

According to a law adopted by the French Parliament in 1991, the French Atomic Energy Commission (CEA) and the French Agency for the Management of Radioactive Waste (ANDRA) are responsible for the disposal and storage of nuclear waste and spent nuclear fuel in France. Both organisms are interested in providing simulation tools so as to evaluate nuclear waste disposal safety. In this context, the ALLIANCES project was launched by CEA [9]. Its aim is to produce a software platform for the simulation of nuclear waste

storage and disposal repository. We focus in particular on reactive transport. The model describes the spatial and temporal evolution of a set of chemical species submitted to transport phenomena and chemical reactions. The transport is characterized by a set of partial differential equations and the chemistry by a set of algebro-differential equations. A sequential iterative coupling scheme has already been implemented, qualified and validated on numerous configurations involving aqueous speciation, dissolution-precipitation, sorption and surface complexation [1].

Indeed, sequential iterative algorithms (SIAs) [2], in which the resolution of the transport is separated from the resolution of the chemical speciation, are often used in industrial computations. To ensure an accurate prediction of the storage geological sites evolution, on large time and space scales, the coupling between chemistry and transport needs to be improved since stronger coupling situations, such as a dissolution/precipitation front, may occur, leading to difficulties when using SIAs. This paper presents conjugate gradient algorithms (CG) with two goals in mind; the first goal is to improve the coupling accuracy and to handle the CPU cost; the second goal is to handle the coupling of two distinct codes, one for the chemistry and one for the transport.

After a description of our reactive transport model and its discretization, nonlinear conjugate gradient methods are presented. Then, in the one dimensional case, nonlinear CG methods are shown to be efficient compared with fixed point methods or Quasi-Newton methods. These first results seem to be promising for more complex simulations.

2 Description of the reactive transport model

We introduce a classical model for reactive transport which describes the spatial and temporal evolution of a set of chemical species submitted to transport phenomena and chemical reactions.

2.1 Assumptions

The following assumptions were made :

- the porosity is assumed to be constant.
- the flow is stationary and the porous media is saturated.
- the dispersion-diffusion tensor $\overline{\overline{D}}$ does not depend on the species.
- transport phenomena only affect species in liquid phase.
- chemical processes are under thermodynamical equilibrium, except some precipitation-dissolution processes assumed to be kinetics-controlled. We also assume that all chemical processes, even kinetics, are instantaneous compared to transport processes.

2.2 Chemical equations

Definition of component species.

Let us consider a geochemical system composed of N_e aqueous chemical species, N_p precipitated chemical species and N_s sorbed chemical species. We assume that there are N_r aqueous independent chemical reactions, with $N_r \leq N_e$. Within the N_e aqueous species, we choose $N_c = N_e - N_r$ component species such that every species can be uniquely represented as a combination of these components, and no component can be represented by another component than itself [2]. All species other than components are called secondary species. A secondary species is the result of a chemical reaction involving the components as reactants. In the chemical system involved here, all precipitated and sorbed species are secondary.

Aqueous equilibrium reactions for the secondary species. Each secondary aqueous species i, for $i = 1, \ldots, N_r$ is connected to the component species by the equilirium reactions, which read:

$$x_i \rightleftharpoons \sum_{j=1}^{N_c} \nu_{ij}^x c_j, \tag{1}$$

where x_i is the concentration of the aqueous secondary species i, c_j, the concentration of the aqueous component species j, and ν_{ij}^x, the stoichiometric coefficient of the jth aqueous component in the ith reaction. The equilibrium equation is based on the mass action law which states that:

$$x_i = K_i^x(x_i, c_j) \prod_{j=1}^{N_c} c_j^{\nu_{ij}^x}, \tag{2}$$

where $K_i^x(x_i, c_j)$ is the thermodynamic equilibrium constant.

Precipitation-dissolution. The kinetic of the precipitation-dissolution reaction:

$$p_k \rightleftharpoons \sum_{j=1}^{N_c} \nu_{ij}^p c_j \,, \tag{3}$$

where p_k denotes the concentration of the k-th of the N_p precipitated species, is written as follows

$$\frac{dp_k}{dt} = \tilde{\omega}_k (1 - \prod_{j=1}^{N_c} c_j^{\nu_{ij}^p} K_k^p(c_j)) \,, \tag{4}$$

where $K_k^p(c_j)$ and $\tilde{\omega}_k$ are the thermodynamic equilibrium constant and the kinetic precipitation constant of the kth precipitated species. At equilibrium, p_k verifies :

$$\begin{array}{rll} \text{if} & K_k^p(c_j) \prod_{j=1}^{N_c} c_j^{\nu_{kj}^p} < 1, & \text{then } p_k = 0, \text{ i.e. precipitation occurs,} \\ \text{otherwise:} & K_k^p(c_j) \prod_{j=1}^{N_c} c_j^{\nu_{kj}^p} = 1 & \text{and } p_k \neq 0, \text{ i.e. no precipitation occurs.} \end{array} \tag{5}$$

Adsorption reactions. Recall that all sorbed species are secondary; hence the equations for adsorption equilibrium are obtained using the mass action law for the N_s sorbed secondary species s_l, in the same way as in the case of the aqueous equilibrium reactions.

Mass conservation equation. For the j-th component species, the mass conservation equation may be written as follows:

$$T_j = c_j + \sum_{i=1}^{N_r} \nu_{ij}^x x_i + \sum_{k=1}^{N_p} \nu_{ij}^p p_k + \sum_{l=1}^{N_s} \nu_{ij}^s s_l = C_j + F_j, \tag{6}$$

where T_j, C_j and F_j are the total analytical (dissolved, sorbed, and precipitated) concentration, total dissolved concentration and total fixed (sorbed and precipitated) concentration, respectively, of the jth component species.

2.3 Hydrologic transport equations

The transport of solutes is described by a set of partial differential equations based on the principle of mass conservation. The transport equation of the jth aqueous component species writes:

$$\omega \frac{\partial c_j}{\partial t} - \nabla.\left(\overline{\overline{D_j}} \nabla c_j - c_j \vec{u}\right) = R_j^{(c)}(c_1, .., c_{N_c}, x_1, .., x_{N_r}, p_1, ..., p_{N_p}, s_1, ..., s_{N_s}),\ j = 1, \ldots, N_c \tag{7}$$

where ω is the porous media porosity, supposed constant, $\overline{\overline{D_j}}$ is the diffusion-dispersion tensor, $\vec{u}$ is the Darcy velocity, $R_j^{(x)}$ is a source term corresponding to the chemical reactions in the aqueous phase. Each secondary aqueous species, chemical precipitated or sorbed species are linked to the other species *via* the reaction terms $R_i^{(x)}$, $R_k^{(p)}$ and $R_l^{(s)}$:

$$\begin{aligned}
\omega \frac{\partial x_i}{\partial t} - \nabla.\left(\overline{\overline{D_i}} \nabla x_i - x_i \vec{u}\right) &= R_i^{(x)}(c_1, .., c_{N_c}, x_1, .., x_{N_r}, p_1, ..., p_{N_p}, s_1, ..., s_{N_s}), \quad (8)\\
i = 1, \ldots, N_r, & \\
\omega \frac{\partial p_k}{\partial t} &= R_k^{(p)}(c_1, .., c_{N_c}, x_1, .., x_{N_r}, p_1, ..., p_{N_p}, s_1, ..., s_{N_s}), \quad (9)\\
k = 1, \ldots, N_p, & \\
\omega \frac{\partial s_l}{\partial t} &= R_l^{(s)}(c_1, .., c_{N_c}, x_1, .., x_{N_r}, p_1, ..., p_{N_p}, s_1, ..., s_{N_s}), \quad (10)\\
l = 1, \ldots, N_s, &
\end{aligned}$$

The Darcy velocity $\vec{u}$ is modelled by the Darcy equation in the stationary state if we assume that the fluid is incompressible :

$$\vec{\nabla}.\vec{u} = 0 \qquad \text{with} \qquad \vec{u} = -\overline{\overline{K}} \vec{\nabla} h\,, \tag{11}$$

where h is the hydraulic head and $\overline{\overline{K}}$ the hydraulic conductivity tensor. Assuming that the transport coefficients are the same for all species, Eq. (6) is used to obtain a set of transport equations for the total concentrations of component species :

$$\omega\frac{\partial C_j}{\partial t}+\omega\frac{\partial F_j}{\partial t}-\nabla.\left(\overline{\overline{D}}\nabla C_j-C_j\vec{u}\right)=0. \tag{12}$$

Eq. (12) can also be written following:

$$\omega\frac{\partial C_j}{\partial t}+\omega\frac{\partial F_j}{\partial t}-L(C_j)=0, \tag{13}$$

where $L(C_j)$ represents the linear convection-diffusion operator.

2.4 The coupling reactive transport model

From Eq. (6) and (13), for the jth component species, we have

$$\left\{\begin{array}{l}\omega\partial_t C_j+\omega\partial_t F_j-L(C_j)=0\\ T_j=C_j+F_j\,.\end{array}\right.$$

The coupling between the transport model and the chemistry is written as:

$$C_j=\Psi_j(T)\,, \tag{14}$$

where $T=(T_1,\ldots,T_{N_c})$. Let $\Psi(T)=(\Psi_1(T),\ldots,\Psi_{N_c}(T))$ denote the overall chemical operator, which computes chemical equilibriums and some kinetics. The formalism of Eq. (14) is reasonable since we assume that the characteristic time of kinetics is negligible compared to that of transport processes. Let the overall transport operator be $\mathbf{L}=(L_1,\ldots,L_{N_c})$ with $L_j=L$. Now, we may write the whole coupling system in the following condensed form:

$$\text{for every component species } j,\ \left\{\begin{array}{l}\omega\partial_t T_j-L(C_j)=0\,,\\ C_j=\Psi_j(T)\,,\\ T_j=C_j+F_j\,.\end{array}\right. \tag{15}$$

3 Discretization and formulation of coupled system

3.1 Discretization by a finite volume scheme

The finite volume method [8] was chosen for treating the system (15). The time derivation term is discretized by a shift scheme of order one. The diffusion is implicit and the convection is treated by an upstream scheme and a θ method in time. We denote by $\mathcal{L}$ the discrete operator of $\mathbf{L}$. The discrete unknowns at time n are denoted by $\mathcal{C}^n=(\mathcal{C}_1^n,\cdots,\mathcal{C}_{N_c}^n)^t$, with $\mathcal{C}_j^n\in\mathbf{R}^N$. Thus, we obtain the discrete system:

$$\left\{\begin{array}{l}\mathcal{C}^{n+1}-\mathcal{L}(\mathcal{C}^{n+1})+\mathcal{F}^{n+1}=S^n\,,\\ \mathcal{C}^{n+1}=\Psi(\mathcal{T}^{n+1})\,,\\ \mathcal{T}^{n+1}=\mathcal{C}^{n+1}+\mathcal{F}^{n+1}\,,\end{array}\right. \tag{16}$$

where S^n depends on $\mathcal{C}^n$, $\mathcal{F}^n$ and the boundary conditions. This system must be solved at each time n.

3.2 Reformulation of the global system

We can reformulate the system (16) as a problem of solving a nonlinear system $\mathbf{F}(x) = 0$ the main unknown x of which could be $\mathcal{C}^{n+1}$, $\mathcal{F}^{n+1}$ or $\mathcal{T}^{n+1}$. If we choose $\mathcal{F}^{n+1}$, the nonlinear function is :

$$\mathbf{F}(\mathcal{F}^{n+1}) = \mathcal{F}^{n+1} - S^n + (id - \mathcal{L}) \circ \Psi \left((id - \mathcal{L})^{-1}(S^n - \mathcal{F}^{n+1}) + \mathcal{F}^{n+1} \right) . \quad (17)$$

Indeed, if $\mathcal{C}^{n+1}$, $\mathcal{F}^{n+1}$, $\mathcal{T}^{n+1}$ are solutions of the system (16), we have

$$\begin{aligned} \mathcal{F}^{n+1} &= S^n - (id - \mathcal{L})\,(\mathcal{C}^{n+1}) \\ &= S^n - (id - \mathcal{L}) \circ \Psi(\mathcal{T}^{n+1})\,. \end{aligned} \quad (18)$$

Note that in the pure diffusion case, the operator $id - L$ is invertible, and this is also the case in the convection diffusion case with an adequate approximation of the convection fluxes. So,

$$\begin{aligned} \mathcal{T}^{n+1} &= \mathcal{C}^{n+1} + \mathcal{F}^{n+1} \\ &= (id - \mathcal{L})^{-1}\,(S^n - \mathcal{F}^{n+1}) + \mathcal{F}^{n+1}\,. \end{aligned} \quad (19)$$

Replacing Eq. (19) in Eq. (18) leads to define $\mathbf{F}(\mathcal{F}^{n+1})$ as in Eq. (17). Proceeding in the same way, one can show that the system (16) is associated with the resolution of one of the three following nonlinear systems :

$$\mathbf{F}(\mathcal{C}^{n+1}) = \mathcal{C}^{n+1} - \Psi \left(S^n + \mathcal{L}(\mathcal{C}^{n+1}) \right) = 0\,, \quad (20)$$

$$\mathbf{F}(\mathcal{T}^{n+1}) = \mathcal{T}^{n+1} - \mathcal{L} \circ \Psi(\mathcal{T}^{n+1}) - S^n = 0\,, \quad (21)$$

$$\mathbf{F}(\mathcal{F}^{n+1}) = \mathcal{F}^{n+1} - S^n + (id - \mathcal{L}) \circ \Psi \left((id - \mathcal{L})^{-1}(S^n - \mathcal{F}^{n+1}) + \mathcal{F}^{n+1} \right) = 0\,. \quad (22)$$

We emphasize that no analytical expression of $\mathbf{F}$ is known, since one may only compute the value of $\mathbf{F}$ at point x. In the next section, we propose different algorithms to solve $\mathbf{F}(x) = 0$ which mainly use the three following primitives:

- a computation of $\mathbf{F}$ at point x, $(P1)$.
- a computation of $\nabla\mathbf{F}(x)d$, for some $d \in \mathbb{R}^N$, which can be made by a direct approximation or a numerical derivation, $(P2)$.
- a preconditioning step, $(P3)$.

The preconditioner is denoted by C.

4 Nonlinear conjugate gradient methods

4.1 Outline of the method

We propose a nonlinear conjugate gradient method with two variants which differ in the way of computing the primitive $(P2)$. Algorithm 1 states the complete method with its two variants. We outline this algorithm. At iteration (k), compute

$$\bar{y}_{(k)} = \nabla \mathbf{F}(x_{(k)}) d_{(k)} \,. \tag{23}$$

with one of the variants of primitive $(P2)$ given below, where $x_{(k)}$ is the current iterate and $d_{(k)}$ the descent direction. The stepsize in the direction $d_{(k)}$ is given by

$$\alpha_{(k)} = \frac{z_{(k)}^T d_{(k)}}{d_{(k)}^T \bar{y}_{(k)}} \,, \tag{24}$$

where $z_{(k)}$ is the residual after preconditioning. To compute the descent direction $d_{(k)}$, we use the Polak-Ribière formula [5], that is:

$$\beta_{(k)}^{PR} = \frac{z_{(k)}^T (r_{(k)} - r_{(k-1)})}{z_{(k)}^T r_{(k)}}$$

with direct restarts

$$d_{(k)} = \begin{cases} z_{(k)} & \text{if } k = 1 \\ z_{(k)} + \beta_{(k)} d_{(k-1)} & \text{else, with } \beta_{(k)} = max\left(\beta_{(k)}^{PR}, 0\right) \,, \end{cases} \tag{25}$$

which seems to be the best method in practice [5].

Remark 4.1 (Precisions on the choice of the stepsize formula) *Assume that* $\mathbf{F}$ *is the gradient of some function* $f \in C^2(\mathbb{R}^N, \mathbb{R})$*. Then finding a zero of* $\mathbf{F}$ *is related to finding an optimum to* f*. If* $d_{(k)}$ *is a descent direction for* f *in* $x_{(k)}$*, a Taylor expansion yields that:*

$$f(x_{(k)} + \alpha d_{(k)}) = f(x_{(k)}) + \alpha \nabla f(x_{(k)})^T \, d_{(k)} + \frac{\alpha^2}{2} d_{(k)}^T \, \nabla^2 f(x_{(k)}) d_{(k)} + o(\alpha^2).$$

If f *reaches its optimum in* $x_{(k)} + \alpha d_{(k)}$ *for some* α*, then we wish to have an optimum of* $\alpha \mapsto f(x_{(k)} + \alpha d_{(k)})$*; hence a reasonable choice for* $\alpha_{(k)}$ *is*

$$\alpha_{(k)} = -\frac{\nabla f(x_{(k)})^T \, d_{(k)}}{d_{(k)}^T \, \nabla^2 f(x_{(k)}) d_{(k)}} = -\frac{\mathbf{F}(x_{(k)})^T \, d_{(k)}}{d_{(k)}^T \, \nabla \mathbf{F}(x_{(k)}) d_{(k)}}.$$

4.2 Approximation of $\bar{y}_{(k)}$

We have considered two ways of computing $\bar{y}_{(k)}$ which lead to two variants of Algorithm 1 called GC_ND and GC_SPD.

- The first one GC_ND, is inspired by [3] in which $\bar{y}_{(k)}$ is computed by numerical derivation,
$$\bar{y}_{(k)} = \frac{1}{\varepsilon} \left(\mathbf{F}(x + \varepsilon d) - \mathbf{F}(x) \right) .$$
- The second one, GC_SPD, supported by a theorical result [4], uses an explicit formula,
$$\bar{y}_{(k)} = Q_{(k)} d_{(k)} ,$$
where $(Q_{(k)})_{k \in \mathbb{N}}$ is some sequence of symmetric positive definite matrix. It may be proven [4] that such a nonlinear conjugate gradient method is globally convergent when it is used for solving unconstrained optimization problems. The assumption required on $\beta_{(k)}$ includes our choice.

  The computation of $Q_{(k)}$ is performed with the $BFGS$ formula in order to preserve symmetric positive definiteness [5] [7]. Moreover, this formula is known as a way of approximating $\nabla \mathbf{F}(x_{(k)})$, in particular in unconstrained optimization. The third advantage of this formula is to provide a limited memory strategy for handling $BFGS$ updates in the case of large systems [6]. In our tests, performed in one dimension, we did not have to use limited memory strategy since the number of unknowns was not too high (< 500). Lastly, the matrix which initializes the $BFGS$ formula could behave as a preconditioning.

5 Numerical results

We compare the methods GC_ND, GC_SPD with the fixed point method (PF, Algorithm 2) and a Quasi-Newton method ($QNewton\_BFGS$, Algorithm 3). The PF method, which searches a fixed point of $x \mapsto x - C^{-1}\mathbf{F}(x)$, is in fact the steepest descent method.

We first study the influence of taking $\mathcal{C}$, $\mathcal{F}$ or $\mathcal{T}$ as the main variable of the system, that is solving one of the three formulations (20), (21) or (22) with a given algorithm. It can be either $x = \mathcal{C}$, or $x = \mathcal{F}$ or $x = \mathcal{T}$. Then, in the second subsection, we compare different algorithms.

The comparisons are performed on a one dimensional case of Portlandite leaching involving five component species. The transport operator is a pure diffusion operator with a constant diffusion parameter. The number of discrete unknowns is $N = 200$. All the chemical phenomena are under equilibrium. This case is inspired from [10] where some water in equilibrium with calcite $CaCO_3$ (pH = 7.5) is transported by diffusion in a concrete (pH = 12.5), leading to the dissolution of the Portlandite $Ca(OH)_2$ (see Figure 1). The Portlandite models the concrete. The main reaction is the following dissolution,

$$Portlandite + 2\,H^+ \Rightarrow 2\,H_2O + Ca^{2+} \quad , K = 10^{11.7} \tag{26}$$

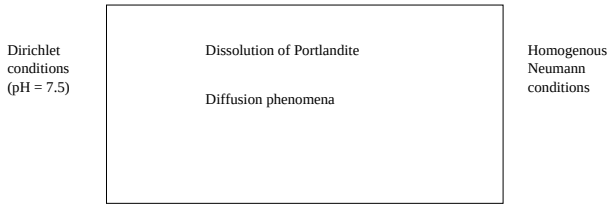

Figure 1: Portlandite leaching in 1D

The performance of the algorithms are presented for different values of the following parameter (which is sometimes called the Fourier parameter):

$$\lambda = d\frac{\Delta t}{\Delta x^2}\,,$$

where d is the diffusion coefficient, Δt the time step and Δx the spatial step. This adimensional number is essential to establish stability properties of explicit schemes applied to classical linear diffusion equation. In practice, λ is going to vary with the time step, at fixed spatial step.

5.1 Influence of the main variable

We compare the efficiency of the various algorithms taking as unknown $x = \mathcal{C}$ or $\mathcal{T}$ or $\mathcal{F}$, and varying the adimensional parameter λ. The formulation (20) in $\mathcal{C}$ denoted by (20,$\mathcal{C}$), is the fastest, see Figures 2,3,4, since the number of iterations is directly linked to the CPU time. Nevertheless, the most robust formulations are (21,$\mathcal{T}$) and (22,$\mathcal{F}$). Indeed, even if λ is increased, the algorithms converge with these formulations, however sometimes slowly. Formulation (22,$\mathcal{F}$) is robust since one computation of $\mathbf{F}(\mathcal{F})$ implies solving a linear system which plays the role of preconditioning. However, the formulation (22,$\mathcal{F}$) is often too costly in terms of the number of iteration. For large values of λ, formulation (21,$\mathcal{T}$) is competitive with (20,$\mathcal{C}$) but seems to be more robust only for $QNewton\_BFGS$ algorithm. For all other algorithms, the best formulation is (20,$\mathcal{C}$).

5.2 Comparison of the algorithms

In Table 1, the computation time of the different primitives is given. A computation of $(P2)$ with a matrix update is cheaper than a computation of $(P2)$ with numerical derivation. We call matrix update, the process of updating the $\overline{BFGS}$ matrix (resp. the $BFGS$ matrix) which arises in the $QNewton\_BFGS$ algorithm (resp. the GC_SPD algorithm). We refer as $BFGS$ updates an approximation of a Jacobian matrix and as $\overline{BFGS}$ updates an approximation of the inverse of a Jacobian matrix.

primitives	$(P1)$	$(P2)$		$(P3)$	$\overline{BFGS}$ updates
		ND	SPD updates		
average CPU time(s)	1.310^{-2}	1.210^{-2}	2.10^{-3}	10^{-3}	8.10^{-3}
relative cost	1	0.92	0.15	0.08	0.62

Table 1: Computation time of the different primitives

ND : numerical derivation, SPD updates : $BFGS$ updates

algorithm / primitives	$(P1)$	$(P2)$		$(P3)$	$\overline{BFGS}$ updates	relative cost
		ND	SPD updates			
PF	1	0	0	1	0	1
GC_ND	1	1	0	1	0	1.85
GC_SPD	1	0	1	1	0	1.14
$QNewton\_BFGS$	1	0	0	1	1	1.57

Table 2: Main computations during one iteration of each algorithm

ND : numerical derivation, SPD updates : $BFGS$ updates

In Table 2, the main computation of each algorithm during one iteration is precised. We also give the relative cost of one iteration using the average CPU time of Table 1. One iteration of GC_ND costs 1.85 iterations of PF whereas one iteration of GC_SPD costs 1.14 iterations of PF. Hence, in order to be competitive with respect to PF , GC_ND (resp. GC_SPD) must use half (resp. 85% of) the number of iterations of PF. As mentionned above, the formulation (20) in $\mathcal{C}$ denoted by (20,$\mathcal{C}$) is the most efficient in terms of iteration number and CPU time in the case of convergence. Hence this formulation is used in the comparison of the different algorithms which we now present, taking the PF algorithm as a reference (See Figure 5).

5.2.1 GC_SPD vs PF.

Numerical results of the GC_SPD method are not presented in this paper. In fact, this method works for low values of λ, but it requires over 10 times the number of iteration used by PF to reach convergence. For higher λ values, the algorithm does not seem to converge. We try to initialize the $BFGS$ update by I or $I-\mathcal{L}$ (which is symmetric definite positive if there is only diffusion); the results were in the same range. If a true step of preconditioning $(P3)$ is used, the results are not improved. This unexpected behaviour is not yet understood. As a matter of fact, the use of $BFGS$ formula to compute Q_k needs to be improved by a suitable preconditioning or initializing step.

5.2.2 GC_ND vs PF.

- GC_ND converges in three times less iterations than PF.
- From Table 2, it is clear that GC_ND is faster than PF, and this is confirmed by Figure 5.
- Both algorithms have the same slope.

5.2.3 $QNewton\_BFGS$ vs PF.

- For low values of λ (≤ 1), $QNewton\_BFGS$ may compete with PF.
- For higher values of the λ (≥ 4), the use of the $BFGS$ formula entails a high number of iterations, and therefore in this latter case PF is much more suitable.

5.3 Further tests

In our tests, GC_ND is shown to be more competitive than PF contrary to GC_SPD which is too slow. Nevertheless, we need to improve two points,

- the comprehension of GC_SPD which poorly performs whereas it seems to be promising. In particular, we will focus on the optimization of $Q_{(k)}$ matrix computations and the initializing choice of $BFGS$ updates.
- the preconditioning of GC_ND. Indeed, we see that the GC_ND method performs better than PF ; but it in these tests there is no effective preconditioning and in practical situations, we expect to need one. We performed some tests preconditioning the whole system by $C = I - \mathcal{L}$ but the results were not satisfactory. In fact, the preconditioning of the whole system is not easy since it must use information on both transport and chemistry. Hence we plan to test preconditioning the transport computations only.

Other directions of ongoing reseach include the implementaiton of a variable time step, the study and implementation of the Newton-GMRES method and the nonlinear BiCGStab method. In the case of transport phenomena involving a non-symmetric matrix, this latter method consists in formally replace the product matrix / descent direction by a numerical derivation.

Aknowledgement

This work was supported by GDR MOMAS.
web http://momas.univ-lyon1.fr/

REFERENCES

[1] Montarnal, Ph. ; Dimier, A. ; Mugler, C.; *Chemical/Transport coupling in the context of a Sofware platform*, SIAM conference on mathematical and computational issues in the geosciences, Austin, Texas, (2003).

[2] Yeh, G.T. ; Tripathi, V.S. ; *A Critical Evaluation of Recent Developments in Hydrogeochemical Transport Models of Reactive Multichemical Components*, Water Resources Research, **25**(1), 93–108, (1989).

[3] Daim, F. ; Eymard, R. ; Hilhorst, D. ; Masson, R. ; Mainguy, M ; *A preconditioned conjugate gradient based algorithm for coupling geomechanical-reservoir simulations*, Oil and Gas Science and Technology-Rev. IFP, (2002).

[4] Sun, J. ; Zhang, J. ; *Global convergence of conjugate gradient methods without line search*, Annals of Operations Research, **103** 161–173, (2001).

[5] Nocedal, J. ; Wright, S.J. ; *Numerical Optimization*, Springer, 1999.

[6] Byrd, R.H. ; Nocedal, J. ; Schnabel, R.B ; *Representations of Quasi-Newton matrices and their use in limited memory methods*, Technical Report NAM-03, Northwestern university, (1996). web www.ece.northwestern.edu/%7Enocedal/publications.html

[7] Bonnans, J.F. ; Gilbert, J.C. ; Lemaréchal, C. ; Sagastizàbal, C.A. ; *Optimisation numérique aspects théoriques et pratiques* ; Mathématiques et applications **27** , Springer Berlin (lieu d'edition), (1997).

[8] Gallouet, T ; Herbin, R ; Eymard, R ; *Finite Volume Method*, Handbook for Numerical Analysis, P.G Ciarlet, J.L. Lions Eds, North Holland, (2000).

[9] Montarnal, Ph. ; Dimier, A. ; Deville, E. ; Adam, E. ; Gaombalet, J. ; Bengaouer, A. ; Loth, L. ; Chavant, C. ; *Coupling methodology within the software platform Alliances* , Coupled problems, ECCOMAS conference, Santorini Island, Greece, May 2005.

[10] Read, D. ; Falck, W. ; *CHEMVAL 2: a coordinated research initiative for evaluating and enhancing chemical models in radiological risk assesment* , Vol. EUR 16648-EN of Nuclear Science and Technology EC Series (1996).

Algorithm 1 *GC_ND* or *GC_SPD*

Choose $x_{(1)}$ and $\varepsilon = 10^{-6}$
$x_{(k)} \leftarrow x_{(1)}$
$k \leftarrow 1$
$r_{(1)} \leftarrow -\mathbf{F}(x_{(1)})$, (P_1)
$r_{(k)} \leftarrow r_{(1)}$
while $\|r_{(k)}\|/\|r_{(1)}\| > threshold$

1. **solve** $Cz_{(k)} = r_{(k)}$, (P_3)
2. $$d_{(k)} = \begin{cases} z_{(k)} & \text{if } k = 1 \\ z_{(k)} + \beta_{(k)} d_{(k-1)} & \text{else, with } \beta_{(k)} = \left(\frac{z_{(k)}^T (r_{(k)} - r_{(k-1)})}{z_{(k)}^T r_{(k)}} \right)^+ \end{cases}$$
3. $$(P_2) \begin{cases} GC\_SPD : \begin{cases} \text{upgrade } : H_{(k+1)} = BFGS(H_{(k)}, r_{(k)} - r_{(k-1)}, x_{(k)} - x_{(k-1)}) \\ \bar{y}_{(k)} = H_{(k+1)} d_{(k)} \end{cases} \\ GC\_ND : \bar{y}_{(k)} = \frac{\mathbf{F}(x_{(k)} + \varepsilon d_{(k)}) + r_{(k)}}{\varepsilon} \end{cases}$$
4. $\alpha_{(k)} = \frac{z_{(k)}^T d_{(k)}}{d_{(k)}^T \bar{y}_{(k)}}$
5. $x_{(k+1)} \leftarrow x_{(k)} + \alpha_{(k)} d_{(k)}$
6. $r_{(k+1)} \leftarrow -\mathbf{F}(x_{(k+1)})$, (P_1)
7. $k \leftarrow k + 1$

endwhile

Algorithm 2 PF

Choose $x_{(1)}$, $k \leftarrow 1$
$r_{(1)} \leftarrow -\mathbf{F}(x_{(1)})$, $(P1)$
while $\|r_{(k)}\|/\|r_{(1)}\| > threshold$

1. **solve** $Cz_{(k)} = r_{(k)}$, (P_3)
2. $x_{(k)} \leftarrow x_{(k)} + z_{(k)}$
3. $r_{(k)} \leftarrow \mathbf{F}(x_{(k)})$, (P_1)
4. $k \leftarrow k + 1$

endwhile

Algorithm 3 $QNewton\_BFGS$

Choose $x_{(1)}$, $k \leftarrow 1$
$r_{(1)} \leftarrow -\mathbf{F}(x_{(1)})$, (P_1).
$r_{(k)} \leftarrow r_{(1)}$.
while $\|r_{(k)}\|/\|r_{(1)}\| > threshold$

1. $$H_{(k+1)}^{(-1)} = \begin{cases} I & \text{if } k = 1 \\ \overline{BFGS}(\gamma H_{(k)}^{(-1)}, r_{(k)} - r_{(k-1)}, x_{(k)} - x_{(k-1)}) & \text{else, with } \gamma = \frac{y_{(k)}^T s_{(k)}}{y_{(k)}^T H_{(k)}^{(-1)} y_{(k)}} \end{cases}$$
2. **solve** $Cz_{(k)} = H_{(k+1)}^{(-1)} r_{(k)}$, (P_3)
3. $x_{(k+1)} \leftarrow x_{(k)} + z_{(k)}$
4. $r_{(k+1)} \leftarrow -\mathbf{F}(x_{(k+1)})$, (P_1)
5. $k \leftarrow k + 1$

endwhile

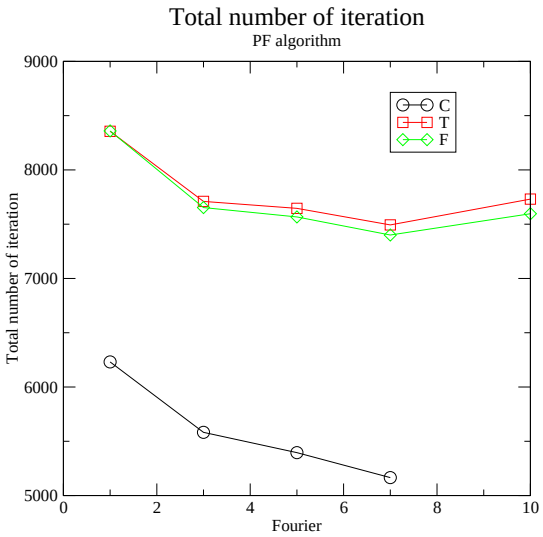

Figure 2: Influence of the Fourier parameter λ, PF algorithm

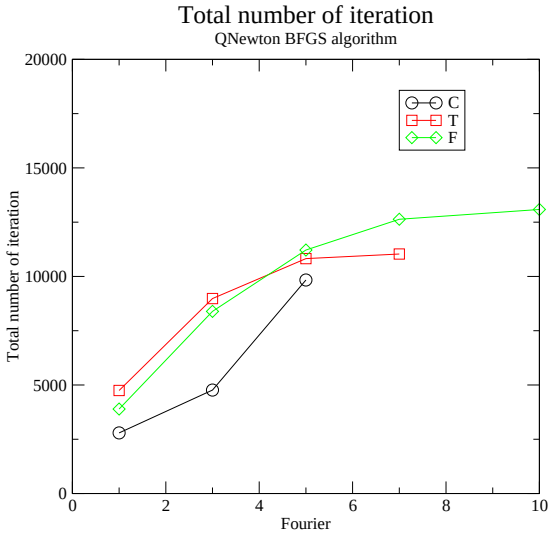

Figure 3: Influence of the Fourier parameter λ, $QNewton\_BFGS$ algorithm

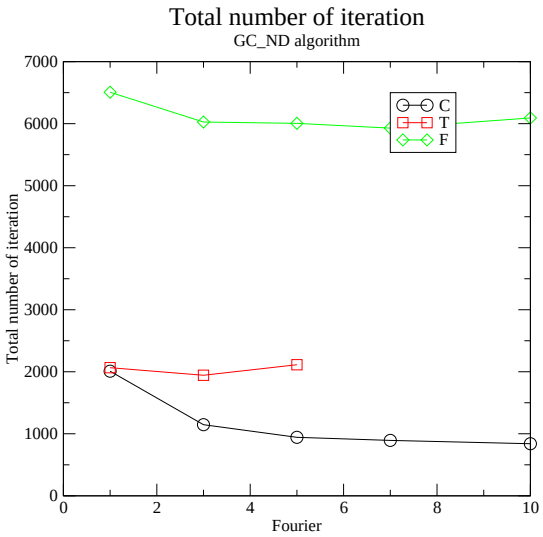

Figure 4: Influence of the Fourier parameter λ, GC_ND algorithm

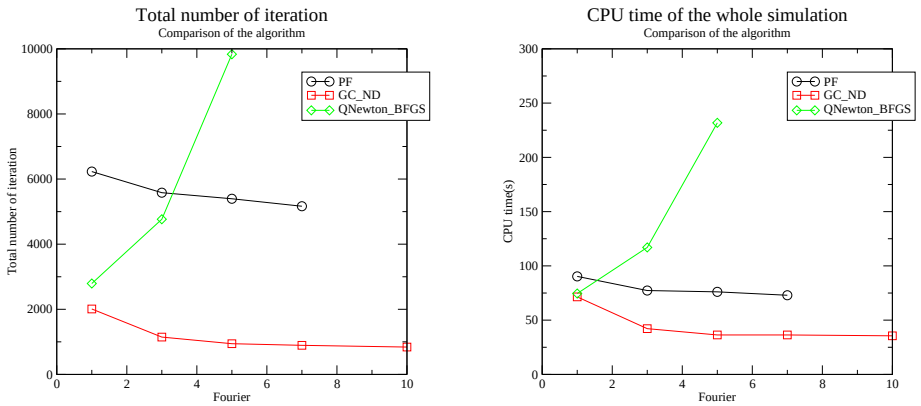

Figure 5: Comparison of the algorithms