European Journal of Computational Mechanics
Vol. 21, Nos. 3–6, June–December 2012, 374–384

Taylor & Francis
Taylor & Francis Group

A Newton–Krylov method for solid mechanics

N. Tardieu[a]* and E. Cheignon[b]

[a]*LAMSID, EDF-R&D, 1 av du Gal de Gaulle, 92141 Clamart Cedex, France;* [b]*NECS, 196 rue Houdan, 92330 Sceaux, France*

In this article a Newton–Krylov method is discussed, which is part of the inexact Newton's methods family. They combine iterative solution methods, especially Krylov methods (conjugate gradient, GMRES, etc.) with the Newton's method, whereas, in solid mechanics, direct solvers are often preferred for the solution of linearised systems. After having introduced a versatile and robust preconditioner, we discuss the efficiency of this approach on a highly non-linear industrial problem.

Dans cet article est présentée une méthode de Newton-Krylov qui appartient à la famille des méthodes de Newton inexactes. Celles-ci associent des méthodes itératives de résolution de systèmes linéaires, que sont les méthodes de Krylov (gradient conjugué, GMRES,...), à la méthode de Newton, là où, de manière traditionnelle en mécanique des solides, on utilise des méthodes directes. Après avoir décrit le préconditionneur versatile et robuste développé à cet effet, l'efficacité de l'approche est illustrée sur un problème industriel hautement non-linéaire.

Keywords: Newton's method; iterative solver; preconditioning

Mots-clés: méthode de Newton; solveur itératif; préconditionnement

1. Introduction

The solution of non-linear systems of equations is at the heart of computational physics. From electromagnetism to fluid dynamics passing by solid mechanics, the numerical solution of discretised non-linear systems of equations is an unavoidable task. The method which is often preferred is the Newton's method, due to its very interesting local convergence properties.

The Newton's method is at the root of a large family of methods which can be seen as variants that mainly try to preserve the essential quadratic convergence property and to avoid the cost by some way. Indeed, in its vanilla version, the Newton's method requires the construction of a linearised system and its resolution that is often performed in the field of solid mechanics by direct solvers. Though the latter have done tremendous progress thanks to the development of parallel multi-frontal approaches, they are often a bottleneck to the solution of very large non-linear systems. In the past decades, the main successful variant of the Newton's method in the field of solid mechanics has been the BFGS method, due to its capability to build iteratively an approximation of the inverse of the Hessian matrix. More recently, a new class of methods has appeared known as the Newton–Krylov method. It is based on the use of iterative Krylov solvers (Saad, 2005), whose adjustable resolution precision is a key

*Corresponding author. Email: nicolas.tardieu@edf.fr

ISSN 1779-7179 print/ISSN 1958-5829 online

http://dx.doi.org/10.1080/17797179.2012.721501
http://www.tandfonline.com

point that will be presented thereafter. This feature makes this approach part of the large class of inexact Newton's method.

When dealing with iterative methods, the question of choosing a good preconditioner must be answered. In the field of solid mechanics, there are several optimal algorithms that provide highly efficient preconditioners. In the domain decomposition framework, let us cite the dual sub-structuring method FETI (Farhat & Roux, 1991) with several variants (Dostal, Horak, & Kucera, 2006; Farhat, Lesoinne, LeTallec, Pierson, & Rixen, 2001) and the primal sub-structuring method BDD (Mandel, 1993). In the multi-grid framework, let us highlight a geometric approach (Adams, 1999) and an algebraic one (Vanek, Mandel, & Brezina, 1996). However, they often need to be specially adapted for some given physical equations and finite elements (plates, shells and incompressible displacement–pressure element). In other words, finding a "catch-all" optimal preconditioner is a hard task. In this paper, we propose a broad-spectrum tool which, though far from optimal, proves to be fast and robust; thanks to an updating adaptive strategy.

The paper is organised as follows. In Section 1, the basics of the Newton's method are recalled. In Section 2, the inexact Newton's method characteristics are described with special emphasis on the Newton–Krylov approach. The importance of the preconditioning phase is stressed leading to the proposition of a robust and versatile mixed precision preconditioner. The last section is devoted to the illustration of the performance of the method on a non-linear industrial problem.

2. The Newton's method

In this section are recalled the essential requirements and properties of the Newton's method.

2.1. Overall presentation

The Newton's method is a method for solving non-linear systems of equations written as:

$$f(\bar{x}) = 0 \tag{1}$$

Let us define U an open subset of $\mathbb{R}^n$, $f \in C^1(U, \mathbb{R}^n)$ and $\bar{x} \in U$ such that $f(\bar{x}) = 0$. From a given starting point $x_0 \in U$, the method builds a sequence $(x^{(k)})_{k\in\mathbb{N}} \subset U$ of approximations of the solution $\bar{x}$. Given two close enough approximations, the first-order Taylor expansion of f at $x^{(k+1)}$ is written as:

$$f(x^{(k+1)}) = f(x^{(k)}) + \nabla f(x^{(k)})(x^{(k+1)} - x^{(k)}) + o(x^{(k+1)} - x^{(k)}) \tag{2}$$

Let us set $f(x^{(k+1)}) \simeq 0$, so that the sequence $(x^{(k)})_{k\in\mathbb{N}}$ is defined as follows:

$$\begin{cases} x^{(0)} = x_0 \\ f(x^{(k)}) + \nabla f(x^{(k)})(x^{(k+1)} - x^{(k)}) = 0 \end{cases} \tag{3}$$

where

$$\nabla f(x) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(x) & \cdots & \frac{\partial f_1}{\partial x_n}(x) \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1}(x) & \cdots & \frac{\partial f_n}{\partial x_n}(x) \end{pmatrix}$$

The process stops at iteration p when the convergence criterion is reached, i.e. for a given ε precision:

$$\| f(x^{(p)}) \| \leq \varepsilon \| f(x^{(0)}) \| \tag{4}$$

For each iteration k, the algorithm consists of the following steps:

(1) compute the matrix $A = \nabla f(x^{(k)})$,
(2) solve the linear system and

$$A\, s^{(k)} = -f(x^{(k)}), \quad x^{(k+1)} = x^{(k)} + s^{(k)} \tag{5}$$

(3) evaluate $f(x^{(k+1)})$.

Note that there is no guarantee that the matrix $\nabla f(x^{(k)})$ is invertible or that the sequence $x^{(k)}$ converges to $\bar{x}$. But, in a neighbourhood of $\bar{x}$ and under some regularity assumptions, the Newton's method converges quadratically (Kelley, 2003).

2.2. Line search

From a practical point of view, having a local convergence property does not suffice since there is no guarantee that the starting point x_0 is close enough to $\bar{x}$. In order to recover a global convergence property, globalisation methods such as the line search must be used.

The line search aims at making the Newton's method more robust in the case where the starting point is not close enough to the solution. Once the linear system has been solved, the solution increment direction $s^{(k)}$ is known. The line search consists of finding the next iterate $x^{(k+1)} = x^{(k)} + \theta s^{(k)}$ with $\theta \in \mathbb{R}_*^+$ such that θ minimises $\| f \|$. From a practical point of view, in order to perform this minimisation, one writes:

$$\phi_k(\theta) = \| f(x^{(k)} + \theta s^{(k)}) \| .$$

and one searches for a value of $\theta > 0$ providing a sufficient decrease of ϕ_k. We insist on the fact that this phase can be very expensive in the case where lots of evaluations of f are necessary.

3. Inexact Newton's methods

The goal of the inexact Newton's methods is to reduce the computation time by limiting the number of total operations. To this end, one can seek to lessen the cost of the construction of the Jacobian matrix and/or the solution cost of the linear system. The latter is at the root of Newton–Krylov methods.

3.1. Sketch of the method

The base idea of the inexact Newton's methods is to replace the solution of (5) by a weaker requirement:

$$\| \nabla f(x^{(k)})s^{(k)} + f(x^{(k)}) \| \leq \eta_k \| f(x^{(k)}) \| \tag{6}$$

The general algorithm is exactly the same as the regular Newton's method; the unique difference being that the linearised system is not solved exactly. The iterate $s_{(k)}$ is only an approximation of the solution of (5). The parameter η_k, called the *forcing term* plays a crucial role. Notice that the closer η_k is to zero, the closer the solution of (6) is to the solution of (5). One can intuitively feel that computing a crude approximate solution (e.g. for a large value of η_k) can be fast but it can lead to convergence issues at the non-linear stage. The key-point of inexact Newton's method is therefore to find the best compromise between the faster solution of the linear system and the fewest number of non-linear iterations.

3.2. Convergence property

Theorem 1 (Local convergence of the inexact Newton's method).
Given $f : U \subset \mathbb{R}^n \mapsto \mathbb{R}^n$ such that:

(1) *the equation (1) has a solution $\bar{x}$,*
(2) *∇f: $U \mapsto \mathbb{R}^{n \times n}$ is Lipschitzian in a neighbourhood of $\bar{x}$,*
(3) *$\nabla f(\bar{x})$ is invertible,*

then there exist two positive scalar δ et $\bar{\eta}$, with $\bar{\eta}<1$, such that $\forall x_0 \in B(\bar{x}, \delta)$ and $\{\eta_k\} \subset [0, \bar{\eta}]$, the sequence $(x^{(k)})_{k \in N}$ defined by:

$$x^{(k+1)} = x^{(k)} + s^{(k)}$$

and by the condition (6) converges linearly.
Furthermore, if $\eta_k \to 0$, the convergence is superlinear.
The proof can be found in Kelley (1995, pp. 96–97).

3.3. The Newton–Krylov methods

As their name implies, the Newton–Krylov methods are methods making simultaneous use of Krylov methods within a Newton algorithm. We shall briefly recall the essential aspects of the Krylov methods.

3.3.1. The Krylov methods

Let us consider the solution of a well-posed linear system $Ax = b$. There are two classes of iterative methods to solve this system: the stationary methods and the Krylov methods. The stationary methods are defined by the following iteration:

$$x_{(k+1)} = Mx_{(k)} + c \tag{7}$$

where M is called the iteration matrix. The expression of this iteration matrix fully defines the method. Given the splitting $A = D + L + U$ when D is the diagonal of A, L and U are the strict lower and upper parts of A. If $M = -D^{-1}(L + U)$, this is the Jacobi method, if $M = -(D + L)^{-1}U$, this is the Gauss–Seidel method. The *stationary* terminology comes from the fact that the iteration matrix is fully independant of the path to convergence. This is not the case for the Krylov methods.

The Krylov methods iteratively build a sequence $x_{(k)}$ that minimises the error $e = \bar{x} - x_{(k)}$ in some given measure over the affine space $x_{(0)} + \mathcal{K}_k$ where $\mathcal{K}_k$ is the vector space:

$$\mathcal{K}_k = span(r_{(0)}, Ar_{(0)}, A^2 r_{(0)}, \cdots, A^{k-1} r_{(0)}) \tag{8}$$

where $r_{(0)} = b - Ax_{(0)}$. $\mathcal{K}_k$ is called the Krylov subspace. It can be easily shown that these vectors are linearly independant so that when $k=n$, they form a base of $\mathbb{R}^n$. This argument is essential in order to prove the convergence of Krylov methods within n iterations. Unfortunately, round-off errors and bad conditioning make this situation almost impossible for reasonable large n. Despite this, Krylov methods are methods of choice for solving linear systems and they have often replaced the stationary methods.

As the Krylov methods are iterative methods, they need a convergence criterion to stop the process. Usually, it is based on the relative decrease of the linear residual to a fixed value, typically 10^{-6}, which is written as:

$$\| r_{(k)} \| = \| Ax_{(k)} - b \| < 10^{-6} \| r_{(0)} \| \tag{9}$$

3.3.2. *Choice of the "forcing term"*

It must be noticed that the latter equation is equivalent to the inexact Newton condition (6). Thus one understands that the previous condition can be used as the stopping criterion of the linear Krylov solver, which is completely determined by the choice of the "forcing term". A good choice is essential for the performance of the global non-linear process.

Different schemes exist in order to compute the sequence of forcing terms η_k starting from an initial η_0. The basic idea is the following: when far from the solution, it is not necessary to compute a precise linear solution; when getting closer to the solution, in order to recover the quadratic convergence of the Newton's method, the linear solution must be precise. Kelley (2003) and Eisenstat and Walker (1996) proposed the following sequence:

$$\eta_{k+1}^{\text{Res}} = \gamma \frac{\| f(x^{(k)}) \|^2}{\| f(x^{(k-1)})) \|^2} \tag{10}$$

where $\gamma \in (0, 1]$ is a user-defined parameter. But in order to avoid a too fast decrease of the sequence η_k causing unnecessarily accurate resolutions or to get unacceptable values of $\eta_k > 1$, Equation (10) must be reinforced by the following condition (Kelley, 2003):

$$\eta_{k+1} = \min\left(\eta_{\max}, \max\left(\eta_{k+1}^{\text{Safe}}, 0.5\varepsilon \frac{\| f(x^{(k)}) \|^2}{\| f(x^{(k-1)})) \|^2}\right)\right) \tag{11}$$

where ε is the precision of Equation (4) and η_k^{Safe} is given by:

$$\eta_{k+1}^{\text{Safe}} = \begin{cases} \eta_{\max} & k = 0 \\ \min(\eta_{\max},\ \eta_{k+1}^{\text{Res}}) & k>0, \quad \gamma\eta_k^2 \leq 0.1 \\ \min(\eta_{\max}, \max(\eta_{k+1}^{\text{Res}}, \gamma\eta_k^2)) & k>0, \quad \gamma\eta_k^2 > 0.1 \end{cases}$$

3.3.3. *Oversolving*

In the context of Newton–Krylov methods, to *oversolve* is to solve the linear system with an unnecessary precision, that is to say without decreasing the non-linear residual $\| f(x^{(k)}) \|$ in the same proportion. Indeed, the same decrease of the non-linear residual can often be

obtained with a coarse linear solve as well as with a precise one. In that case, lots of Krylov iterations may have been useless.

The goal of the Newton–Krylov methods is to always adapt the value of the forcing term η_k in order, on the one hand, to lessen the number of Krylov iterations and, on the other hand, not to increase the number of Newton iterations.

In order to illustrate this issue, one time step of the simulation shown in Section 5.1 has been run in two different ways: in a first time, using the Newton's method with a preconditioned conjugate gradient (CG), the relative precision of the linear solution being constant and set to 10^{-6}; in a second time, using the same preconditioned CG but with the relative precision varying according to (12).

In order to understand the following results, it is necessary to establish precisely the difference between the Newton's iterations and the Krylov iterations. The Newton's method builds a sequence $(x^{(k)})_{k\geq 0}$ of approximations of $\bar{x}$. At each iteration k, the Krylov method builds the sequence $(s_i^{(k)})_{i\geq 0}$ of approximations of s, solution of $\nabla f(x^{(k)})s = -f(x^{(k)})$. We denote then $x_i^{(k)} = x^{(k)} + s_i^{(k)}$. The Krylov iterations stop when the convergence criterion (6) is reached. If one defines p, the number of iterations when the convergence criterion is satisfied, we have $x_p^{(k)} = x^{(k+1)}$.

In Figure 1, the stars represent the successive values of the non-linear residual $f(x^{(k)})$. The solid lines represent the values of the linear residual $\nabla f(x^{(k)})s_i^{(k)} + f(x^{(k)})$ and dotted lines represent a rough estimate of $f(x_i^{(k)})$. It is clearly seen that the precise resolution of the linear system is unnecessary because the non-linear residual does not decrease more on the left figure (CG with fixed convergence criteria) than on the right figure (CG with adaptive convergence criteria). The following table summarises the number of iterations in each case.

	CG	CG + forcing term
No. of Newton's iterations	6	7
No. of CG iterations	83	11

This shows clearly the efficiency of the proposed expression of the forcing term in order to avoid oversolving. The number of Krylov iterations has been considerably reduced while conceding a single extra Newton iteration.

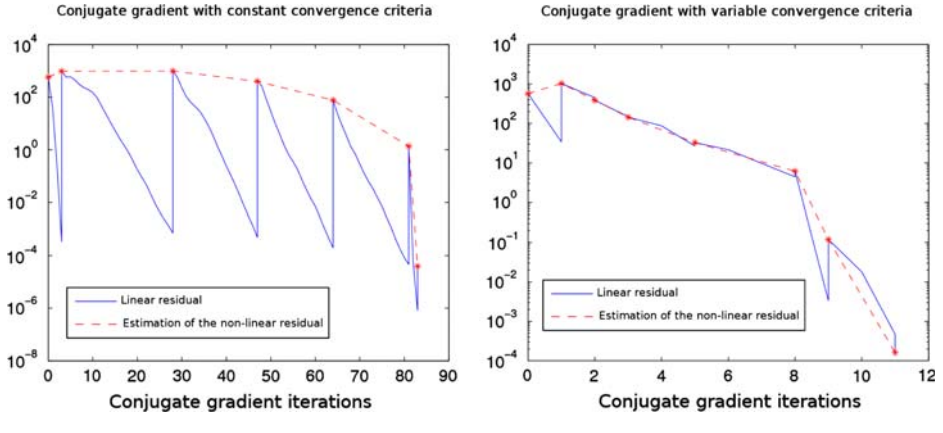

Figure 1. Illustration of the oversolving phenomenon.

4. Solution strategy

4.1. Preconditioning

In order to use a Newton–Krylov method, one must begin by choosing a preconditioner and an iterative method. Depending on the symmetry of the matrix of the system, our choice of iterative method was set to the CG or to the generalised minimum residual (GMRES). But the main issue is in fact the choice of an efficient preconditioner. In the field of solid mechanics, there are several optimal algorithms that provide highly efficient preconditioners. In the domain decomposition framework, let us cite the dual substructuring method FETI (Farhat & Roux, 1991) with several variants (Dostal et al., 2006; Farhat et al., 2001) and the primal substructuring method BDD (Mandel, 1993). In the multi-grid framework, let us highlight a geometric approach (Adams, 1999) and an algebraic one (Vanek et al., 1996).

However, in the case of industrial studies in solid mechanics, the models are often composed of different kinds of finite elements. It is common to deal with a mix of solid and structural elements (shells and beams for instance). Furthermore, in order to correctly transfer the displacements and the forces between these elements with different degrees of freedom (shells and beams are equipped with rotational degrees of freedom), multi-points constraints must be used. A common tool to enforce such relationships is the use of Lagrange multipliers that can deeply change the nature of the linear system: in the case of linear elasticity, it causes the loss of the positive definiteness. In short, the linear systems in solid mechanics are often non-positive definite and generally very ill-conditioned. Finding a robust preconditioner is a hard task. The classical algorithms such as Jacobi, Gauss–Seidel, incomplete factorisation and also the more modern previously cited algorithms generally fail or perform very poorly.

For these reasons, we have developed a preconditioner for the open-source finite element software *Code_Aster* (2011a) called `LDLT_SP` (2011b). This preconditioner is based on the parallel single-precision factorisation of the tangent (stiffness) matrix, while the iterative method is performed in double precision. During the preconditioning phase, the effect of the inverse of the matrix A is very accurately approximated, since the factorisation is exact up to single precision. Then, just 2 or 3 iterations of the Krylov method are needed in order to reach convergence.

It is noteworthy that the use of single precision reduces the memory consumption by 50% compared to a double precision factorisation. But it also reduces the CPU time by 30% – more data can be stored on the cache of the processor leading to fewer cache misses and better performance. Nevertheless, computing this preconditioner at each Newton iteration is expensive both in memory and CPU time compared to classical preconditioning methods. But there is no need of reforming it so often – this preconditioner is so efficient that it can be kept constant for several solution phases even if the tangent matrix changes. This strategy is the key of the performance of the method:

- The quadratic convergence of the Newton's method is kept since the tangent matrix is updated at each iteration.
- The application of the previously computed preconditioner is fast since it consists only in a forward–backward substitution.
- The preconditioner is only updated when its efficiency decreases. The number of Krylov iterations is an excellent indicator since it is closely related to the conditioning of the preconditioned system (Shewchuk, 1994): as soon as more than, say 100, iterations are needed to solve the linear system, the preconditioner must be reformed.

Within this adaptive framework, it provides an extremely robust fully-algebraic preconditioner.

4.2. Forcing term

Each iteration of the Newton–Krylov method implies the integration of the constitutive law, the computation of the tangent matrix and the solution of the linearised system. From our experience, the expression of the forcing term (11) can cause a quite slow convergence of the non-linear process and thus be computationally costly.

Following this observation, we propose another expression of the forcing term. According to our experience, it ensures a safer convergence, at the cost of a little oversolving. So we define the following expression:

$$\eta_{k+1}^{\text{Safe}} = \begin{cases} \eta_0 & k = 0 \\ \max(\min(\eta_k/2, \eta_{k+1}^{\text{Res}}), \eta_{\min}) & k>0, \quad \gamma\eta_k^2 \leq 0.1 \\ \min(\eta_k/2, \max(\eta_{k+1}^{\text{Res}}, \gamma\eta_k^2)) & k>0, \quad \gamma\eta_k^2>0.1 \end{cases} \tag{12}$$

Finally, the following parameters $\gamma = 0.9$ and $\eta_0 = 0.9$ were chosen, as in Eisenstat and Walker (1996) and Kelley (2003), and $\eta_{\min} = 10^{-4}$.

5. Results

This section presents the application of the proposed Newton–Krylov method in order to evaluate its performance and its robustness.

5.1. Moisture separator reheater

This calculation is a problem of solid mechanics; it models an equipment of nuclear power plants called the moisture separator reheater, whose function is to improve the quality of the water vapour in the secondary circuit before re-injection. The material of the reheater obeys

Table 1. Comparison of the three solution approaches.

	Resolution time		
Linear solver	MUMPS	PETSc + LDLT_SP	PETSc + LDLT_SP + Forcing term
Resolution time	9176 s	8530 s	5554 s
CPU time gain/MUMPS	/	7%	39%

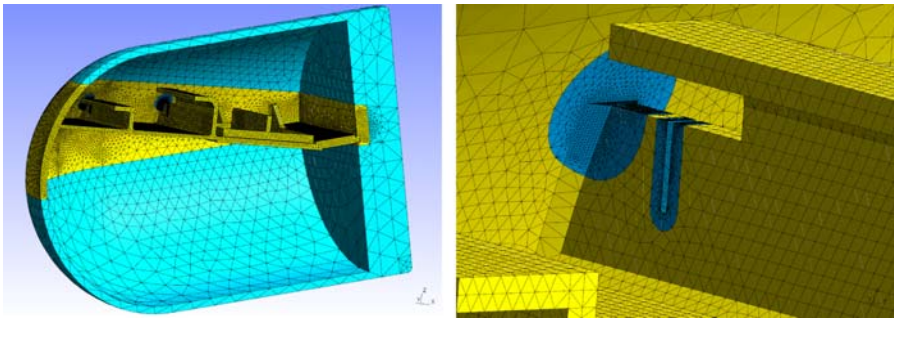

Figure 2. Mesh of the moisture separator reheater.

an elasto–plastic constitutive law with isotropic non-linear hardening; a large deformation kinematics is used. The tetrahedral mesh is of order 2 and generates a problem of 1,084,773 unknowns (Figure 2). The tangent matrix of this problem is symmetric.

In order to perform some comparisons, we first started the calculation with the classical Newton's method, using the direct solver `MUMPS` (Amestoy, Guermouche, L'Excellent, & Pralet, 2006) as a linear solver, then the CG of the library `PETSc` (Balay et al., 2011) preconditioned by `LDLT_SP` and finally with the Newton–Krylov method using the CG of `PETSc` with `LDLT_SP`. The simulations were done on a parallel cluster using eight processors. Table 1 shows the different computation times and the gains compared to the calculation with `MUMPS`. Figures 3 and 4, respectively, indicate the number of Newton's iterations made by the three calculations and the number of CG iterations for each time step by the last two calculations.

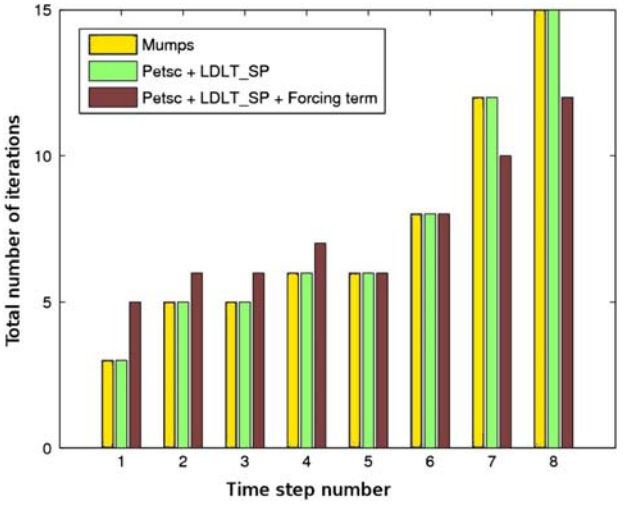

Figure 3. Comparision of the number of Newton's iterations.

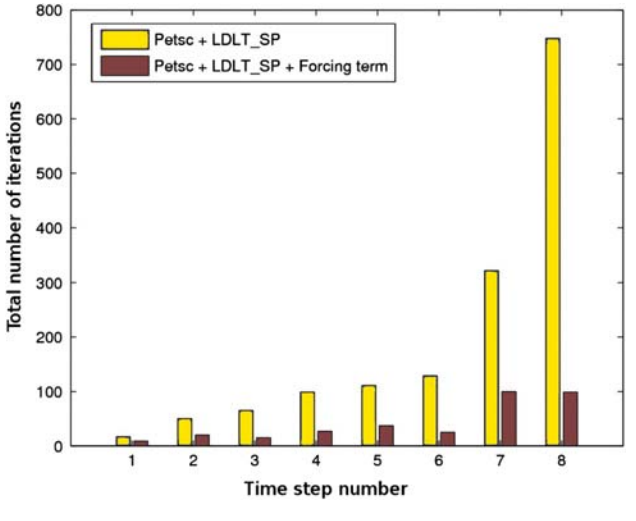

Figure 4. Comparision of the number of Krylov iterations.

In Table 1, we observe the very low gain provided by the use of the iterative solver PETSc+LDLT_SP compared to the use of the MUMPS direct solver. This is quite surprising given the non-linear problem is quite large. Some explanations are given below.

On the contrary, the gain of the inexact approach PETSc+LDLT_SP+Forcing term is quite significant. In Figure 3, we note that the inexact method needs more Newton's iterations on the first time step than the exact one, but, from the fifth time step, the number of iterations is equal to that of the other two methods and it becomes even lower from step 7. This situation is described in the literature (Eisenstat & Walker, 1996); it is due to the fact that an accurate resolution of the linearised system far from equilibrium can be counterproductive. The number of Krylov iterations is thereby decreased (see Figure 4).

Let us give some more insight on the CPU time of each solution phase:

```
MUMPS within the Newton's method :
   NUMBER OF NEWTON ITERATIONS            :     60
   NUMBER OF CONST. LAW INTEGRATION       :    168
   TIME FOR MATRIX FACTORISATIONS         :  1 h 46 m  1 s
   TIME FOR CONST. LAW INTEGRATION        : 19 m  9 s
   TIME FOR LINEAR SOLVES                 :  7 m 10 s

Conjugate gradient within the Newton's method :
   NUMBER OF NEWTON ITERATIONS            :     60
   NUMBER OF CONST. LAW INTEGRATION       :    168
   TIME FOR MATRIX PRECONDITIONING        : 38 m 43 s
   TIME FOR CONST. LAW INTEGRATION        : 18 m 52 s
   TIME FOR LINEAR SOLVES                 :  1 h  6 m 47 s

Conjugate gradient within the Newton-Krylov method :
   NUMBER OF NEWTON ITERATIONS            :     60
   NUMBER OF CONST. LAW INTEGRATION       :    146
   TIME FOR MATRIX PRECONDITIONING        : 38 m 55 s
   TIME FOR CONST. LAW INTEGRATION        : 16 m 44 s
   TIME FOR LINEAR SOLVES                 : 19 m  7 s
```

We first consider the statistics of the use of the direct and the iterative method within the Newton's method. From that, the respective merits of each solvers are obvious:

- The cost of the direct methods is related to the factorisation of the matrix, the resolution itself (backward forward substitution) is very fast.
- The cost of iterative methods is related to the resolution of the linear system, the preconditioning is quite fast.

Now we consider the statistics of the iterative method within the Newton–Krylov method; it can be noted that:

- The low cost of the preconditioning phase is kept as in the case of an iterative method within the Newton's method.
- The low cost of the solution phase is kept as in the case of an direct method within the Newton's method.

That is, the Newton–Krylov method is a synthesis of the strengths of the previous approaches. Thanks to this property, the Newton–-Krylov method exhibits good performances on large problems.

6. Conclusions

This paper presents an application of an inexact Newton method on an industrial problem of non-linear solid mechanics. To carry out this study, we have proposed a specific expression of the forcing term and a robust fully-algebraic preconditioner. Thus equipped, our Newton–Krylov method proved to be both efficient and robust on the studied very non-linear problem. Other non-linear calculations, including thermo-hydro-mechanics, have been successfully completed using this method.

When solving a non-linear problem, there are three different CPU time consuming phases: the integration of the constitutive law, the preconditioning/factorisation phase and the solution phase. It has been shown in this paper that the proposed method saves computation time both on the preconditioning/factorisation phases (compared to a conventional Newton's method with a direct solver) and the solution phase (compared to a conventional Newton's method with an iterative solver). Thus, if most of the time of a simulation is spent in the integration of the constitutive law, large time savings cannot be expected. Therefore, this Newton–Krylov method is suitable for large problems, where most of the time is spent in the preconditioning and solution phases.

References

Adams, M. (January 1999). *Parallel multigrid algorithms for unstructured 3d large deformation elasticity and plasticity finite element problems*. Technical Report UCB/CSD-99–1036, EECS Department, University of California, Berkeley.

Amestoy, P.R., Guermouche, A., L'Excellent, J.-Y., & Pralet, S. (2006). Hybrid scheduling for the parallel solution of linear systems. *Parallel Computing, 32*(2), 136–156.

Balay, S., Brown, J., Buschelman, K., Gropp, W.D., Kaushik, D., Knepley, … Zhang, H. (2011). *PETSc Web page*. Retrieved from http://www.mcs.anl.gov/petsc

Code_Aster. (2011a). Code_Aster web page. Retrieved from http://www.code-aster.org

Code_Aster. (2011b). *Overview of the conjugate gradient: GCPC and use of PETSc*. Retrieved from http://www.code aster.org/V2/doc/default/en/man r/r6/r6.01.02.pdf

Dostal, Z., Horak, D., & Kucera, R. (2006). Total FETI – an easier implementable variant of the FETI method for numerical solution of elliptic PDE. *Communications in Numerical Methods in Engineering, 22*(12), 1155–1162.

Eisenstat, S.C., & Walker, H.F. (1996). Choosing the forcing terms in an inexact Newton method. *SIAM Journal on Scientific Computing, 17*(1), 16–32.

Farhat, C., Lesoinne, M., LeTallec, P., Pierson, K., & Rixen, D. (2001). FETI-DP: A dual-primal unified FETI method. I. A faster alternative to the two-level FETI method. *International Journal for Numerical Methods in Engineering, 50*(7), 1523–1544.

Farhat, C., & Roux, F.-X. (1991). A method of finite element tearing and interconnecting and its parallel solution algorithm. *International Journal for Numerical Methods in Engineering, 32*(6), 1205–1227.

Kelley, C.T. (1995). Iterative methods for solving linear and nonlinear equations. In: *Frontiers in applied mathematics*, Vol. 16. Philadelphia, PA: SIAM.

Kelley, C.T. (2003). *Solving nonlinear equations with Newton's method*. Philadelphia, PA: SIAM.

Mandel, Jan. (1993). Balancing domain decomposition. *Communications in Numerical Methods in Engineering, 9*, 233–241.

Saad, Y. (2005). *Iterative methods for sparse linear systems*. Philadelphia, PA: SIAM.

Shewchuk, J.R. (1994). *An introduction to the conjugate gradient method without the agonizing pain*. Technical Report, University of California at Berkeley.

Vanek, P., Mandel, J., & Brezina, M. (1996). Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. *Computing, 3*, 179–196.