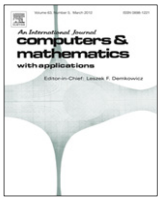
Contents lists available at ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa



Iterative residual-based vector methods to accelerate fixed point iterations

Isabelle Ramière [*], Thomas Helfer

CEA, DEN, DEC, SESC, F-13108 Saint-Paul Lez Durance, France

ARTICLE INFO

Article history:
Received 11 May 2015
Received in revised form 20 July 2015
Accepted 20 August 2015
Available online xxxx

Keywords:
Fixed point method
Acceleration iterative technique
Vector sequences
Residual-based method
Nonlinear mechanics
Mechanical behaviours

ABSTRACT

Fixed point iterations are still the most common approach to dealing with a variety of numerical problems such as coupled problems (multi-physics, domain decomposition, ...) or nonlinear problems (electronic structure, heat transfer, nonlinear mechanics, ...). Methods to accelerate fixed point iteration convergence or more generally sequence convergence have been extensively studied since the 1960's. For scalar sequences, the most popular and efficient acceleration method remains the Δ^2 of Aitken. Various vector acceleration algorithms are available in the literature, which often aim at being multi-dimensional generalizations of the Δ^2 method.

In this paper, we propose and analyse a generic residual-based formulation for accelerating vector sequences. The question of the dynamic use of this residual-based transformation during the fixed point iterations for obtaining a new accelerated fixed point method is then raised. We show that two main classes of such iterative algorithms can be derived and that this approach is generic in that various existing acceleration algorithms for vector sequences are thereby recovered.

In order to illustrate the interest of such algorithms, we apply them in the field of nonlinear mechanics on a simplified "point-wise" solver used to perform mechanical behaviour unit testings. The proposed test cases clearly demonstrate that accelerated fixed point iterations based on the elastic operator (quasi-Newton method) are very useful when the mechanical behaviour does not provide the so-called consistent tangent operator. Moreover, such accelerated algorithms also prove to be competitive with respect to the standard Newton–Raphson algorithm when available.

1. Introduction

Fixed point iterations (also called Picart iterations) are used in many applications to deal with nonlinear equations. In fact, it is well-known that any nonlinear system of equations $F(X) = 0$ with $F : \mathbb{R}^N \to \mathbb{R}^N$ is equivalent to a fixed point problem $G(X) = X$ with $G : \mathbb{R}^N \to \mathbb{R}^N$. The fixed point iterations method is therefore the simplest way to obtain nonlinear solutions without any *a priori* knowledge about F, its derivative in particular is not required. On the other hand, fixed point iterations converge generally only linearly and very slowly. This is the reason why acceleration algorithms are required. As the fixed point iterations method generates a sequence, classical sequence acceleration methods can be applied. Many of such acceleration or extrapolation algorithms for vector sequences were proposed in the 20th century [1], especially in the 1960's (e.g. [2–5]) and in the 1990's (e.g. [6–9]). Most of them are generalizations of the powerful Aitken's Δ^2 scalar acceleration algorithm [10]. Applications of such vector acceleration techniques for the fixed point iterations method are various and diverse. For example, fixed point acceleration methods have recently enjoyed a renewed interest in the

[*] Corresponding author.
E-mail addresses: isabelle.ramiere@cea.fr (I. Ramière), thomas.helfer@cea.fr (T. Helfer).

http://dx.doi.org/10.1016/j.camwa.2015.08.025
0898-1221/© 2015 Elsevier Ltd. All rights reserved.

development of advanced numerical methods (domain decomposition [11,12], multigrid [13,14], ...) and of multiphysics coupling in a black-box context (fluid–structure interaction [15,16], thermomechanics [17,18], ...).

The aim of this article is to propose a common framework to build new vector acceleration methods. This formalism is build as a generalization of the nonlinear hybrid acceleration procedure [9] but can also be viewed as an extension of both the reduced rank extrapolation method [19] and the Anderson's method [4]. By the way, some of the most popular and efficient extrapolation methods can be recovered using this generic approach.

As no function derivative is involved in the proposed approach, this unified formalism can be applied on various type of problems as for example black box solver systems. Moreover, by construction, high-dimensionality problems can also be treated within this formalism.

The paper is organized as follows. We first propose in Section 2 a survey of most popular vector acceleration methods in a common fixed point sequence framework. In particular, we present them as iterative methods that consist in dynamically generating a new accelerated fixed point sequence with the transformed iterates. Then, we introduce in Section 3 an extension of the nonlinear hybrid acceleration procedure [9] in order to deal with linear combinations of fixed point iteration residuals. We show in Section 4 that two main classes of iterative (or dynamic) vector acceleration methods can be derived from this generic residual-based formulation. Various already known acceleration methods are retrieved within these two classes. Finally, Section 5 is dedicated to some performance comparisons between acceleration methods stem from these two formulations. The application concerns a simplified "point-wise" solver, developed within the PLEIADES platform [20] as part of the MFront software [21], which is mainly used to perform mechanical behaviour unit testings. This solver allows us, among other things, to compare our accelerated algorithms to the standard Newton–Raphson algorithm used by default in most mechanical finite element solvers.

2. Survey of iterative vector acceleration methods

The aim of this section is to briefly describe most of the existing vector sequence acceleration methods. As it is pointed out in the literature (see for example [4,6]), static sequence transformations from an existing sequence are less efficient than dynamic or cycling sequence transformations that consist in considering the accelerated iterate as the new iterate of the sequence. Therefore, we shall only present here the iterative version of the vector acceleration methods.

Moreover, for the sake of clarity, we shall focus on fixed point iteration sequences as they are the most common sequences dealt with in an engineering context (even the Newton's method generates a fixed point sequence). Within this fixed point iteration framework, expressions of existing vector sequence acceleration approaches can be more easily compared.

As a universal acceleration method for all types of sequences cannot exist (see for example [6]), in this paper we shall only consider transformations that have been designed to accelerate first order (or linearly converging) sequences. Only few authors proposed some approaches to accelerate sequences of order greater than one, the interested reader is referred to [22,9].

2.1. Scalar sequences

Let us begin this survey be the scalar case, which is the basis of most of the vector acceleration methods. Considering a scalar nonlinear fixed point equation

$$x = g(x), \quad x \in \mathbb{R}, \qquad g : \mathbb{R} \to \mathbb{R}, \tag{1}$$

the most popular and powerful acceleration method of the basic fixed point substitution iteration

$$x_{n+1} = g(x_n) \tag{2}$$

remains the Δ^2 of Aitken [10] and its recursive application by the Steffensen algorithm which lead to a second-order method (e.g. [23,24]):

$$x_{n+1} = x_n - \frac{(g(x_n) - x_n)^2}{g(g(x_n)) - 2g(x_n) + x_n} = x_n - \frac{(\Delta x_n)^2}{\Delta^2 x_n} \tag{3}$$

with Δ the difference operator, $\Delta x_n = g(x_n) - x_n$, $\Delta g(x_n) = g(g(x_n)) - g(x_n)$ and $\Delta^2 x_n = \Delta g(x_n) - \Delta x_n$.

Various equivalent formulations exist, including for example

$$x_{n+1} = g(x_n) - \frac{\Delta x_n \Delta g(x_n)}{\Delta^2 x_n} \tag{4}$$

or

$$x_{n+1} = g(g(x_n)) - \frac{(\Delta g(x_n))^2}{\Delta^2 x_n}. \tag{5}$$

Whatever the expression, each Steffensen iteration requires two new basic fixed point iterations (or function evaluations) to be applied. The secant method (see [23] for example) can also be viewed as a Δ^2 transformation requiring only one new fixed point iteration

$$x_{n+1} = x_n - \frac{(x_n - x_{n-1})(g(x_n) - x_n)}{(g(x_n) - x_n) - (g(x_{n-1}) - x_{n-1})} = x_n - \frac{(x_n - x_{n-1})\Delta x_n}{\Delta x_n - \Delta x_{n-1}} \tag{6}$$

or equivalently

$$x_{n+1} = g(x_n) - \frac{(g(x_n) - g(x_{n-1}))\Delta x_n}{\Delta x_n - \Delta x_{n-1}}. \tag{7}$$

The secant method reduces to the Steffensen method by applying the acceleration only at every other step of the basic fixed point iteration.

The secant method converges with the order $\frac{1+\sqrt{5}}{2}$ (see for example [23]), but as it requires only one new basic fixed point iterate (or one new function evaluation per iteration), its efficiency index defined as $r^{1/s}$ with r the order of the method and s the number of function evaluations per iteration required by the method, is little better ($\frac{1+\sqrt{5}}{2} \simeq 1.618$) than the Steffensen's one ($\sqrt{2} \simeq 1.414$).

Another famous acceleration algorithm is the relaxation method (see [25] for example), sometimes called mixing algorithm [26,27]:

$$x_{n+1} = \omega g(x_n) + (1 - \omega)x_n. \tag{8}$$

The determination of the optimal relaxation parameter ω_{opt} is the main drawback of this algorithm because it depends on the derivative of g which is *a priori* unknown. Hence empiric static values are often set in practice (most often $\omega \equiv 0.5$, see for example [28]). Using the equivalent formulation

$$x_{n+1} = x_n + \omega(g(x_n) - x_n) \tag{9}$$

it is obvious that the secant and the Steffensen methods are particular relaxation methods with a varying relaxation parameter. Hence, in the literature the secant method is also called Aitken relaxation or dynamic relaxation [4,15,17].

Many other scalar acceleration techniques, also called extrapolation techniques, have been proposed since Aitken, but these methods often aim to be generalizations of the Aitken's process, see [6] for a detailed description of some of these algorithms. In particular, let us mention the ϵ-algorithm [29]

$$\begin{aligned} &\epsilon^n_{-1} = 0, \qquad \epsilon^n_0 = x_n, \qquad \epsilon^{n+1}_0 = g(x_n) \\ &\epsilon^n_{k+1} = \epsilon^{n+1}_{k-1} + \frac{1}{\epsilon^{n+1}_k - \epsilon^n_k}, \end{aligned} \tag{10}$$

whose ϵ_2 version, i.e. $x_{n+1} = \epsilon^n_2$, is equivalent to the Steffensen method. An interesting generalization of several acceleration methods is proposed by the E-algorithm [6].

2.2. Vector sequences

Practical problems deal most often with many unknowns and systems of nonlinear equations. So let us now consider the vector fixed point equation

$$X = G(X), \quad X \in \mathbb{R}^N, \qquad G : \mathbb{R}^N \to \mathbb{R}^N, \quad N \in \mathbb{N}^\star,\ N > 1, \tag{11}$$

leading to the basic vector fixed point sequence

$$X_{n+1} = G(X_n). \tag{12}$$

Vector sequence acceleration methods have been extensively studied in the literature [4,5,3,30,7–9,27]. Most of them are extensions of scalar acceleration methods with the following definition of the inverse of a real vector

$$X^{-1} = \frac{X}{\|X\|^2}. \tag{13}$$

The norm involved in the preceding definition is often chosen to be the Euclidean norm even if other choices can be made, see [4] for example. More generally, the inverse of a vector can be defined via the inner product (denoted here by $\cdot$) with another vector

$$X^{-1} = \frac{Y}{Y \cdot X}, \quad \forall Y \in \mathbb{R}^N. \tag{14}$$

As the Δ^2-Steffensen method can be expressed in different equivalent manners in the scalar case (see for example Eqs. (3)–(5)), many vector formulations based on the definition (13) and (14) are available. For example the method proposed by Irons and Tuck in [5] is obtained from Eq. (5) with the definition (13) of the inverse of a vector:

$$X_{n+1} = G(G(X_n)) - \frac{\Delta G(X_n) \cdot \Delta^2 X_n}{\|\Delta^2 X_n\|^2} \Delta G(X_n) \tag{15}$$

where as previously $\Delta X_n = G(X_n) - X_n$, $\Delta G(X_n) = G(G(X_n)) - G(X_n)$ and $\Delta^2 X_n = \Delta G(X_n) - \Delta X_n$.

Sometimes in the literature, the following expression is also attributed to Irons and Tuck (see for example [6,8]) but it seems in fact to be the approach of Lemaréchal [31]

$$X_{n+1} = X_n - \frac{\Delta X_n \cdot \Delta^2 X_n}{\|\Delta^2 X_n\|^2} \Delta X_n. \tag{16}$$

This method can be directly obtained from the combination of definition (13) and the Steffensen's scalar equation (3).

The following methods are based on the second definition of the inverse of a vector (14). Starting from the scalar equation (4), the approach of Graves-Morris [7] gives

$$X_{n+1} = G(X_n) - \frac{\|\Delta X_n\|^2}{\Delta X_n \cdot \Delta^2 X_n} \Delta G(X_n), \tag{17}$$

while the first step of the vector A-algorithm of Sedogbo [8] can be expressed as

$$X_{n+1} = G(X_n) - \frac{\|\Delta G(X_n)\|^2}{\Delta G(X_n) \cdot \Delta^2 X_n} \Delta X_n. \tag{18}$$

Starting from the scalar equation (5), the method of Jennings [32] writes

$$X_{n+1} = G(G(X_n)) - \frac{\Delta X_n \cdot \Delta G(X_n)}{\Delta X_n \cdot \Delta^2 X_n} \Delta G(X_n), \tag{19}$$

as Zienkiewicz and Lohner [33] proposed the following vector acceleration method

$$X_{n+1} = G(G(X_n)) - \frac{\|\Delta G(X_n)\|^2}{\Delta G(X_n) \cdot \Delta^2 X_n} \Delta G(X_n). \tag{20}$$

The vector ϵ-algorithm of Wynn [3] is also a direct extension of the scalar ϵ-algorithm (see [29] and Eq. (10)) with the definition (13) of the inverse of a vector. Hence the vector ϵ_2-algorithm writes

$$X_{n+1} = G(X_n) + \frac{\Delta G(X_n)\|\Delta X_n\|^2 - \Delta X_n \|\Delta G(X_n)\|^2}{\|\Delta^2 X_n\|^2}. \tag{21}$$

In his article, Macleod [34] compared 9 multi-dimensional Δ^2 methods that reduce in one dimension to the Steffensen method. Some of them are listed previously but others were also introduced. In particular, the application of the scalar Steffensen algorithm on each component of the vector was tested. However the results presented in [34], in accordance with [6], show that vector extrapolation algorithms (based on projection approaches) are more interesting than the scalar ones on each component. In these test cases, the Irons and Tuck [5] approach was the more efficient.

Many variants of the multi-dimensional secant method [4,15,27] can be formally obtained in the same manner from the scalar equations (6) and (7).

Other vector extensions of scalar extrapolation methods are detailed in [6,9]. These methods can be viewed as multi-step generalizations of iterative vector Δ^2 approaches, in the sense that they require more basic fixed point iterations.

The nonlinear hybrid approach proposed in [9] introduces the vector Δ^k method:

$$X_{n+1} = X_n - \frac{\Delta X_n \cdot \Delta^{k+1} X_n}{\|\Delta^{k+1} X_n\|^2} \Delta^k X_n \tag{22}$$

where

$$\begin{cases} \Delta^k X_n = \Delta^{k-1} G(X_n) - \Delta^{k-1}(X_n) \\ \Delta^k G^l(X_n) = \Delta^{k-1} G^{l+1}(X_n) - \Delta^{k-1} G^l(X_n) \\ \Delta^0 = Id \\ G^l(X_n) = \underbrace{G(G(\ldots G}_{l \text{ times}}(X_n))) \\ G^0 = Id. \end{cases}$$

For $k = 1$, the Lemaréchal's method is recovered, see Eq. (16). This generalization for $k > 1$ seems suitable to deal with bifurcation problems (see [35] for example).

Another interesting generalization of multi-dimensional secant approaches has been originally proposed by Anderson [4]. This method has been directly built for vector sequences and is based, like the nonlinear hybrid approach [9], on a minimization process. This approach is a one-step method as it is applied every fixed point iteration. As it has been pointed out in some recent articles [26,27,36], this method seems to be efficient in many situations. In the following sections, we shall see that the reduced rank extrapolation method of Eddy [19] can be viewed as a multi-step Anderson's method.

3. A generic residual-based acceleration approach

Let $(X_n)_n$ be a sequence of vectors of $\mathbb{R}^N$ converging to an unknown limit X, M an integer such that $M \leq N$ and $(Z_n^i)_n,\ i = 1, .., M$ arbitrary vector sequences of $\mathbb{R}^N$ converging to zero. We define the transformation Y that transforms the sequence $(X_n)_n$ into a new sequence $(Y_n)_n$

$$Y_n = X_n - \sum_{i=1}^{M} \lambda_n^i Z_n^i \tag{23}$$

with $\lambda_n^i,\ i = 1, .., M$, some scalars.

When $M = 1$, expression (23) reduces to the nonlinear hybrid procedure of Brezinski and Chehab [9]. Obviously if the λ_n^i are constant parameters independent of n: $\lambda_n^i \equiv \lambda^i,\ \forall n$, the sequence $(Y_n)_n$ converges to X and hence the transformation Y is regular. As in practice the λ_n^i vary with n, the regularity of Y cannot be ensured for all convergent sequences. This drawback is quite widespread in the existing extrapolation methods, it is the case for example of the Aitken's Δ^2 process (e.g. [6]).

However, under the usual assumption (see for example [10,6,37]) that consists in setting the same coefficients in the expressions of Y_n and Y_{n+1}, an estimation of X can be obtained by minimizing $\delta Y_n = Y_{n+1} - Y_n$ with respect to these coefficients. In our case, we can write

$$Y_{n+1} = X_{n+1} - \sum_{i=1}^{M} \lambda_n^i Z_{n+1}^i, \tag{24}$$

and hence

$$\delta Y_n = \delta X_n - \sum_{i=1}^{M} \lambda_n^i \delta Z_n^i \tag{25}$$

where the forward difference operator δ acts on the lower index: $\delta S_n = S_{n+1} - S_n$.

By construction, the optimal parameters $\lambda_n^i,\ i = 1, .., M$ lead to

$$\|\delta Y_n\| \leq \|\delta X_n\| \tag{26}$$

for the chosen norm. The vector sequence $(Y_n)_n$ therefore converges faster than $(X_n)_n$.

For the Euclidean norm, an expression of the λ_n^i can be obtained using the normal equation solution.[1] Defining the following $N \times M$ matrices

$$Z_n = (Z_n^1 \ldots Z_n^M), \qquad \delta Z_n = (\delta Z_n^1 \ldots \delta Z_n^M)$$

the least-square minimization gives

$$\lambda_n = \begin{pmatrix} \lambda_n^1 \\ \vdots \\ \lambda_n^M \end{pmatrix} = (\delta Z_n^T \delta Z_n)^{-1} \delta Z_n^T \delta X_n. \tag{27}$$

In practice, if a vector δZ_n^i is collinear to another vector $\delta Z_n^j,\ j \neq i$, the matrix $(\delta Z_n^T \delta Z_n)$ becomes singular and then its inversion is impossible. In this case, the corresponding λ_n^i is set to zero and the system is reduced.

The expression (27) of λ_n is then used to define two sequence transformations which will lead to extrapolation methods (see Section 4):

$$Y_n = X_n - Z_n(\delta Z_n^T \delta Z_n)^{-1} \delta Z_n^T \delta X_n \tag{28}$$

or

$$Y_{n+1} = X_{n+1} - Z_{n+1}(\delta Z_n^T \delta Z_n)^{-1} \delta Z_n^T \delta X_n. \tag{29}$$

The generic formulation (28) in the end is very close to the polynomial extrapolation method formalism as described in [37]. The reduced rank extrapolation method [19] is recovered for $Z_n^i = \delta X_{n+i-1}$ which reduces for $M = 1$ to the Lemaréchal's method [31] (see Eq. (16)).

The choice $Z_n^i = \delta X_{n-i}$ enables us to deal with a linear combination of the previous sequence residuals. In this case, expression (28) leads for $M = 1$ to a vector extension of the Steffensen's method of Eq. (4) whereas a generalization of the

[1] In practice, other minimization strategies are often used such as the Gram–Schmidt orthogonalization algorithm or the QR decomposition. This latter method seems in this case to be the most efficient [16,36].

Irons and Tuck method (see [5] or Eq. (15)) is obtained with expression (29). Moreover, the basic idea of Anderson [4] is also recovered with this approach. For this choice, the kernel of the transformation Y is the same as the Shanks [2] transformation:

$$\sum_{i=0}^{k} a_i(X_{n-i} - X) = 0 \tag{30}$$

with $\sum_{i=0}^{k} a_i \neq 0$ and $a_0.a_k \neq 0$.

Many other residuals Z_n^i could be chosen, for example $Z_n^i = \delta^k X_{n-i},\ k > 1$ as proposed in [9] for the Δ^k method (see Eq. (22)).

4. Two classes of iterative residual-based acceleration methods

This section is devoted to the application of the residual-based transformations described in Section 3 to accelerate the fixed point iteration method (see Eq. (12)). As mentioned in the introduction of Section 2, we are only interested in iterative acceleration methods. Moreover, we want to obtain one-step iterative methods (i.e. only one new fixed point evaluation between each new accelerate iterate) as opposed to multi-step iterative methods such as the Irons and Tuck method [5] (see Eq. (15)), the vector ϵ-algorithm [3] (see Eq. (21)) or the polynomial extrapolation algorithms performed in [37]. Considering that the iterative acceleration process applied to the fixed point iteration generates two sequences $(X_n)_n$ and $(G(X_n))_n$ (see [4] for example), two main formalisms of iterative methods are available. They differ in the way the two coupled sequences are taken into account in the definition of Y_n and Y_{n+1} in Eqs. (23)–(24). To illustrate our point, for each formalism we shall decline the iterative method obtained for the most common choice $Z_n^i = \delta X_{n-i}$ in the transformation (29) of Section 3. Methods derived from other choices of Z_n^i can be easily deduced from this example.

As it is not guaranteed that the transformations Y defined in Section 3 are regular with respect to the fixed point limit X, the convergence criterion must still focus on the standard fixed point iteration residual $\Delta X_n = G(X_n) - X_n$ to obtain the desired solution.

4.1. First class: crossed sequences method

The first formalism focuses on the basic fixed point sequence $(G(X_n))_n$

$$Y_n = G(X_{n-1}) - \sum_{i=1}^{M} \lambda_n^i Z_n^i \tag{31}$$

$$Y_{n+1} = G(X_n) - \sum_{i=1}^{M} \lambda_n^i Z_{n+1}^i, \tag{32}$$

and takes into account the accelerated sequence $(X_n)_n$ in the definition of the vanishing sequences $(Z_n^i)_n,\ i = 1, ..M$. Indeed, Z_n^i depends in this case on the fixed point iteration residual

$$\Delta X_n = G(X_n) - X_n. \tag{33}$$

For example, the case $Z_n^i = \delta X_{n-i}$ in the generic approach of Section 3 becomes for the iterative crossed sequences method $Z_n^i = \Delta X_{n-i}$ as $Z_n^i = \delta^2 X_{n-i-1}$ becomes $Z_n^i = \Delta(X_{n-i}) - \Delta(X_{n-i-1}) = G(X_{n-i}) - X_{n-i} - G(X_{n-i-1}) + X_{n-i-1}$.

Example. For the case $Z_n^i = \Delta X_{n-i}$, the crossed sequences method gives

$$\delta Y_n = (G(X_n) - G(X_{n-1})) - \sum_{i=1}^{M} \lambda_n^i (\Delta X_{n-i+1} - \Delta X_{n-i}). \tag{34}$$

Using the notation

$$\begin{aligned} R_n &= (\Delta X_{n-1} \cdots \Delta X_{n-M}) \\ \delta R_n &= ((\Delta X_n - \Delta X_{n-1}) \cdots (\Delta X_{n+1-M} - \Delta X_{n-M})), \end{aligned}$$

we obtain

$$\begin{aligned} X_{n+1} &= Y_{n+1} \\ &= G(X_n) - R_{n+1}(\delta R_n^T \delta R_n)^{-1} \delta R_n^T (G(X_n) - G(X_{n-1})). \end{aligned} \tag{35}$$

For $M = 1$, the standard vector secant acceleration or dynamic relaxation method is recovered (e.g. [4,15,17]). This method corresponds formally to the vector extension of the scalar secant method (7) with the definition (13) of the inverse of the vector:

$$X_{n+1} = G(X_n) - \frac{(G(X_n) - G(X_{n-1})) \cdot (\Delta X_n - \Delta X_{n-1})}{\|\Delta X_n - \Delta X_{n-1}\|^2} \Delta X_n. \tag{36}$$

In a manner similar to the scalar case, if this acceleration method is applied alternately with a basic fixed point iteration step, the Irons and Tuck method (15) is recovered. △

4.2. Second class: alternate sequences method

In this case, Y_n is concerned by the sequence $(X_n)_n$ as Y_{n+1} is concerned by the sequence $(G(X_n))_n$:

$$Y_n = X_n - \sum_{i=1}^{M} \lambda_n^i Z_n^i \tag{37}$$

$$Y_{n+1} = G(X_n) - \sum_{i=1}^{M} \lambda_n^i Z_{n+1}^i, \tag{38}$$

where Z_n^i is a function of $X_{n-i},\ i = 0, ..M$ whereas Z_{n+1}^i depends only on $G(X_{n-i}),\ i = 0, ..M$.

We may have for example

$$\begin{cases} Z_n^i = \delta X_{n-i} = X_{n-i+1} - X_{n-i} \\ Z_{n+1}^i = \delta G(X_{n-i}) = G(X_{n-i+1}) - G(X_{n-i}) \end{cases}$$

or

$$\begin{cases} Z_n^i = \delta^2 X_{n-i-1} = X_{n-i+1} - 2X_{n-i} + X_{n-i-1} \\ Z_{n+1}^i = \delta^2 G(X_{n-i-1}) = G(X_{n-i+1}) - 2G(X_{n-i}) + G(X_{n-i-1}). \end{cases}$$

For any choice of $Z_n^i = \delta^k X_{n-i-k+1},\ k > 1$, the expression of $\delta Y_n = Y_{n+1} - Y_n$ is then a linear combination of fixed point residuals $\Delta X_n = G(X_n) - X_n$, the coefficients of which are identical to the sequence coefficients in the accelerated sequences Eqs. (37) or (38). In this case, if the acceleration process is chosen to be a two-step method (the acceleration is then applied every other step of the basic fixed point iteration), the equation δY_n to minimize becomes the same in both formalisms. However, except for the case $k = 1$ and $M = 1$, the expression of X_{n+1} remains different between each formalism, and so the obtained acceleration methods differ.

Remark. When a nonlinear equation $F(X) = 0$ is under consideration, the residual $F(X_n)$ is often used instead of $G(X_n) - X_n$, see for example [26,27,38].

Example. For the case $Z_n^i = \delta X_{n-i}/Z_{n+1}^i = \delta G(X_{n-i})$, the alternate sequences method reads

$$\begin{aligned} \delta Y_n &= (G(X_n) - X_n) - \sum_{i=1}^{M} \lambda_n^i \delta(G(X_{n-i}) - X_{n-i}) \\ &= \Delta X_n - \sum_{i=1}^{M} \lambda_n^i \delta \Delta X_{n-i} \\ &= \Delta X_n - \sum_{i=1}^{M} \lambda_n^i (\Delta X_{n-i+1} - \Delta X_{n-i}). \end{aligned} \tag{39}$$

With the same definition of δR_n as in the crossed sequence method and introducing

$$\delta G_n = (\delta G(X_{n-i}) \cdots \delta G(X_{n-M})),$$

the accelerate iterate writes:

$$\begin{aligned} X_{n+1} &= Y_{n+1} \\ &= G(X_n) - \delta G_n (\delta R_n^T \delta R_n)^{-1} \delta R_n^T \Delta X_n. \end{aligned} \tag{40}$$

This extrapolation method was first introduced by Anderson in the 1960's [4] and has recently attracted more attention [26,27,36].[2] Moreover, this approach is similar to the so-called "interface quasi-Newton method" [16] or "reduced order models" method used in [39].

[2] In the original formulation, Anderson proposed to define X_{n+1} as a relaxation between Y_{n+1} and Y_n. However, in practice the relaxation parameter was set to 1 and hence $X_{n+1} \equiv Y_{n+1}$.

For $M = 1$, this method reduces to another vector secant method

$$X_{n+1} = G(X_n) - \frac{(\Delta X_n - \Delta X_{n-1}) \cdot \Delta X_n}{\|\Delta X_n - \Delta X_{n-1}\|^2}(G(X_n) - G(X_{n-1})). \tag{41}$$

Both vector secant methods of Eqs. (36) and (41) can be obtained directly from Eq. (7) with the definition of the inverse of a vector (13) but they differ from the numerator's term involved in the inner product. Moreover, these two secant methods reduce to the Irons and Tuck method if the acceleration method is applied at every other step. △

5. Application

5.1. MTest open-source tool

To illustrate the performances of some of the residual-based acceleration algorithms presented in this paper (see Section 4), we introduced them in an open-source tool named MTest which is developed in the PLEIADES platform [20] as part of the MFront project [21]. MFront is a code generator that enables engineers and researchers in the field of structural mechanics to implement various types of mechanical behaviour easily, reliably and efficiently.

MTest has been designed to test mechanical behaviours on a single material point by imposing constraints on each component of the strains $\underset{\sim}{\epsilon}$ or the stresses $\underset{\sim}{\sigma}$. Since $\underset{\sim}{\epsilon}$ and $\underset{\sim}{\sigma}$ are rank-2 symmetric tensors, they will be described by two vectors E and Σ. In 3D the number of components of E and Σ is equal to $N = 6$.

For the sake of clarity, in the sequel we shall only describe the case in which all the stress components are imposed. Imposing strain components is accomplished by introducing Lagrange multipliers which increase the problem size.

Equilibrium equation. At each time step between t and $t + \Delta t$, a strain increment $X = E|_{t+\Delta t} - E|_t$ which respects the mechanical equilibrium is sought. This strain increment thus satisfies the following nonlinear equation:

$$R(X) = \Sigma|_{t+\Delta t}(X, \Delta t, V|_t) - \Sigma|_{t+\Delta t}^{\mathrm{imp}} = 0, \tag{42}$$

where $\Sigma|_{t+\Delta t}$ is a generally nonlinear function computing the stresses as a result of the mechanical behaviour integration, V is a set of internal state variables known at the beginning of the time step and whose value at the end of the time step is also a result of the mechanical behaviour integration, and $\Sigma|_{t+\Delta t}^{\mathrm{imp}}$ are the imposed stress values. In this paper, the mechanical behaviour integration will be considered as a black box.

On the use of the Newton–Raphson algorithm. Problem (42) can be solved by the Newton–Raphson algorithm if the behaviour provides the so-called consistent tangent operator $\frac{\partial \Sigma|_{t+\Delta t}}{\partial X}$ [40]. This strategy is considered to be the default in most implicit finite element solvers dedicated to structural mechanics [41–44]. The Cast3M finite element solver is a noticeable exception and its default algorithm paved the way to the method presented in this section [45,38].

Providing consistent tangent operator is only feasible analytically in certain cases when the behaviour integration is done using an implicit scheme [46]. Even in this case, its computation may require a tremendous amount of work.

It is also worth noting that quadratic convergence of the Newton–Raphson algorithm strongly depends on the quality of this operator and can easily be lost due to implementation mistakes or, even worse, to a poor convergence criteria for the behaviour integration. This point is specifically outlined in the Abaqus user guide [41].

In structural mechanics, the use of the consistent tangent operator may have other drawbacks: this matrix may be unsymmetric and not definite positive (due to softening for example), which forbids the use of standard efficient linear solvers.

Alternative algorithm. It is therefore worth considering algorithms not relying on the consistent tangent operator to solve Eq. (42). One interesting characteristic of mechanical behaviour is that it usually introduces an elastic operator K which allows us to rewrite Eq. (42) as a fixed point problem using a quasi-Newton method:

$$X = G(X) \quad \text{with } G(X) = X - K^{-1} R(X). \tag{43}$$

In practice, for the great majority of mechanical behaviours, the elastic operator K is "stiff enough" to guarantee that the spectral radius of the Jacobian matrix of G is less than 1 and hence that the fixed point iterations method will converge.

Solving Eq. (43) with a standard fixed point iterations method thus has the following advantages:

- K is easy to compute.
- K is symmetrical and definite positive.
- K can be factorized only once through the whole computation.
  In practice, if finite strains are considered or if the elastic properties evolve with external parameters, such as temperature, it may be worth updating this matrix from time to time.

However, as mentioned in the introduction of this paper, such a fixed point iterations method leads to a very slow linear convergence. Therefore, use of acceleration procedures is mandatory to obtain an efficient method.

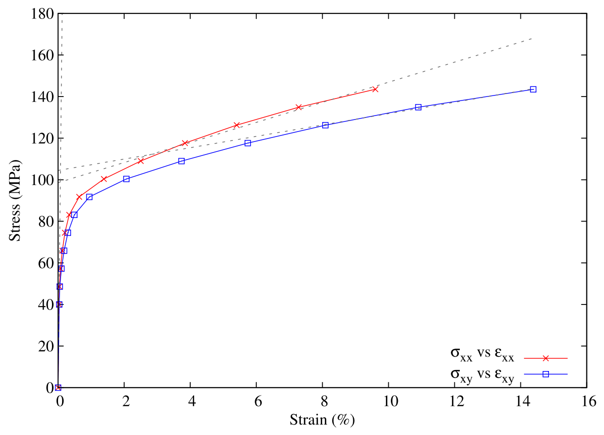


Fig. 1. Chaboche test case.

Stopping criterion and accuracy. Whatever the algorithm used, iterations are stopped once the absolute maximum norm of the residual $\Delta X_n = G(X_n) - X_n$ is lower than a prescribed accuracy ε. This stopping criterion guarantees that the obtained solution is the fixed point equation solution.

MTest can be used to simulate simple mechanical tests. In this case, ε is usually chosen equal to 10^{-6} or 10^{-8} to obtain satisfactory results. This corresponds to a relative precision lower than 10^{-3} or 10^{-5} as the strains typically have a magnitude ranging between 10^{-3} and 10^{-1}.

A practical way of checking that the behaviour integration gives repeatable results whatever compilers, compiler optimization flags and/or operating systems is to use a ε value smaller than the criterion values used within the behaviour integration. Typically, a stringent value of 10^{-12}, corresponding to a relative precision of about 10^{-9}, is used for ε (precision by default) while the behaviour integration is performed with a criterion of 10^{-8}.

5.2. *Test cases*

In order to illustrate the performances of the various algorithms described in this paper, we have chosen two test cases, called Chaboche and Polycrystals, respectively and which shall now be briefly described.

5.2.1. *Chaboche test case*

This test case describes an isotropic standard plastic behaviour with two kinematic hardening rules derived from the works of J.L. Chaboche [47,48]. Such a behaviour is commonly used to describe metal plasticity in engineering. A whole description of the behaviour can be found in the Code-Aster documentation [49].

The behaviour is time-independent, so the time scale is arbitrary. The test case imposes two components of the stresses: σ_{xx} and σ_{xy} which monotonically and linearly increase from 0 at the beginning of the test to 143.5 MPa at the end of the test. The mechanical equilibrium imposes that the other components of the stresses are null. This loading is described in 13 time steps. The results of the simulation are reported in Fig. 1.

The results exhibit two linear stages. At the very beginning, the behaviour is elastic. The initial slope has a magnitude close to the Young modulus, which is equal to 145.2 GPa (quasi vertical line on Fig. 1). At the end of the test, the stress is also almost a linear function of the strain: the effective slope is around 300 smaller than the initial one. This ratio gives an estimation of how close the elastic operator is from the consistent tangent operator. Performances of the standard fixed point iterations for solving the quasi-Newton problem (43) are expected to decrease as this ratio increases.

Furthermore, the behaviour is integrated in this case with an implicit scheme that supplies the consistent tangent operator: we can thus compare the accelerated fixed point algorithms proposed in this paper to the standard Newton–Raphson algorithm.

5.2.2. *Polycrystals test case*

The behaviour used in this second test case is the result of the Berveiller–Zaoui homogenization scheme to a FCC-polycrystal made of 30 grains [50]. The resulting behaviour is orthotropic. FCC crystals have 12 sliding systems, the plasticity in each grain is described by flow rules, no by sliding systems, derived from dislocation dynamics (see the Code-Aster documentation for details [51]). This behaviour introduces 1272 state variables, which is very large and unusual.

The material is submitted to a uniaxial tensile test: the axial strain ε_{zz} monotonically and linearly increases from 0 at the beginning of the test to 5% at the end of the test. The loading is described in 15 time steps. All stress components are null, except the axial stress σ_{zz}. The result of this test case is reported in Fig. 2.

As for the Chaboche test case, Fig. 2 shows two linear stages. The slope ratio between those stages in this case is close to 70. The standard fixed point iterations method is hence expected to converge faster than in the Chaboche test case.

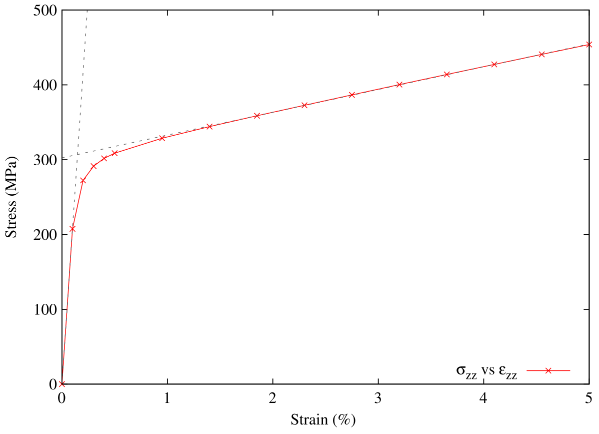


Fig. 2. Polycrystals_dd_cfc test case.

Implicit schemes are not considered for the integration of this behaviour for two main reasons:

1. the Jacobian matrix of the implicit system would be fairly complex to compute;
2. such an implicit scheme would require to solving a 1272×1272 dense system at each Newton–Raphson iteration.

The integration is thus performed using a standard Runge–Kutta algorithm. As a consequence, the consistent tangent operator is not available, so the equilibrium equation (42) cannot be solved using a Newton–Raphson algorithm.

The solver must then rely on standard fixed point iterations based on Eq. (43). Acceleration methods for fixed point iterations are thus extremely appealing for this test case.

5.3. Results

5.3.1. Acceleration methods compared

In this section we shall compare different residual-based acceleration algorithms generated thanks to the two formalisms detailed in Section 4. As the dimension of the involved problem is $N = 6$ (see Section 5.1), we will compare acceleration methods involving only two or three iterations.

Concerning the methods involving the last two iterations, we study the performances of the residual methods with $M = 1$ and $Z_n^1 = \delta X_{n-1}$:

- the crossed secant method or crossed 1-δ method (often called Aitken relaxation or dynamic relaxation in the literature [4,15])

$$X_{n+1} = G(X_n) - \frac{(G(X_n) - G(X_{n-1})) \cdot (\Delta X_n - \Delta X_{n-1})}{\|\Delta X_n - \Delta X_{n-1}\|^2} \Delta X_n; \tag{44}$$

- the alternate secant method or alternate 1-δ method (also called Anderson extrapolation method with $M = 1$ [4])

$$X_{n+1} = G(X_n) - \frac{(\Delta X_n - \Delta X_{n-1}) \cdot \Delta X_n}{\|\Delta X_n - \Delta X_{n-1}\|^2} (G(X_n) - G(X_{n-1})); \tag{45}$$

- the well-known Irons and Tuck method [5] in which both preceding methods reduce if applied alternately with a standard fixed point iteration

$$\begin{aligned} X_n &= G(X_{n-1}) \\ X_{n+1} &= G(X_n) - \frac{\Delta X_n \cdot \Delta^2 X_{n-1}}{\|\Delta^2 X_{n-1}\|^2} \Delta X_n. \end{aligned} \tag{46}$$

The residual methods with $M = 1$ and $Z_n^1 = \delta^2 X_{n-i-1}$ as well as the residual methods with $M = 2$ and $Z_n^i = \delta X_{n-i}, \; i = 1, 2$ are based on the last three iterations and will also be compared:

- the crossed δ^2 method

$$X_{n+1} = G(X_n) - \frac{(G(X_n) - G(X_{n-1})) \cdot (\Delta X_n - 2\Delta X_{n-1} + \Delta X_{n-2})}{\|\Delta X_n - 2\Delta X_{n-1} + \Delta X_{n-2}\|^2} (\Delta X_n - \Delta X_{n-1}); \tag{47}$$

- the alternate δ^2 method

$$X_{n+1} = G(X_n) - \frac{(\Delta X_n - 2\Delta X_{n-1} + \Delta X_{n-2}) \cdot \Delta X_n}{\|\Delta X_n - 2\Delta X_{n-1} + \Delta X_{n-2}\|^2} (G(X_n) - 2G(X_{n-1}) + G(X_{n-2})); \tag{48}$$

- the crossed 2-δ method

$$\begin{aligned}&X_{n+1} = G(X_n) - \lambda_n^1 \Delta X_n - \lambda_n^2 \Delta X_{n-1},\\ &\text{with } \lambda_n^1, \lambda_n^2 \text{ minimizing}\\ &\delta Y_n = (G(X_n) - G(X_{n-1})) - \lambda_n^1(\Delta X_n - \Delta X_{n-1}) - \lambda_n^2(\Delta X_{n-1} - \Delta X_{n-2});\end{aligned} \tag{49}$$

- the alternate 2-δ method (equivalent to the Anderson extrapolation method with $M = 2$ [4])

$$\begin{aligned}&X_{n+1} = G(X_n) - \lambda_n^1(G(X_n) - G(X_{n-1})) - \lambda_n^2(G(X_{n-1}) - G(X_{n-2})),\\ &\text{with } \lambda_n^1, \lambda_n^2 \text{ minimizing}\\ &\delta Y_n = \Delta X_n - \lambda_n^1(\Delta X_n - \Delta X_{n-1}) - \lambda_n^2(\Delta X_{n-1} - \Delta X_{n-2}).\end{aligned} \tag{50}$$

It can be easily shown that the alternate 2-δ method remains the same if we consider $G(X_n) - G(X_{n-2})$ instead of $G(X_{n-1}) - G(X_{n-2})$ in the expression of X_{n+1}. The new coefficients $\tilde{\lambda}_n^1$ and $\tilde{\lambda}_n^2$ are hence linear combinations of λ_n^1 and λ_n^2:

$$\begin{aligned}\tilde{\lambda}_n^1 &= \lambda_n^1 - \lambda_n^2\\ \tilde{\lambda}_n^2 &= \lambda_n^2.\end{aligned}$$

However, this is no longer true for the crossed 2-δ method. So we can also consider the following variant of the crossed 2-δ method

- crossed 2-δ bis method

$$\begin{aligned}&X_{n+1} = G(X_n) - \lambda_n^1(G(X_n) - X_n) - \lambda_n^2(G(X_n) - X_{n-1}),\\ &\text{with } \lambda_n^1, \lambda_n^2 \text{ minimizing}\\ &\delta Y_n = (G(X_n) - G(X_{n-1})) - \lambda_n^1(\Delta X_n - \Delta X_{n-1}) - \lambda_n^2(G(X_n) - G(X_{n-1}) - X_{n-1} + X_{n-2}).\end{aligned} \tag{51}$$

In the following sections, we shall compare the acceleration performances of each residual methods in terms of number of iterations required for the whole loading (total number of iterations) or at each time step to reach the convergence criterion. We designate by number of iterations the number of evaluations of the fixed point function G. As the behaviour integration (used to evaluate R, see Section 5.1) is the most costly part of the computation per iteration, this measure is relevant. Other situations are discussed in Section 5.3.4.

5.3.2. Two iteration residual methods comparison

Chaboche test case. For the Chaboche test case, Table 1 presents the total number of iterations (sum of all the time steps) required by the two iteration residual methods introduced in Section 5.3.1 as well as by the standard fixed point iterations method. As the consistent tangent operator is available for this test case, we also report in the table the total number of iterations required by the standard second-order Newton method. Moreover, the numerical order of convergence obtained for each method is also given.

The first conclusion to be drawn from Table 1 is that all the listed acceleration methods converge to the fixed point solution in this case. We can also note the very poor convergence of the fixed point iterations method. The residual-based methods greatly accelerate the fixed point convergence in the sense that the number of iterations drastically decreases between the standard fixed point iteration method and the residual-based acceleration methods. For this test case, the alternate secant method seems the most powerful two iteration residual method to accelerate the fixed point convergence whatever the precision required. Moreover, the numbers of iterations involved by the alternate secant method are quite close to those of the Newton's method, which confirms the efficiency and the performance of this method. Numerically, the convergence order of both secant methods is around 1.6 which is in good agreement with the theoretical order of convergence of the scalar secant approach (see Section 2.1). However the rate of convergence in this case is in favour of the alternate secant method. For the Irons and Tuck approach, no numerical order of convergence appears clearly during the simulations.

Table 1
Total number of iterations and numerical convergence order for the Chaboche test case—two iteration residual methods.

	Crossed secant	Alternate secant	Irons and Tuck	Newton	Fixed point
$\varepsilon = 10^{-6}$	84	70	93	62	9 565
$\varepsilon = 10^{-8}$	123	85	200	64	19 725
$\varepsilon = 10^{-12}$	281	102	365	78	39 363
Cvg order	1.6	1.6	–	2	1

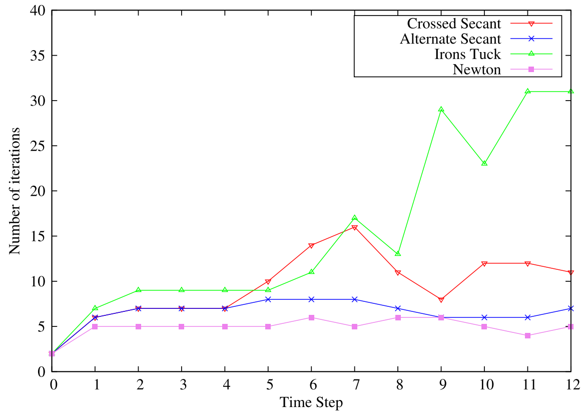


Fig. 3. Number of iterations per time step—$\varepsilon = 10^{-8}$—Chaboche test case—two iteration residual methods.

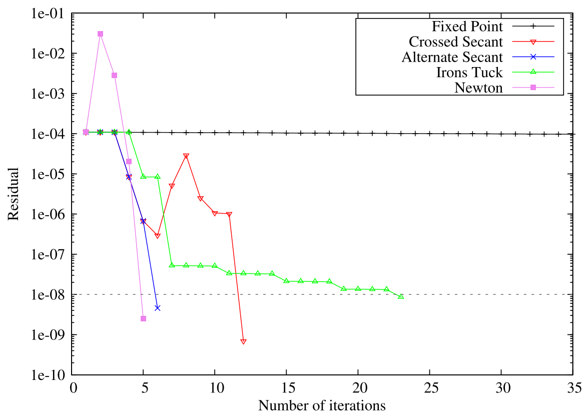


Fig. 4. Residuals for time step #10—$\varepsilon = 10^{-8}$—Chaboche test case—two iteration residual methods.

Fig. 3 presents a detailed view of the number of iterations required per acceleration method and per time step for a desired precision of $\varepsilon = 10^{-8}$.

This figure confirms that whatever the time step and hence the loading, the alternate secant method is the more efficient two iteration residual-based method. The behaviour of this acceleration method is furthermore very stable and similar to the Newton's one. From this figure, we can also remark that the Irons and Tuck method has some convergence difficulties in the final linear stage. If we take a deep look at time step #10 for example, we can see in Fig. 4 that when the Irons and Tuck method is applied, the residual decreases slowly after a precision of 10^{-7} (but still faster than for the fixed point iteration method). This figure also shows the quite chaotic residual convergence of the crossed secant method and the Newton's method. For the Newton's method, we get back to the situation where the initial solution is far from the converged solution and hence the convergence of the method is not monotone. In this case, the Newton's method may diverge and is then often combined with line-search methods for example.

Polycrystals test case. Table 2 recapitulates the total number of iterations required by every two iteration residual-based method for the Polycrystals test case. The total number of iterations of the standard fixed point iteration method is also reported. This test case is not suitable to the evaluation of the numerical order of convergence: in fact no method presents a regular order of convergence, for the three approaches, it ranges from 0.5 to 3 according to the time step.

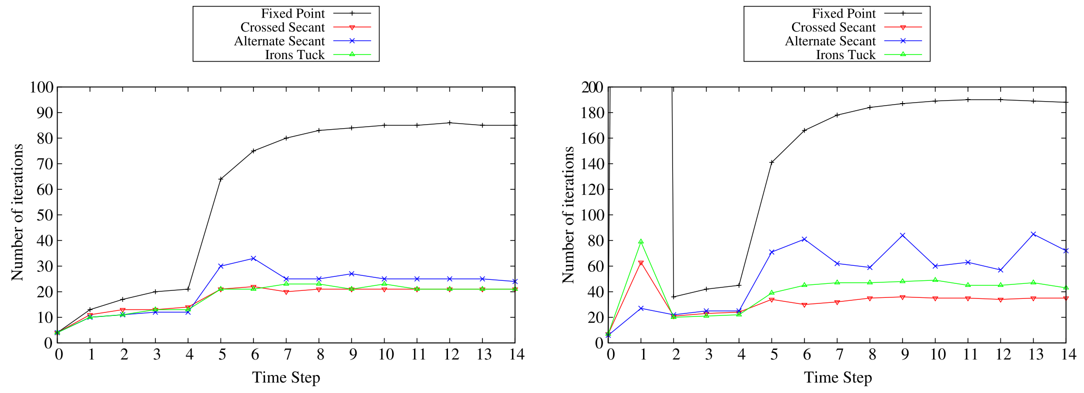
As for the Chaboche test case, all the listed acceleration methods converge to the problem solution whatever the precision required. Moreover, the standard fixed point iteration method converges definitely faster than in the Chaboche case. This is in good agreement with the elastic operator properties in this case, see Section 5.2.2. However, the proposed residual-based acceleration methods still accelerate the fixed point iterations convergence.

For this test case, the best acceleration method depends on the required precision. On the whole, the crossed secant method seems to be the most efficient, especially for the most stringent precision. The Irons and Tuck approach is also interesting in this case. On Fig. 5, the number of iterations required by each acceleration method is represented versus the time step for $\varepsilon = 10^{-8}$ (Fig. 5(a)) and $\varepsilon = 10^{-12}$ (Fig. 5(b)).

As expected by the results reported in Table 2, the number of iterations per time step obtained for $\varepsilon = 10^{-8}$ by the crossed secant method and the Irons and Tuck method are very close. The number of iterations required by the alternate secant method is always greater than or equal to those of these two methods whatever the time step. An example of residuals decrease is given in Fig. 6(a).

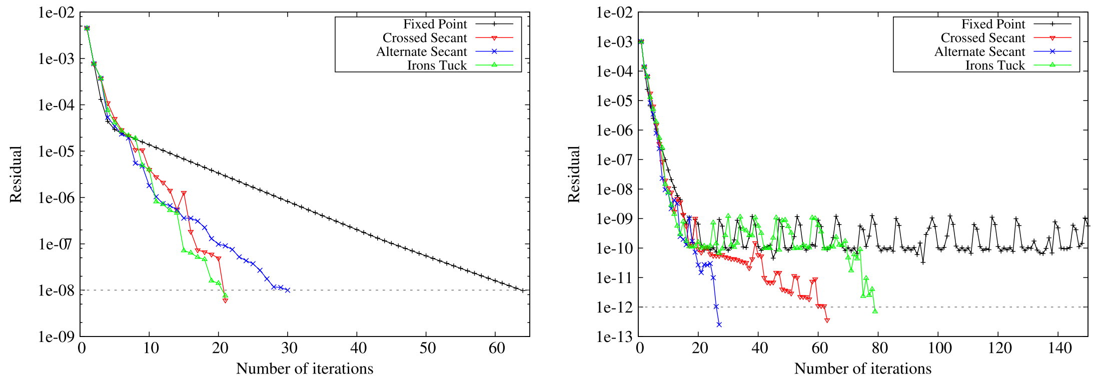
Table 2
Total number of iterations for the polycrystals test case—two iteration residual methods.

	Crossed secant	Alternate secant	Irons and Tuck	Fixed point
$\varepsilon = 10^{-6}$	173	142	145	376
$\varepsilon = 10^{-8}$	265	313	267	887
$\varepsilon = 10^{-12}$	479	799	607	5302



(a) $\varepsilon = 10^{-8}$. (b) $\varepsilon = 10^{-12}$.

Fig. 5. Number of iterations per time step – polycrystals test case – two iteration residual methods.



(a) $\varepsilon = 10^{-8}$—time step #5. (b) $\varepsilon = 10^{-12}$—time step #1.

Fig. 6. Residuals for the polycrystals test case—two iteration residual methods.

Behaviours are quite similar for $\varepsilon = 10^{-12}$, except on time step #1 where the fixed point iterations method has great difficulties reaching the convergence: 3371 iterations are required! This time step corresponds to the slope break (elastic/plastic) in the material response, see Fig. 2. In order to obtain a truly precise solution in this kind of situation, acceleration methods are necessary. In this case, for a very slow convergence of the fixed point iterations method, the alternate secant is the most efficient (as for the Chaboche test case), as confirmed by Fig. 6(b).

Conclusion. These two test cases enable us to conclude that both vector secant methods are efficient in accelerating the fixed point iterations method convergence. According to the Chaboche test case, they also exhibit the same order of convergence of 1.6, which is in good agreement with the theoretical scalar secant convergence order. Finally, the best acceleration factor (or rate of convergence) between both approaches seems to depend on the problem under consideration. It appears that the more slowly the fixed point iteration method converges, the more the alternate secant method is efficient compared to the crossed secant method.

5.3.3. Three iteration residual methods comparison

In this section, we present the performances obtained for the two test cases with the five 'three iteration residual methods' listed in Section 5.3.1, Eqs. (47)–(51). For each method and for each time step, at iteration #2, the corresponding secant method is first applied.

Chaboche test case. The total numbers of iterations required for the Chaboche test case by the three iteration methods studied in this paper are reported in Table 3. The numerical orders of convergence are also reported in this table.

Table 3
Total number of iterations and numerical convergence order for the Chaboche test case—three iteration residual methods.

	Crossed δ^2	Alternate δ^2	Crossed 2-δ	Alternate 2-δ	Crossed 2-δ bis
$\varepsilon = 10^{-6}$	9 509	83	80	70	71
$\varepsilon = 10^{-8}$	19 098	102	145	85	Not Cvg
$\varepsilon = 10^{-12}$	37 357	122	687	102	Not Cvg
Cvg order	1	–	1.6	1.6	–

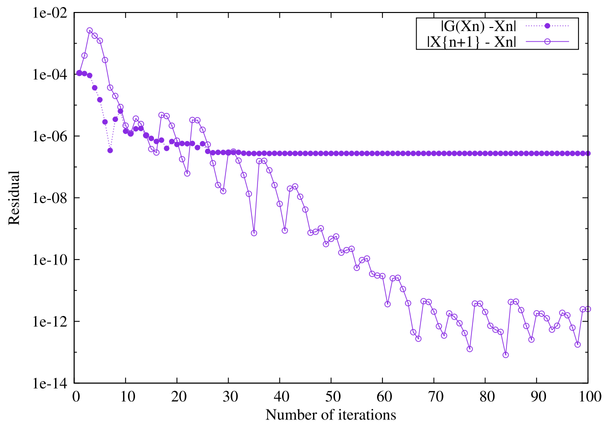


Fig. 7. Residuals of the crossed 2-δ bis method for $\varepsilon = 10^{-8}$ for time step #6—Chaboche test case.

The first conclusion to be drawn from this table is that all acceleration approaches except the crossed 2-δ bis method, converge to the fixed point solution. However, as shown in Fig. 7, the accelerated sequence generated by the crossed 2-δ bis method converges, but not towards the limit of the fixed point iteration method. This example clearly demonstrates the fact that in order to guarantee that the solution obtained thanks to an acceleration algorithm is truly the desired solution of the fixed point problem, the convergence criterion must remain based on the fixed point residual $\Delta X_n = G(X_n) - X_n$ whatever the acceleration algorithm performed.

The second conclusion is that the crossed δ^2 method is not a good acceleration method, since the number of iterations and the order of convergence obtained with this method are similar to those of the fixed point iterations method, see Table 1. The other three iteration methods (alternate δ^2 method, crossed 2-δ and alternate 2-δ) are interesting, the alternate methods being the most efficient, especially more powerful than the crossed secant method. No regular order of convergence had been numerically obtained for the alternate δ^2 method. It varies from 0.5 to 2.5 according to the time step. It is worth noting that the 2-δ methods recover the order of convergence of the corresponding secant (or 1-δ) methods. Moreover, in this case the alternate 2-δ method also has the same number of iterations as the alternate secant method (see Table 1) even if the different residual vectors involved in the minimization of Eq. (50) are not always collinear. For this test case, these two latter methods (which reduce to the Anderson's method with $M = 1$ and $M = 2$) are the most efficient among all the proposed methods.

Polycrystals test case. For the polycrystals test case, the five 'three iteration residual methods' have also been performed. The total number of iterations required by each method is reported in Table 4. Again, no numerical order of convergence can be clearly drawn from this test case.

Table 4
Total number of iterations involved for the Polycrystals test case—three iteration residual methods.

	Crossed δ^2	Alternate δ^2	Crossed 2-δ	Alternate 2-δ	Crossed 2-δ bis
$\varepsilon = 10^{-6}$	431	287	326	129	Not Cvg
$\varepsilon = 10^{-8}$	940	678	579	204	Not Cvg
$\varepsilon = 10^{-12}$	Not Cvg	1691	1613	476	Not Cvg

As for the Chaboche test case, the crossed 2-δ bis method does not converge to the fixed point solution, even if the accelerated sequence converges. For this test case, the crossed δ^2 approach is worse in term of number of iterations than the standard fixed point iteration method (see Table 2). Moreover for $\varepsilon = 10^{-12}$, this method has not yet reached the convergence criterion at time step #1 after 10,000 iterations!

As opposed to the Chaboche test case, here the crossed 2-δ method is on the whole little more efficient than the alternate δ^2 method but less interesting than the crossed secant method (see Table 2). Here again, it is the alternate 2-δ method that proves to be the most reliable acceleration method involving the last three iterations. This method is in particular more

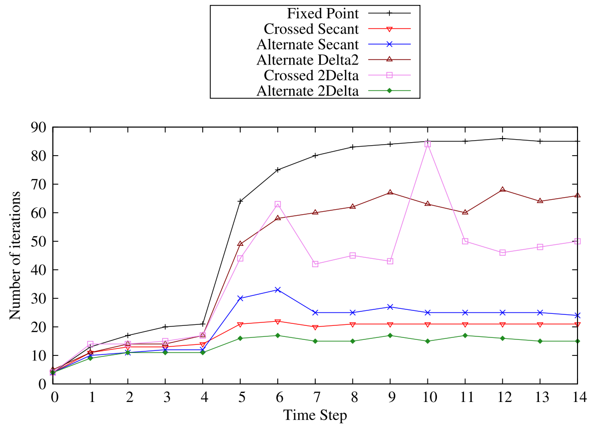


Fig. 8. Number of iterations per time step for $\varepsilon = 10^{-8}$ – polycrystals test case – three iteration residual methods.

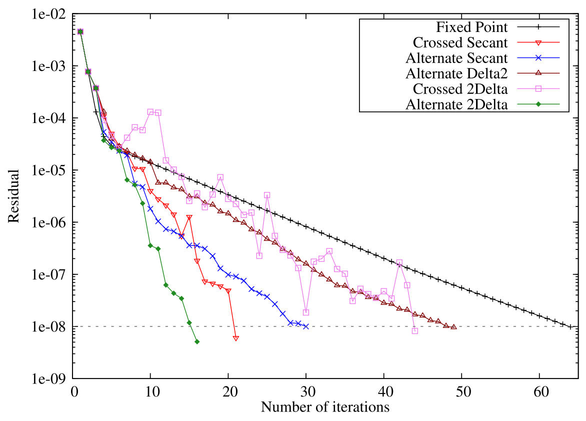


Fig. 9. Residuals for the time step #5 and $\varepsilon = 10^{-8}$ – polycrystals test case – three iteration residual methods.

efficient than the crossed secant, which was the most powerful two iteration method for this test case. These conclusions are clearly summarized by Fig. 8.

Fig. 9 confirms that crossed extrapolation methods have a more chaotic convergence behaviour per time step than the alternate methods and the fixed point iteration method. Moreover, the efficiency of the alternate 2-δ method as well as the convergence improvement brought by taking into account a third iteration in the alternate formalism also clearly appear in this figure.

Conclusion. For the test cases under consideration here, the acceleration methods based on the δ^2 residual do not seem relevant whatever the formalism.

Secondly, for both test cases, the alternate 2-δ, already known as the Anderson extrapolation method with $M = 2$ in literature, is the best outlined iterative acceleration method. It reduces at least to the most efficient vector secant method and even may improve it. This last conclusion is in good agreement with the experiments of Anderson [4] in which the alternate δ methods were successfully applied on a special class of nonlinear equations.

5.3.4. Discussion

In this paragraph we shall discuss some points related to the application of the proposed acceleration methods to problems with a large number N of degrees of freedom.

It is worth noting that the cost specifically attached to the acceleration methods proposed is generally negligible in comparison to the cost of the evaluation of the fixed point function. For example, in the case of the crossed secant method (44), we must compute two scalar products and update a vector. The cost of those operations grows linearly with N and can be easily parallelized by highly scalable implementations (such as BLAS routines, for example). This remark still holds true in distributed memory.

Moreover, the conclusion that the Newton–Raphson method, when available, appears as the best nonlinear resolution method is difficult to generalize as N grows. Indeed, in this case, due to the cost of Jacobian matrix factorization (which is negligible in our test cases), the number of iterations is no longer a reliable indicator of the efficiency of this method.

Concerning the memory space, the conclusions drawn from previous numerical examples encourage us to consider alternate M-δ methods with M being greater than 2. However it is worth underlining that in this case, $M+1$ iterates must be stored and this may limit the performances of the approach for high dimensional problems. In those cases, a balance would have to be found between acceleration factor and memory space.

6. Conclusions and perspectives

In this paper, we have proposed and analysed a generic residual-based acceleration approach to construct sequence acceleration processes. This approach can be viewed as a generalization of various existing vector acceleration methods (also called extrapolation methods). Then, this generic formalism has been derived in two main classes of methods to accelerate fixed point iterations convergence: the crossed and the alternate sequences approaches.

We proposed to test the performances of the obtained acceleration algorithms on the first order fixed point iterations method obtained from the simulation of mechanical behaviour unit testings when the consistent tangent coherent operator is unknown or difficult to obtain. A set of iterative residual methods based on the two or three last iterates has been compared. It follows that some of the proposed acceleration methods, especially those derived from the alternate approach, are really efficient methods: they are able for example to compete against the second-order Newton–Raphson method.

In the future, it would be worthwhile to confirm the performances of such strategies on structural mechanics solvers involving large numbers of degrees of freedom.

Finally, as the proposed iterative residual-based approach is really generic and independent of the application context, the proposed formalism and the derived algorithms can be useful for many other applications.

Acknowledgements

This work was accomplished within the framework of the PLEIADES project, financially supported by the CEA (Commissariat à l'Énergie Atomique et aux Énergies Alternatives), EDF (Électricité de France) and AREVA.

The test cases used in Section 5 were implemented by Jean-Michel Proix. Authors are grateful to P. Verpeaux and the whole Cast3M development team for fruitful discussions on their use of acceleration methods in the context of structural mechanics.

References

[1] C. Brezinski, Convergence acceleration during the 20th century, J. Comput. Appl. Math. 122 (2000) 1–21.
[2] D. Shanks, Non linear transformations of divergent and slowly convergent sequences, J. Math. Phys. 34 (1955) 1–42.
[3] P. Wynn, Acceleration techniques for iterated vector and matrix problems, Math. Comp. 16 (1962) 301–322.
[4] D.G. Anderson, Iterative procedures for nonlinear integral equationse, J. Assoc. Comput. Mach. 12 (4) (1965) 547–560.
[5] B.M. Irons, R. Tuck, A version of the Aitken accelerator for computer iteration, Internat. J. Numer. Methods Engrg. 1 (1969) 275–277.
[6] C. Brezinski, M. Redivo Zaglia, Extrapolation Methods: Theory and Practice, North Holland, 1991.
[7] P.R. Graves-Morris, Extrapolation method for vector sequences, Numer. Math. 61 (1992) 475–487.
[8] G.A. Sedogbo, Some convergence acceleration processes for a class of vector sequences, Appl. Math. 24 (3) (1997) 299–306.
[9] C. Brezinski, J.-P. Chehab, Nonlinear hybrid procedures and fixed point iterations, Numer. Funct. Anal. Optim. 19 (5–6) (1998) 465–487.
[10] A.C. Aitken, On Bernouilli's numerical solution of algebraic equations, Proc. Roy. Soc. Edinburgh 46 (1926) 289–305.
[11] L.D. Marini, A. Quarteroni, A relaxation procedure for domain decomposition methods using finite elements, Numer. Math. 55 (5) (1989) 575–598.
[12] M. Garbey, Acceleration of the Schwarz method for elliptic problems, SIAM J. Sci. Comput. 26 (2005) 1871–1893.
[13] T. Washio, C.W. Oosterlee, Krylov subspace acceleration for nonlinear multigrid schemes, Electron. Trans. Numer. Anal. 6 (1997) 271–290.
[14] A. Jemcov, J.P. Maruszewski, H. Jasak, Acceleration and stabilization of algebraic multigrid solver applied to incompressible flow problems, in: 18th AIAA Computational Fluid Dynamics Conference, 2007.
[15] U. Küttler, W.A. Wall, Fixed-point fluid–structure interaction solvers with dynamic relaxation, Comput. Mech. 43 (2008) 61–72.
[16] J. Degroote, K.-J. Bathe, Vierendeels, Performance of a new partitioned procedure versus a monolithic procedure in fluid–structure interaction, Comput. Struct. 87 (2009) 793–801.
[17] P. Erbts, A. Düster, Accelerated staggered coupling schemes for problems of thermoelasticity at finite strains, Comput. Math. Appl. 64 (2012) 2408–2430.
[18] B. Michel, C. Nonon, J. Sercombe, F. Michel, V. Marelle, Simulation of pellet-cladding interaction with the PLEIADES fuel performance software environment, Nucl. Technol. 182 (2013).
[19] R.P. Eddy, Extrapolation to the limit of a vector sequence, in: P.C.C. Wang (Ed.), Information Linkage Between Applied Mathematics and Industry, Academic Press, New York, 1979, pp. 387–396.
[20] D. Plancq, G. Thouvenin, J.-M. Ricaud, C. Struzik, T. Helfer, F. Bentejac, P. Thévenin, R. Masson, PLEIADES: a unified environment for multi-dimensional fuel performance modeling, in: International meeting on LWR fuel performance, Florida, 2004.
[21] CEA, EDF, MFront Web Site, 2014. URL: http://www.tfel.sourceforge.net/.
[22] B. Jones, Extrapolation for higher orders of convergence, J. Inst. Math. Appl. 17 (1976) 27–36.
[23] P. Henrici, Elements of Numerical Analysis, John Wiley, New York, 1964.
[24] Y. Nievergelt, Aitken's and Steffensen's accelerations in several variables, Numer. Math. 59 (1) (1991) 295–310.
[25] A. Quarteroni, R. Sacco, F. Saleri, Numerical Mathematics, Springer, 2007.
[26] V. Eyert, A comparative study on methods for convergence acceleration of iterative vector sequences, J. Comput. Phys. 124 (1996) 271–285.
[27] H. Fang, Y. Saad, Two classes of multisecant methods for nonlinear acceleration, Numer. Linear Algebra Appl. 16 (2009) 197–221.
[28] P. Erbts, S. Hartmann, A. Düster, A partitioned solution approach for electro-thermo-mechanical problems, Arch. Appl. Mech. (2014) 1–27.
[29] P. Wynn, On a device for computing the $e_m(S_n)$ transformation, Math. Tables Aids Comput. 10 (1956) 91–96.
[30] D.A. Smith, W.F. Ford, A. Sidi, Extrapolation methods for vector sequences, SIAM Rev. 29 (1987) 199–233.
[31] C. Lemaréchal, Une méthode de résolution de certains systèmes non linéaires bien posés, C. R. Math. Acad. Sci. Paris (1971) 605–607 (in French).
[32] A. Jennings, Accelerating the convergence of matrix iterative processes, J. Inst. Math. Appl. 8 (1971) 99–110.
[33] O.C. Zienkiewicz, R. Lohner, Accelerated relaxation or direct solution. Future prospects for FEM, Internat. J. Numer. Methods Engrg. 21 (1985) 1–11.

[34] A.J. Macleod, Acceleration of vector sequences by multi-dimensional Δ^2 methods, Commun. Appl. Numer. Methods 2 (1986) 385–392.
[35] B. Marder, H. Weitzner, A bifurcation problem in E-layer equilibria, Plasma Phys. 12 (1970) 435–445.
[36] H.F. Walker, P. Ni, Anderson acceleration for fixed-point iterations, SIAM J. Numer. Anal. 49 (4) (2011) 1715–1735.
[37] K. Jbilou, H. Sadok, Vector extrapolation methods. Applications and numerical comparison, J. Comput. Appl. Math. 122 (2000) 149–165.
[38] P. Verpeaux, Algorithmes et méthodes du code éléments finis Cast3M, 2014 (in French). URL: http://www-cast3m.cea.fr/index.php?xml=supportcours.
[39] J. Vierendeels, L. Lanoye, J. Degroote, P. Verdonck, Implicit coupling of partitioned fluid–structure interaction problems with reduced order models, Comput. Struct. 85 (2007) 970–976.
[40] J.C. Simo, R.L. Taylor, Consistent tangent operators for rate-independent elastoplasticity, Comput. Methods Appl. Mech. Engrg. (ISSN: 0045-7825) 48 (1) (1985) 101–118.
[41] Abaqus, Abaqus analysis user's manual, Tech. Rep., Dassault Systèmes, 2015.
[42] ANSYS, USER material subroutine USERMAT, Tech. Rep., ANSYS, Inc., 1999. URL: http://ansys.net/ansys/papers/nonlinear/usermat.pdf.
[43] EDF, Algorithm for nonlinear quasi-static analysis (operator STAT_NON_LINE), Code-Aster reference guide R5.03.01 révision: 10290, EDF-R&D/AMA, 2015. URL: http://www.code-aster.org.
[44] I. Northwest Numerics and Modeling, ZeBuLoN, 2015. URL: http://www.zset-software.com/products/zebulon/.
[45] CEA, Cast3M Web Site, 2015. URL: http://www-cast3m.cea.fr.
[46] J. Besson, D. Desmorat, Numerical implementation of constitutive models, in: J. Besson (Ed.), Local Approach to Fracture, Ecole des Mines de Paris—les presses, 2004.
[47] J.L. Chaboche, Constitutive equations for cyclic plasticity and cyclic viscoplasticity, Int. J. Plast. (ISSN: 0749-6419) 5 (3) (1989) 247–302.
[48] J.L. Chaboche, Cyclic viscoplastic constitutive equations, J. Appl. Mech. 60 (1993) 813–828.
[49] EDF, Elasto-visco-plastic Chaboche constitutive law, Code-Aster reference guide R5.03.04, EDF-R&D/AMA, 2015. URL: http://www.code-aster.org.
[50] M. Berveiller, A. Zaoui, An extension of the self-consistent scheme to plastically-flowing polycrystals, J. Mech. Phys. Solids (ISSN: 0022-5096) 26 (5–6) (1978) 325–344.
[51] EDF, Mono and polycrystalline elastoviscoplastic constitutive laws, Code-Aster reference guide R5.03.11 révision: 10623, EDF-R&D/AMA, 2015. URL: http://www.code-aster.org.