A review of higher order Newton type methods and the effect of numerical damping for the solution of an advanced coupled Lemaitre damage model

Hélène Morch[a, 1], Sibo Yuan[a], Laurent Duchêne[a], Ridha Harzallah[b], Anne Marie Habraken[a,c]

[a] ArGEnCo Department, MSM team, University of Liège, Quartier Polytech 1, allée de la Découverte 9, Liège 4000, Belgium

[b] John Cockerill Energy, avenue Greiner 1, 4100 Seraing, Belgium

[c] Fonds de la Recherche Scientifique – F.N.R.S.–F.R.S., Belgium

Abstract

In this paper, several Newton-type methods of convergence order 2 or higher were tested on various nonlinear systems of equations and on an advanced material law implemented in a finite-element code. The computational speed, numerical efficiency, and robustness of each method were evaluated for each studied case. The effect of numerical damping was also studied. The results were then compared to put in light the strengths and weaknesses of each method. The most efficient and robust method for the material law in the finite-element code was identified as the Newton method with a selective numerical damping.

Keywords: Newton method; Nonlinear equations; Finite-element model; Numerical methods

1 Introduction

In many fields, the design and study of components requires advanced mathematical models. The use of finite-element codes allows to numerically

[1] Corresponding author:
E-mail address: helene.morch@uliege.be (H. Morch)

reproduce the behavior of components subjected to various types of loadings. In this respect, many advanced constitutive laws, such as crystal plasticity models [1] for microscopic analysis or Chaboche-type models [2] for macroscopic approach, have been developed over the years. The goal of these laws is to model the behavior of materials under complex types of loadings such as fatigue, creep, thermal loading, or combinations of these. Ahmed and Hassan [3] developed a behavior model based on the works of Chaboche [4]. This model contains advanced features such as kinematic hardening, static recovery, temperature-dependent parameters, and cyclic hardening. In order to evaluate the lifetime of components, a damage model such as the Lemaitre model can be coupled to the behavior law. Lemaitre and Desmorat [5] proposed a unified damage model taking into account the effect of microdefect closure which can occur when the material is in compression. The resulting coupled model is complex and requires solving a system of 31 nonlinear difference equations corresponding to the evolution of elastic strains, plastic strains, stresses, and back-stresses.

In a general manner, consider a system of nonlinear equations $F(x) = \{f_1(x), f_2(x), \dots, f_n(x)\}^T = 0$, where x is a vector of size n. The most common method to solve this type of system is the Newton method: starting from an initial guess x^0, the solution is obtained by calculating the successive iterations given by (1) until convergence:

$$x^{k+1} = x^k - F'\left(x^k\right)^{-1} F\left(x^k\right) \quad (1)$$

This method is, however, not infallible. This is especially the case when the initial guess is far from the actual solution or when the Jacobian matrix $F'\left(x^k\right)$ is singular ($\det\left(F'\left(x^k\right)\right) = 0$) or nearly singular ($\det\left(F'\left(x^k\right)\right) \approx 0$). In such cases, the numerical computation of the inverse of the Jacobian may be unreliable.

The goal of this study is to identify a robust and efficient method for solving systems of nonlinear equations. The study focuses on gradient-based algorithms, typically Newton-type methods of convergence order 2 or more. Although other methods for solving nonlinear equations exist - such as genetic algorithms [6], particle swarm optimization [7], etc. - this study is limited to gradient-based algorithms as they tend to be more accurate and usually require less computational time and less data storage. Moreover, these methods were better adapted to the finite-element code Lagamine implemented in this study, in which Newton method is already used for solving constitutive equations. Likewise, algorithms that require the computation of a second-order gradient, such as Chebyshev-Halley type methods [8], were not included in the study because the evaluation of the second-order gradient is very costly from a computational point of view, especially for large systems of equations.

2 Numerical study of Newton-type methods

In this part, different Newton-type methods are evaluated on various analytical systems of nonlinear equations. The objective is to quantify the efficiency of each method and compare their speed and robustness.

2.1 Newton-type methods

The study is conducted on 14 methods in addition to the Newton method. Table 1 sums up the different methods studied with their equations, order of convergence, and references. The order of convergence is a measure of how fast the iterative scheme converges towards a solution. It can be determined analytically from the iterative equations of the Newton-type methods.

The Newton method, also known as Newton-Raphson method, is referred to as 'NR'. It is a second-order convergence method. The Wu method [9] is a modified version of Newton's method that aims at preventing the divergence of the

algorithm in cases where $F'(x^k)$ is singular. A non-zero term $diag(v) \cdot diag\left(F(x^k)\right)$ is added to the Jacobian $F'(x^k)$, with $diag(v)$ a diagonal matrix whose coefficients are defined in Table 1. The addition of this term guarantees that the matrix $\left[diag(v) \cdot diag\left(F(x^k)\right) + F'(x^k)\right]$ that is used to compute x^{k+1} can always be inverted.

Several methods with higher orders of convergence are two-step methods in which the first step is identical to a Newton step and the second step is a variation of it. The 2-step Newton method (2-step NR) proposed by Darvishi and Barati [10] simply consists in computing two steps of the Newton method in one iteration. This results in a third-order convergence method. Another way to derive a third-order convergence method from NR is to only reevaluate the Jacobian $F'(x^k)$ every other step, as detailed in [11]. This leads to the method referred to as NR-3 in Table 1. The Arithmetic Mean NR method (AMNR), studied extensively in [12], can be derived from a third-order accurate limited Taylor expansion. It uses the arithmetic mean of the 1st step and 2nd step Jacobian matrices. The Midpoint NR method is obtained by using the midpoint integration rule to derive the second step, as explained in [13] and [14] for the one-dimensional case. The method was extended to the p-dimensional case by Frontini and Sormani [15]. Babajee et al. [16] proposed two third-order Chebyshev-like methods in which the Hessian matrix (second-order derivative) is approximated using Taylor's theorem. The first one is referred to as 'CL' in Table 1 and the second one is equivalent to NR-3. Hasanov et al. [17] derived a third-order convergence method from Newton's equations by using Simpson's rule. Haijun [18] proposed a third-order two-step method similar to the above-mentioned Wu method. The method is equivalent to NR-3 with the addition of a term to the Jacobian to guarantee that the computation of y^k and x^{k+1} is possible even if the Jacobian is singular.

Using two-step methods, it is also possible to obtain fourth-order convergence methods. Jarratt [19] proposed a class of two-step methods of which 2 particular cases are given in Table 1. Sharma et al. [20] developed a method consisting of two weighted-Newton steps. Soleymani [21] proposed a three-step method for the solution of one-dimensional equations. A simplified version of the method, applicable to multidimensional systems of equations, was then proposed by Montazeri et al. [22] and is listed in Table 1 as 'Soleymani'. In the same study, Montazeri et al. also proposed a three-step method which has a sixth-order rate of convergence. Finally, Darvishi and Barati [23] developed a three-step method based on a combination of NR-3 and midpoint NR with a fourth-order convergence.

Table 1 - Newton type methods

Name	Equations	Order	Ref.
NR	$x^{k+1} = x^k - F'(x^k)^{-1} F(x^k)$	2	
Wu	$x^{k+1} = x^k - \left[diag(v) \cdot diag\left(F(x^k)\right) + F'(x^k)\right]^{-1} F(x^k)$ $v_i = sign\left(f_i(x^k)\frac{\partial f_i}{\partial x_i}(x^k)\right) * 1$	2	[9]
2-step NR	$x^{k+1} = y^k - F'(y^k)^{-1} F(y^k)$ $y^k = x^k - F'(x^k)^{-1} F(x^k)$	3	[10]
NR-3	$x^{k+1} = y^k - F'(x^k)^{-1} F(y^k)$ $= x^k - F'(x^k)^{-1}\left(F(x^k) + F(y^k)\right)$ $y^k = x^k - F'(x^k)^{-1} F(x^k)$	3	[11]
AMNR	$x^{k+1} = x^k - \left[\frac{F'(x^k) + F'(y^k)}{2}\right]^{-1} F(x^k)$ $y^k = x^k - F'(x^k)^{-1} F(x^k)$	3	[12]
Midpoint NR	$x^{k+1} = x^k - F'\left(\frac{x^k + y^k}{2}\right)^{-1} F(x^k)$ $y^k = x^k - F'(x^k)^{-1} F(x^k)$	3	[13], [14], [15]

CL	$x^{k+1} = y^k - \frac{1}{2}F'(x^k)^{-1}[F'(y^k) - F'(x^k)](y^k - x^k)$ $y^k = x^k - F'(x^k)^{-1}\,F(x^k)$	3	[16]
Hasanov	$x^{k+1} = x^k - 6\left[F'(y^k) + 4F'\left(\frac{x^k + y^x}{2}\right) + F'(x^k)\right]^{-1} F(x^k)$ $y^k = x^k - F'(x^k)^{-1}\,F(x^k)$	3	[17]
Haijun	$x^{k+1} = x^k + [D(x^k) - F'(x^k)]^{-1}\big(F(x^k) + F(y^k)\big)$ $y^k = x^k + [D(x^k) - F'(x^k)]^{-1}F(x^k)$ $D(x^k) = diag(\alpha_k) \cdot diag\big(F(x^k)\big)$ $\alpha_k = sign\big[diag\big(F'(x^k)\big) \cdot F(x^k)\big]\frac{\max(\lvert f_i(x^k)\rvert)}{\sum_{j=1}^{k}\max(\lvert f_i(x^j)\rvert)}$	3	[18]
Jarratt 1	$x^{k+1} = x^k - \frac{1}{2}F'(x^k)^{-1}\,F(x^k) + [F'(x^k) - 3F'(y^k)]^{-1}F(x^k)$ $y^k = x^k - \frac{2}{3}F'(x^k)^{-1}\,F(x^k)$	4	[19]
Jarratt 2	$x^{k+1} = x^k - F'(x^k)^{-1}\,F(x^k) - \frac{3}{2}F'(y^k)^{-1}F(x^k)$ $+ 3[F'(x^k) + F'(y^k)]^{-1}F(x^k)$ $y^k = x^k - \frac{2}{3}F'(x^k)^{-1}\,F(x^k)$	4	[19]
Sharma	$x^{k+1} = x^k - \frac{1}{2}\left[-I + \frac{9}{4}F'(y^k)^{-1}F'(x^k) + \frac{3}{4}F'(x^k)^{-1}F'(y^k)\right]$ $* F'(x^k)^{-1}F(x^k)$ $y^k = x^k - \frac{2}{3}F'(x^k)^{-1}\,F(x^k)$	4	[20]
Soleymani	$x^{k+1} = x^k - \left[I - \frac{3}{8}\Big(I - \big(F'(y^k)^{-1}F'(x^k)\big)^2\Big)\right] * F'(x^k)^{-1}F(x^k)$ $y^k = x^k - \frac{2}{3}F'(x^k)^{-1}\,F(x^k)$	4	[21], [22]
Montazeri	$x^{k+1} = z^k - \left[\frac{5}{2}I - \frac{3}{2}F'(x^k)^{-1}F'(y^k)\right] * F'(x^k)^{-1}F(z^k)$ $y^k = x^k - \frac{2}{3}F'(x^k)^{-1}\,F(x^k)$	6	[22]

$$z^k = x^k - \left[\frac{23}{8}I - 3F'(x^k)^{-1}F'(y^k) + \frac{9}{8}\left(F'(x^k)^{-1}F'(y^k)\right)^2\right]F'(x^k)^{-1}F(x^k)$$

Darvishi	$x^{k+1} = x^k - \left[\frac{1}{6}F'(x^k) + \frac{2}{3}F'\left(\frac{x^k + z^k}{2}\right) + \frac{1}{6}F'(z^k)\right]^{-1} F(x^k)$ $y^k = x^k - F'(x^k)^{-1}\,F(x^k)$ $z^k = x^k - F'(x^k)^{-1}\left(F(x^k) + F(y^k)\right)$	4	[23]

2.2 Method of evaluation

The objective of the study is to compare the efficiency and the robustness of the methods listed in Table 1. All the methods were tested on 10 systems of nonlinear equations of different sizes found in [9], [10], [16], [18], [20], [22]. The 15 methods and the 10 systems of equations were implemented in a MATLAB program. Table 2 lists the 10 systems $F(x)$. n is the size of the system, $F'(x)$ the Jacobian matrix, and x^* the solution(s) to the system. The solutions x^* are written with a 10-digit precision, corresponding to the convergence criterion later defined.

Table 2 - Systems of nonlinear equations, Jacobian matrices, and solution to the system

n°	n	$F(x)$	$F'(x)$	x^*
1	2	$\begin{Bmatrix} x_1^2 + 3\ln(x_1) - x_2^2 \\ 2x_1^2 - x_1x_2 - 5x_1 + 1 \end{Bmatrix}$	$\begin{bmatrix} 2x_1 + \frac{3}{x_1} & -2x_2 \\ 4x_1 - x_2 - 5 & -x_1 \end{bmatrix}$	$\begin{Bmatrix} 1.3192058033 \\ -1.6035565518 \end{Bmatrix}$
2	3	$\begin{Bmatrix} x_1^2 + x_2^2 + x_3^2 - 1 \\ 2x_1^2 + x_2^2 - 4x_3 \\ 3x_1^2 - 4x_2^2 + x_3^2 \end{Bmatrix}$	$\begin{bmatrix} 2x_1 & 2x_2 & 2x_3 \\ 4x_1 & 2x_2 & -4 \\ 6x_1 & -8x_2 & 2x_3 \end{bmatrix}$	$\begin{Bmatrix} 0.6982886100 \\ 0.6285242980 \\ 0.3425618969 \end{Bmatrix}$
3	2	$\begin{Bmatrix} x_1^5 - x_2 + 1 \\ x_2^5 - x_1 + 1 \end{Bmatrix}$	$\begin{bmatrix} 5x_1^4 & -1 \\ -1 & 5x_2^4 \end{bmatrix}$	$\begin{Bmatrix} -1.1673039783 \\ -1.1673039783 \end{Bmatrix}$
4	3	$\begin{Bmatrix} 0.5x_1^3 - 6x_2^2 + 21.5x_3 - 22 \\ 0.5x_2^3 - 6x_3^2 + 21.5x_1 - 22 \\ 0.5x_3^3 - 6x_1^2 + 21.5x_2 - 22 \end{Bmatrix}$	$\begin{bmatrix} 1.5x_1^2 & -12x_2 & 21.5 \\ 21.5 & 1.5x_2^2 & -12x_3 \\ -12x_1 & 21.5 & 1.5x_3^2 \end{bmatrix}$	$\begin{Bmatrix} 4.0 \\ 4.0 \\ 4.0 \end{Bmatrix}$

5	2	$\begin{Bmatrix} 10x_1e^{-x_2^2} - 1 \\ 10x_2e^{-x_1^2} - 1 \end{Bmatrix}$	$\begin{bmatrix} 10e^{-x_2^2} & -20x_1x_2e^{-x_2^2} \\ -20x_1x_2e^{-x_1^2} & 10e^{-x_1^2} \end{bmatrix}$	$\begin{Bmatrix} 1.6796306104 \\ 1.6796306104 \end{Bmatrix}$ or $\begin{Bmatrix} 0.1010258483 \\ 0.1010258483 \end{Bmatrix}$
6	2	$\begin{Bmatrix} x_1^2 - x_2 + 1 \\ x_1 - \cos\left(\frac{\pi x_2}{2}\right) \end{Bmatrix}$	$\begin{bmatrix} 2x_1 & -1 \\ 1 & \frac{\pi}{2}\sin\left(\frac{\pi x_2}{2}\right) \end{bmatrix}$	$\begin{Bmatrix} -0.7071067812 \\ 1.5 \end{Bmatrix}$
7	3	$\begin{Bmatrix} -x_1^4 + 3x_2^2 + x_3 \\ -x_2^4 + 3x_1^2 + x_3 \\ -x_3^4 + 3x_1x_2 \end{Bmatrix}$	$\begin{bmatrix} -4x_1^3 & 6x_2 & 1 \\ 6x_1 & -4x_2^3 & 1 \\ 3x_2 & 3x_1 & -4x_3^3 \end{bmatrix}$	$\begin{Bmatrix} 0.0 \\ 0.0 \\ 0.0 \end{Bmatrix}$
8	2	$\begin{Bmatrix} x_1 + e^{x_2} - \cos(x_2) \\ 3x_1 - x_2 - \sin(x_2) \end{Bmatrix}$	$\begin{bmatrix} 1 & -e^{x_2} + \sin(x_2) \\ 3 & -1 - \cos(x_2) \end{bmatrix}$	$\begin{Bmatrix} 0.0 \\ 0.0 \end{Bmatrix}$
9	4	$\begin{Bmatrix} x_2x_3 + x_4(x_2 + x_3) \\ x_1x_3 + x_4(x_1 + x_3) \\ x_1x_2 + x_4(x_1 + x_2) \\ x_1x_2 + x_1x_3 + x_2x_3 - 1 \end{Bmatrix}$	$\begin{bmatrix} 0 & x_3 + x_4 & x_2 + x_4 & x_3 + x_2 \\ x_3 + x_4 & 0 & x_1 + x_4 & x_1 + x_3 \\ x_2 + x_4 & x_1 + x_4 & 0 & x_1 + x_2 \\ x_3 + x_2 & x_1 + x_3 & x_1 + x_2 & 0 \end{bmatrix}$	$\begin{Bmatrix} 0.5773502692 \\ 0.5773502692 \\ 0.5773502692 \\ -0.2886751346 \end{Bmatrix}$
10	99	$\begin{Bmatrix} x_1x_2 - 1 \\ x_2x_3 - 1 \\ \vdots \\ x_{98}x_{99} - 1 \\ x_{99}x_1 - 1 \end{Bmatrix}$	$\begin{bmatrix} x_2 & x_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & x_3 & x_2 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & x_{99} & x_{98} \\ x_{99} & 0 & 0 & 0 & \cdots & 0 & x_1 \end{bmatrix}$	$\begin{Bmatrix} 1 \\ \vdots \\ 1 \end{Bmatrix}$ or $\begin{Bmatrix} -1 \\ \vdots \\ -1 \end{Bmatrix}$

The starting point of the algorithm plays a crucial role in the capacity of the method to converge or not. It is well-known that the Newton method works best when the starting point is close to the solution. However, a robust algorithm should converge even when starting from a point far from the solution. To evaluate the robustness of the Newton-type methods, each system was tested with 3x3 different starting points:

- 3 values of deviation ε from the solution were used: $(0.05; 0.5; 5.0)$
- For each ε, the 3 following starting rules were tested:
  - $\begin{Bmatrix} x_1^0 \\ \dots \\ x_n^0 \end{Bmatrix} = \begin{Bmatrix} x_1^* + \varepsilon \\ \dots \\ x_n^* + \varepsilon \end{Bmatrix}$;
  - $\begin{Bmatrix} x_1^0 \\ \dots \\ x_n^0 \end{Bmatrix} = \begin{Bmatrix} x_1^* - \varepsilon \\ \dots \\ x_n^* - \varepsilon \end{Bmatrix}$;

- $\begin{Bmatrix} x_1^0 \\ \dots \\ x_n^0 \end{Bmatrix} = \begin{Bmatrix} x_1^* + (-1)^{r_1}\varepsilon \\ \dots \\ x_n^* + (-1)^{r_n}\varepsilon \end{Bmatrix}$ where r_i is assigned the value of 0 or 1 at random.

Note that, neglecting the particular cases of a null solution for systems 7 and 8, the solutions of the studied systems are of a similar order of magnitude ($|x_i^*| < 5$), so the values chosen for the deviation are consistent with how far from the solution the algorithm starts.

The algorithms were stopped after a maximum number of 250 iterations or when the following convergence criterion was achieved, where $\|.\|$ is the L^2 norm (Euclidian norm):

$$\left\|x^{k+1} - x^k\right\| + \left\|F\left(x^k\right)\right\| < 10^{-10} \quad (2)$$

To assess the efficiency of the algorithms, two requirements were considered: a low number of iterations and a low computation time. The computation time (CPU time) was obtained using the MATLAB *timeit* function. To evaluate the robustness, a failure rate was calculated. The algorithm was considered to have failed if the convergence criterion has not been met within a maximum number of 250 iterations.

2.3 Results

Figure 1 shows the average number of iterations required to reach the solution for each method with consideration of the initial deviation ε. The average covers the 10 systems of equations and the 3 starting rules of section 2.2. The results only include cases where the method converges in order to evaluate the speed of the methods independently from their robustness. The results are presented by increasing global number of iterations, i.e., 2-step NR is the one that has the smallest number of iterations all deviations considered. It can be observed that second-order methods (NR and Wu) are amongst the slowest in terms of iterations.

However higher-order methods do not necessarily perform better, as can be seen with Jarrat 2 (4^{th}-order), Soleymani (4^{th}-order) and CL (3^{rd}-order).

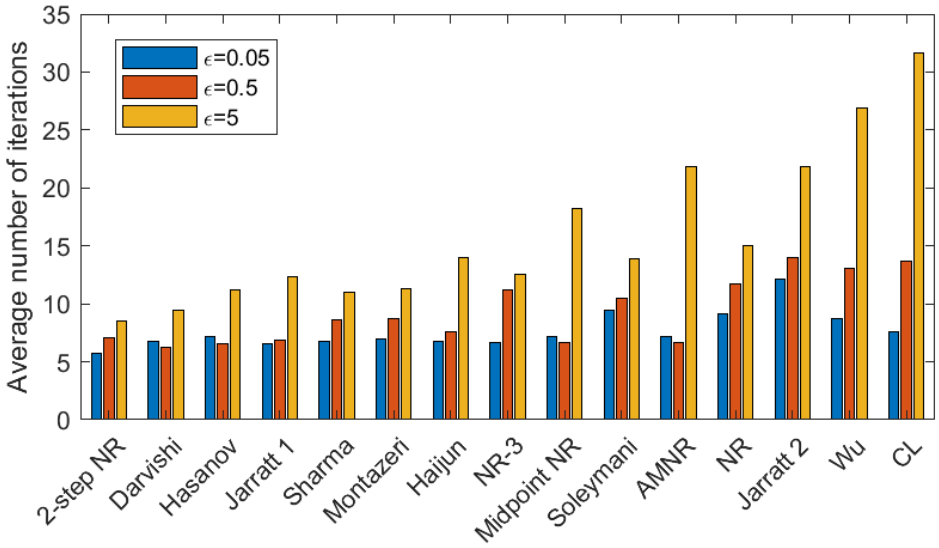

Figure 1 - Average number of iterations to reach solution when algorithm is successful

With regards to the number of iterations, 2-step NR seems to be the most efficient method. However, comparing the CPU time of the different methods shows this is not really the case. In Figure 2, the methods are ranked by increasing CPU time. Again, the average of the successful cases is used. NR is the fastest method, even though it requires more iterations than most methods. The reason behind this is that one iteration of NR method is much faster to compute than one iteration of any other method. This can easily be understood when comparing NR and 2-step NR. The 2-step NR simply consists in computing 2 steps of the NR method in one iteration (see equations in Table 1). From a computational point of view, the only difference between the two methods is that the convergence criterion is only checked every other step in the 2-step NR method. Since the CPU time required for checking convergence is negligible compared to the CPU time required for calculating a Newton step, the overall CPU times of the two methods are equivalent. Moreover, in some cases NR reaches convergence after an odd number

of iterations, that is, using an odd number of steps. This means 2-step NR requires one additional superfluous step, which makes it slightly more costly.

It can also be observed in Figure 2 that the CPU time for a given method increases with the initial deviation from the solution. This is also generally the case for the number of iterations (Figure 1).

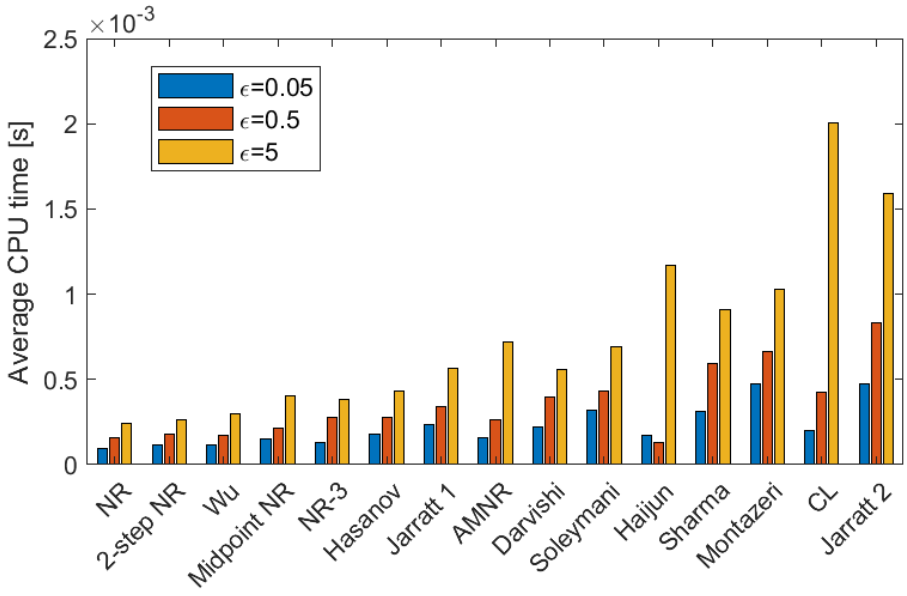

Figure 2 - Average computation time when algorithm is successful

Figure 3 gives a better understanding of the relation between the computation time and the number of iterations. For each test performed (9*10 tests per algorithm), the CPU time is plotted as a function of the number of iterations N_{iter}. The size of the point corresponds to the maximum value of the condition number of the Jacobian $cond\left(F'\left(x^k\right)\right)$ over all iterations. More precisely, the size s in points squared is expressed as:

$$s = 10 * \log_{10}\left(\max_k\left(cond\left(F'\left(x^k\right)\right)\right)\right) \quad (3)$$

In the case where $s = \infty$, the point is represented by a cross. The vertical group of points at the far right of the graph ($N_{iter} = 250$) corresponds to tests that did not converge. For many of them, this can be linked to a badly conditioned Jacobian

(high or infinite condition number). In general, however, there is no direct correlation between the condition number and the success or failure of the algorithm: some failures occur despite small condition numbers, and algorithms converge in cases where the maximum condition number is high.

Two distinct groups of points appear in Figure 3. The upper group (higher CPU times) was identified to correspond to tests performed with system of equations n°10, which is of size 99. This can be explained easily, as computations made with matrices of order 99 are much more costly than similar calculations with matrices of order 2 to 4.

For both groups of points, it can be observed that the tests that converged rapidly ($N_{iter} \leq 10$) correspond to small values of the maximum condition number. Tests with higher condition numbers tend to converge more slowly.

It can also be observed that the vertical scattering of the points seems to depend on the order of convergence of the method used. Indeed, the points corresponding to methods with low order of convergence (particularly NR and Wu, represented in dark blue in Figure 3) tend to be at the bottom of each group of points, which corresponds to low CPU times. In contrast, methods of convergence order four or higher (in colors ranging from yellow to red) tend to be on the upper part of the group of points. This is consistent with the observations made above: methods with higher orders of convergence require more complex calculations at every iteration and can therefore be less efficient in regard to the computation time.

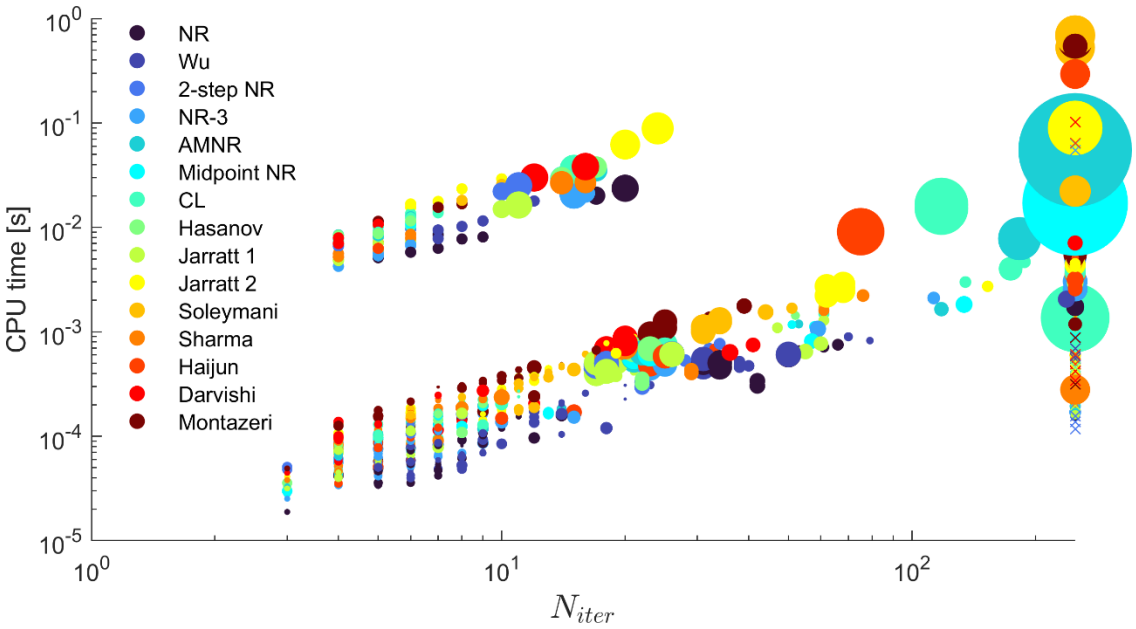

Figure 3 - Number of iterations versus CPU time for the different tests; the size of the points depends on the maximum condition number of the Jacobian over all iterations; an infinite value is represented by a cross

Finally, the robustness of the methods is evaluated using the failure rate. Figure 4 shows the failure rate of each method, defined as the proportion of computations that did not reach convergence within 250 iterations, among the 10 systems of equations and the 3 starting rules. The failure rate is shown for the 3 different initial values of deviation and the methods are ranked according to the mean failure rate. Sharma, Jarratt 1, NR and 2-step NR are the best performing methods when $\varepsilon \leq 0.5$ with 0% failure rates (meaning the method always converges towards the solution). However, for a high value of initial deviation, these methods have a failure rate higher than 20%. It is important to note that for a deviation $\varepsilon = 5.0$, none of the methods is very robust. The minimum failure rate is obtained with Hasanov and is approximately 21%, which is not negligible. Globally, the most robust methods are Sharma, Hasanov, Jarratt 1, and Midpoint NR.

The methods of Wu and Haijun, although developed to be more robust than Newton's method (see section 2.1), actually have higher failure rates than NR. In these methods, the addition of a term to the Jacobian allows the algorithm to continue even when the Jacobian would be singular, but it also makes the direction

of progression of the algorithm less accurate and therefore these methods are more likely to diverge.

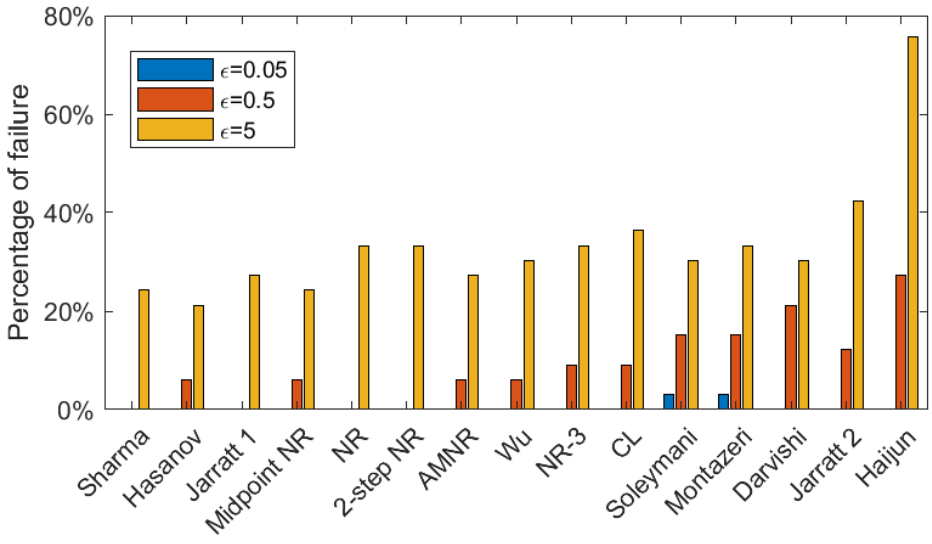

Figure 4 - Failure rate

The success of a method depends not only on the initial deviation from the solution, but also on the system of equations to be solved. Indeed, Figure 5 shows the percentage of failure obtained with each method for systems of equations n°2 (F_2) and n°10 (F_{10}) separately (see Table 2). Soleymani and Montazeri are the only methods that have a 0% failure rate for F_2. However, both methods have a failure rate of 33% for F_{10} while most other methods are always successful. Consequently, although some methods seem to give better results overall, it is not possible to infer that these methods would be the best suited for any given problem. For cases that are particularly problematic, no method can a priori guarantee a good result. The methods have to be tested in order to find which one is best suited for the problem.

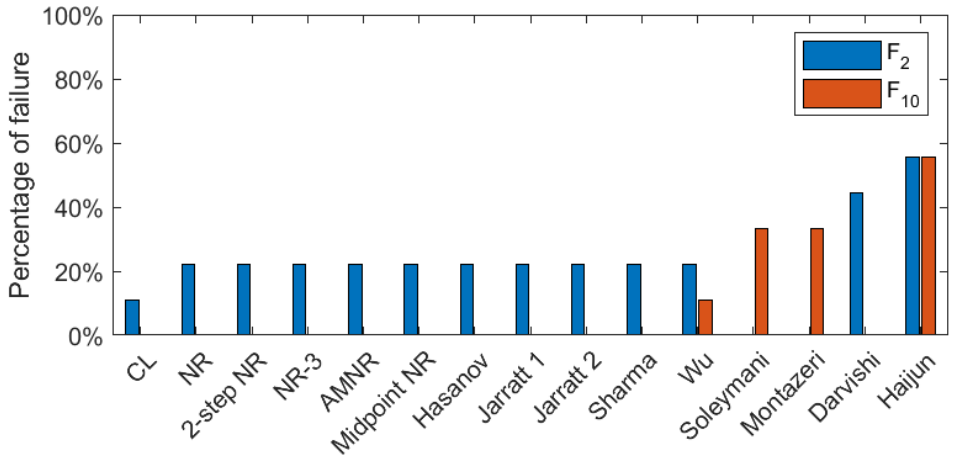

Figure 5 - Percentage of failure for systems of equations n°2 and 10

2.4 Effect of numerical damping

The use of numerical damping is known to improve the robustness of the Newton method, especially in cases where the initial value is far from the solution. Other techniques can be combined with the Newton method to improve its robustness such as trust-region type methods, however only damping was considered in this study because of its simplicity to implement.

Numerical damping consists in applying a coefficient $0 < \alpha < 1$ to the second term in order to reduce the correction of each Newton step, as written in Eq. (4). The algorithm is therefore slower (a larger number of iterations is expected since the progression at each step is smaller) but less likely to go in the wrong direction, i.e., more robust [24].

$$x^{k+1} = x^k - \alpha F'(x^k)^{-1} F(x^k) \quad (4)$$

Ideally, the value of coefficient α should be dependent on the value of $F(x^k)$. At the beginning of the iterative process during a computation, when x^k can be far from the solution and $F(x^k)$ far from 0, a low value of α should be used to activate numerical damping. For values of $F(x^k)$ close to zero – i.e., when the algorithm is close to the solution – α should be close to 1 in order to accelerate the convergence

of the method. In this study, for simplicity, a fixed value of 0.5 (or 1 when damping is not activated) was chosen to study the effect of numerical damping.

To test damping on methods with 2 steps, a damping coefficient is applied on the first step (coefficient α) and another one is applied on the second step (coefficient β). Eq. (5) shows an example of the application of the damping coefficients for the 2-step NR method (NB: Coefficient β is not applicable on 1-step algorithms like NR and Wu). For 3-step methods, the coefficients are only applied on the 1st step (α for y^k) and 2nd step (β for z^k).

$$\begin{aligned} x^{k+1} &= y^k - \beta F'(y^k)^{-1} F(y^k) \\ y^k &= x^k - \alpha F'(x^k)^{-1} F(x^k) \end{aligned} \tag{5}$$

The 2 damping coefficients are set to either 1 or 0.5 independently. In this way, the effect of damping on the 1st, 2nd, or both steps is studied.

The effect of damping is evaluated on the systems of equations detailed in Table 2 using the methodology described in section 2.2. The value of initial deviation is set to $\varepsilon = 5$ only as it is the value that leads to the highest failure rates.

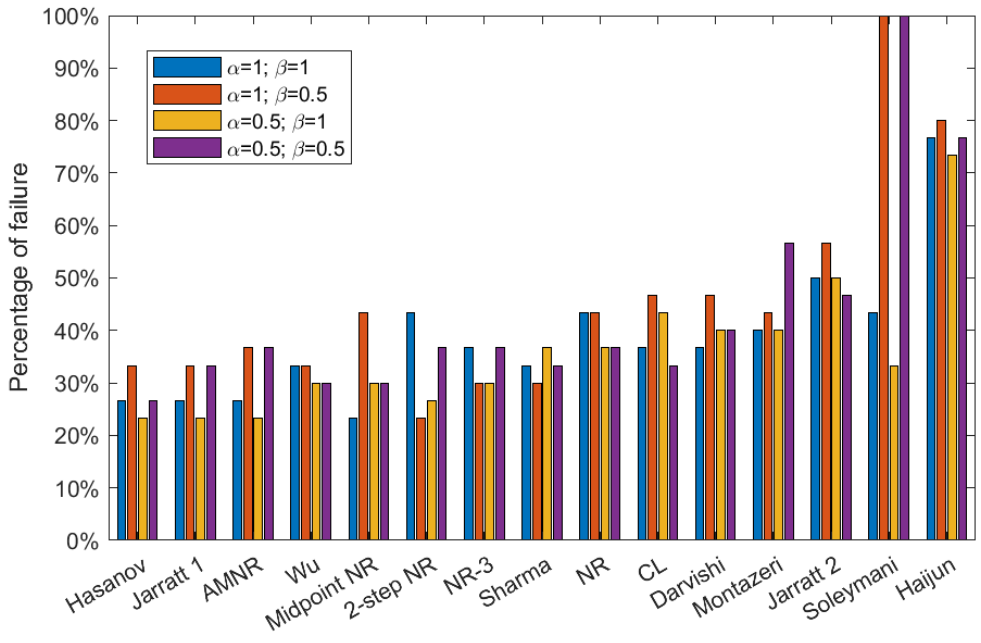

Figure 6 - Effect of numerical damping on the failure rate of Newton-type methods

Figure 6 shows the failure rate obtained for the different methods for each combination of damping coefficients. The blue bar corresponds to the method with no numerical damping. Note that the results can differ slightly from the results shown in Figure 4 because the 3rd rule for the definition of the starting point includes random numbers, which are different from one computation to another (see section 2.2). However, for each system of equations, the same 3 starting points are used for all algorithms and damping values, therefore their respective efficiency can be compared. For NR and Wu, which are 1-step methods, the value of β has no influence, therefore the blue and red bars (1st and 2nd), and yellow and purple (3rd and 4th) are the same. For both methods, damping slightly improves the robustness.

The case of 2-step NR is also interesting: when damping is not applied or applied on both steps, the failure rate is the same as NR. However, when damping is applied only on the 1st or 2nd step ($\{\alpha;\beta\} = \{1; 0.5\}$ or $\{0.5; 1\}$), the failure rate is significantly lower. This indicates that numerical damping on Newton method can actually be more efficient if applied only every other iteration.

Globally, the effect of damping varies depending on the method. For some methods, damping improves or does not significantly affect robustness, while for others, damping can actually have a negative impact. This is the case for the Soleymani method where the value $\beta = 0.5$ leads to a 100% failure rate. Overall, numerical damping is not as effective as could be expected for the set of nonlinear systems studied.

3 Application to the Lemaitre-Chaboche model

In finite-element models, the various state variables – stresses, strains, hardening, etc. – are computed for each integration point of each element at each time step in the material law. The material law usually consists of a set of nonlinear difference equations that describe the behavior of a specific material. In this study, a

Lemaitre-Chaboche type law was used to model the behavior of nickel-alloy HAYNES 230 at high temperature.

Within a finite-element model, a failure of the method to solve the equations of the constitutive law can lead to a reduction of the time increment which can greatly impact the overall computation time, or a total failure of computation due to divergence problems. Therefore, a robust method is essential to ensure that the model works properly.

3.1 Lemaitre-Chaboche advanced model

The model used is based on the works of Ahmed and Hassan [3] for the behavior law and on the works of Lemaitre and Desmorat [5] for the damage evolution. A detailed description of the equations of the behavior model without damage can be found in [25]. The effect of damage is coupled to the behavior model using the isotropic damage variable D. The full model was implemented in the finite-element code Lagamine [26] developed at the University of Liège. In the code, the Newton method is used to solve the system of nonlinear equations presented hereafter.

The equations of the coupled model are presented in their residual form in equations (7) to (10). In the following equations, the difference operator Δ is used to indicate the variation of a variable z on the finite-element time step: $\Delta z = z_{n+1} - z_n$. For simplification, values at the end of the time step z_{n+1} are written without the subscript: $z_{n+1} = z$. Variables written in bold are symmetrical 3x3 second-order tensors which are transformed in a contracted form as 6x1 vectors, as shown in Eq. (6) for the stress. As a result, the system to be solved is of size 31 – 5 unknowns of size 6: $\Delta\boldsymbol{\varepsilon^e}$, $\Delta\boldsymbol{\sigma}$, $\Delta\boldsymbol{X_i}$ with $i = 1{:}\,3$ and 1 unknown of size 1: Δr.

$$\begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{bmatrix} = \begin{bmatrix} \sigma_1 & \sigma_4 & \sigma_5 \\ \sigma_4 & \sigma_2 & \sigma_6 \\ \sigma_5 & \sigma_6 & \sigma_3 \end{bmatrix} \equiv \begin{Bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \sigma_4 \\ \sigma_5 \\ \sigma_6 \end{Bmatrix} = \boldsymbol{\sigma} \tag{6}$$

326

$$\boldsymbol{R}_{\boldsymbol{\varepsilon}^e} = \Delta\boldsymbol{\varepsilon}^{\boldsymbol{e}} + \Delta\boldsymbol{\varepsilon}^{\boldsymbol{th}} + \Delta r\boldsymbol{n} - \Delta\boldsymbol{\varepsilon} = \boldsymbol{0} \tag{7}$$

$$R_r = \|\boldsymbol{s} - \boldsymbol{X}\|_{VM} - R(r) - \sigma_y = 0 \tag{8}$$

$$\boldsymbol{R}_{\boldsymbol{\sigma}} = \boldsymbol{\varepsilon}^{\boldsymbol{e}} - \frac{(1+\nu)}{E}\left[\frac{\langle\boldsymbol{\sigma}\rangle^+}{1-D} + \frac{\langle\boldsymbol{\sigma}\rangle^-}{1-hD}\right] + \frac{\nu}{E}\left[\frac{\langle tr\boldsymbol{\sigma}\rangle}{1-D} - \frac{\langle -tr\boldsymbol{\sigma}\rangle}{1-hD}\right]\boldsymbol{I}_{\mathbf{6x1}} = \boldsymbol{0} \tag{9}$$

$$\boldsymbol{R}_{\boldsymbol{X}_i} = \Delta\boldsymbol{X}_{\boldsymbol{i}} - \frac{2}{3}\Delta r C_i \boldsymbol{n}^{\boldsymbol{X}} + \gamma_i(\boldsymbol{X}_{\boldsymbol{i}} - \boldsymbol{Y}_{\boldsymbol{i}})\Delta r + b_i\|\boldsymbol{X}_{\boldsymbol{i}}\|_{\mathrm{VM}}^{r_i - 1}\Delta t\boldsymbol{X}_{\boldsymbol{i}} - \frac{1}{C_i}\frac{\partial C_i}{\partial T}\Delta T\boldsymbol{X}_{\boldsymbol{i}} = \boldsymbol{0} \tag{10}$$

with $i = 1{:}3$

327 In Eq. (7), the total strain $\boldsymbol{\varepsilon}$ is decomposed as the sum of the elastic strain $\boldsymbol{\varepsilon}^{\boldsymbol{e}}$, the
328 plastic strain $\boldsymbol{\varepsilon}^{\boldsymbol{p}}$, and the thermal strain $\boldsymbol{\varepsilon}^{\boldsymbol{th}}$. The plastic strain variation $\Delta\boldsymbol{\varepsilon}^{\boldsymbol{p}}$ is
329 expressed as $\Delta r\boldsymbol{n}$ where Δr is the variation of the equivalent plastic multiplier p
330 corrected by the damage value: $\Delta r = \Delta p(1-D)$, and $\boldsymbol{n}$ is the plastic normal
331 defined by Eq. (11), where $\|.\|_{VM}$ is the von Mises equivalent operator, $\boldsymbol{X} =$
332 $\sum_{\boldsymbol{i=1}}^{\boldsymbol{3}}\boldsymbol{X}_{\boldsymbol{i}}$ is the total back-stress, and $\widehat{\boldsymbol{\sigma}}$ the deviatoric stress.

$$\boldsymbol{n} = \frac{3}{2}\frac{1}{1-D}\frac{\boldsymbol{s}-\boldsymbol{X}}{\|\boldsymbol{s}-\boldsymbol{X}\|_{VM}} \tag{11}$$

with $\boldsymbol{s} = \frac{\widehat{\boldsymbol{\sigma}}}{1-D}$

333 Equation (8) corresponds to the von Mises yield criterion. $R(r)$ is the isotropic
334 hardening defined by Eq. (12), where b and Q are material parameters. σ_y is the
335 yield stress.

$$R(r) = Q(1 - \exp(-br)) \tag{12}$$

Eq. (9) establishes the relation between the stress and the elastic strain according to a modified Hooke's law, with E the Young modulus and ν the Poisson ratio. Indeed, the effect of microdefects closure is considered in this model. As a result, the relation between $\boldsymbol{\sigma}$ and $\boldsymbol{\varepsilon^e}$ is not direct. The microdefects closure effect is modeled by considering a smaller fraction of damage hD (where $h < 1$ is a material parameter) when the stress is negative, i.e., when the material is under compressive stress. To do so, the stress tensor $\boldsymbol{\sigma}$ needs to be decomposed into a positive part $\langle\boldsymbol{\sigma}\rangle^+$ and a negative part $\langle\boldsymbol{\sigma}\rangle^-$, as defined by Eq. (13)**,** where σ_K are the eigenvalues of $\boldsymbol{\sigma}$, $\boldsymbol{q}^K$ the corresponding eigenvectors, and $\langle . \rangle$ the Macauley brackets.

$$\langle\boldsymbol{\sigma}\rangle_{\boldsymbol{ij}}^{+} = \sum_{K=1}^{3}\langle\sigma_K\rangle\, \boldsymbol{q}_i^K \boldsymbol{q}_j^K \text{ and } \langle\boldsymbol{\sigma}\rangle_{\boldsymbol{ij}}^{-} = \boldsymbol{\sigma_{ij}} - \langle\boldsymbol{\sigma}\rangle_{\boldsymbol{ij}}^{+} \tag{13}$$

$\boldsymbol{I_{6x1}}$ is the 6-component vector form of the second-order identity tensor.

Finally, Eq. (10) defines the evolution of the 3 back-stresses $\boldsymbol{X_i}$ that model kinematic hardening. C_i, γ_i, b_i and r_i are material parameters, T is the temperature. The normal $\boldsymbol{n^X}$ is defined by Eq. (14). The three first terms of Eq. (10) correspond to the Armstrong-Fredericks hardening rule [27]. The 4th term is used to model static recovery, and the 5th term is used to take into account the effect of temperature variation. The variable $\boldsymbol{Y_i}$ models the evolution of the cyclic mean stress under creep-fatigue loading conditions that results from the effect of the dislocation network, as defined by Yaguchi et al. [28]. Its evolution is defined by Eq. (15), where Δt is the time increment, $\alpha_{b,i}$ is the rate of evolution of $\boldsymbol{Y_i}$**,** and $Y_{st,i}$ controls the saturated value of $\boldsymbol{Y_i}$.

$$\boldsymbol{n^X} = (1 - D)\boldsymbol{n} \tag{14}$$

where $\boldsymbol{Y_i} = \left[\boldsymbol{Y_{i,n}} - \frac{3}{2}\Delta t \alpha_{b,i} Y_{st,i} \|\boldsymbol{X_i}\|_{VM}^{r_i - 1} \boldsymbol{X_i}\right] \left[1 + \Delta t \alpha_{b,i} \|\boldsymbol{X_i}\|_{VM}^{r_i}\right]^{-1}$ (15)

The damage is semi-coupled with the behavior: at every time step, equations (7)-(10) are solved considering a constant value of damage on the step equal to D_n (value from the previous step). The equations for the calculation of D at the end of the current step are not detailed here as they are not relevant to the study, but can be found in [5].

The system to solve can be rewritten as Eq. (16), where the total vector of the unknowns is written as $\mathcal{X}$ to avoid confusion with the back-stress $\boldsymbol{X}$:

$$\{\mathcal{R}(\mathcal{X})\} = \begin{Bmatrix} \boldsymbol{R_{\varepsilon^e}} \\ R_r \\ \boldsymbol{R_\sigma} \\ \boldsymbol{R_{X_1}} \\ \boldsymbol{R_{X_2}} \\ \boldsymbol{R_{X_3}} \end{Bmatrix} = \{0\} \tag{16}$$

The Jacobian matrix of the system is detailed in Appendix A. It is important to note that because of the decomposition of the stress tensor in Eq. (9), there is no exact analytical expression for $\frac{\partial \boldsymbol{R_\sigma}}{\partial \boldsymbol{\Delta\sigma}}$; the expression shown in Appendix A is an approximation. This can have an impact on the efficiency of Newton-type methods, as the Jacobian matrix defines the direction of progress of the algorithms.

In the case where $h = 1$ however, Eq. (9) can be rewritten as Eq. (17):

$$\boldsymbol{R_\sigma} = \boldsymbol{\varepsilon^e} - \frac{(1+\nu)}{E}\frac{\boldsymbol{\sigma}}{1-D} + \frac{\nu}{E}\frac{tr\boldsymbol{\sigma}}{1-D}\,\boldsymbol{I_{6x1}} = \boldsymbol{0} \tag{17}$$

The expression for $\frac{\partial \boldsymbol{R_\sigma}}{\partial \boldsymbol{\Delta\sigma}}$ is therefore very easy to derive and the equations can be solved more easily.

3.2 Test of the methods

3.2.1 Methodology

In the material law, the initial point $\mathcal{X}^0$ is determined based on the results of the previous time step. Thus, the methodology proposed in section 2.2 with varying starting points to test the different methods is not suitable to establish the robustness and efficiency of the Newton-type methods within the material law. Instead, it was decided to test the methods using values of $\mathcal{X}^0$ obtained from an actual finite-element simulation using the Lagamine code [26]. The chosen finite-element simulation consists in a single-element case with imposed displacements along the x direction, as shown in Figure 7 (a). The methods were tested at 4 different time instants of a cyclic loading shown in Figure 7 (b) where the solution of system (16) was likely to be unsuccessful. In the finite-element code, a failure to solve the equations within the material law results in a reduction of the time increment, which increases the overall computation time. Therefore, it is interesting to increase the robustness of the iterative algorithm in the material law to avoid too many step reductions. In the present comparative analysis, the chosen time increment corresponds to its actual value in the finite-element simulation before reduction due to non-convergence. Table 3 sums up the different time instants used for testing and their specificity. The yield point (point 1) is likely to cause problems because it is a plastic step preceded by an elastic step. When estimating the initial value of $\mathcal{X}^0$, the code will therefore consider no plastic strain and overestimate the elastic strain and the stress. Similarly, when a change occurs in the loading direction (points 3 and 4), the initial guess for the variation of stress will be made in the direction of evolution of the previous step. For instance, at point 4 the initial estimate will consider a positive $\Delta\boldsymbol{\sigma}$ when in reality the stress drops due to relaxation. Finally, point 2 was added to the study because the finite-

element code did not converge immediately at that point, which led to a time increment reduction.

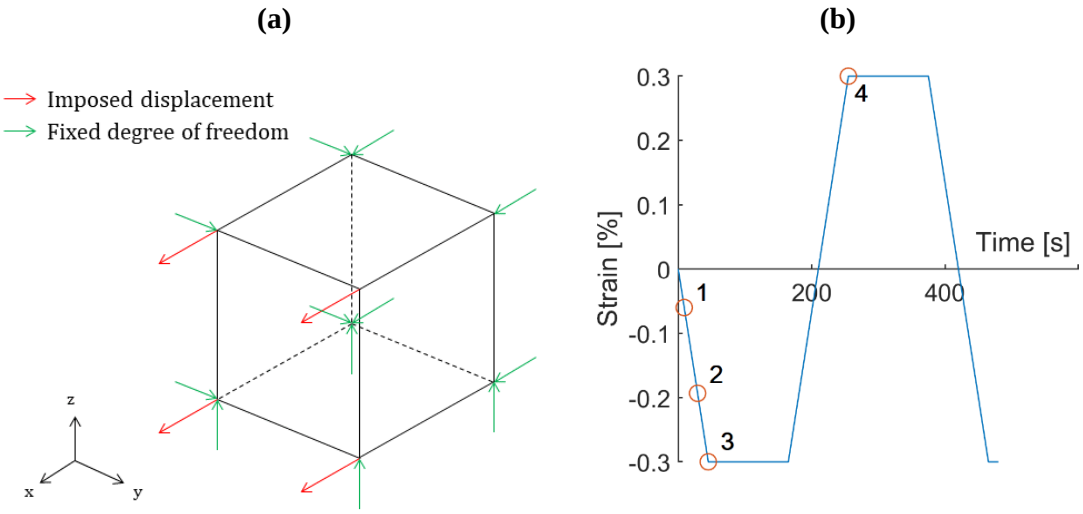

Figure 7 - Finite-element modeling (a) scheme of the geometry and boundary conditions; (b) Representation of the cyclic loading with identification of the 4 studied time steps

Table 3- Definition of studied time instants

N°	Time instants	Step size Δt	Meaning
1	9s	2.5	Yield point
2	29s	2.5	No convergence in NR
3	45s	1.378640	Change in loading direction
4	255s	1.5	Change in loading direction

To test the different Newton-type methods on the finite-element model, the precision for the norm of convergence was set to 10^{-10} and the maximum number of iterations was 100.

3.2.2 Results

The different methods were first tested for the constitutive law with $h = 1$, i.e., with a closed-form expression of the Jacobian and a more straightforward relation between the stress and the elastic strain (see Eq. (17)).

Figure 8 shows the number of iterations calculated for each method and for the 4 studied time instants depending on the damping coefficients. When the number of iterations reaches 100, it means the method failed to converge. The methods of Soleymani, Haijun, and Montazeri never converged towards a solution within the required precision, due to the Jacobian becoming singular after a few iterations (Soleymani and Montazeri) or because they ended up diverging (Haijun). The rest of the methods gave good results without damping, with the Wu method being the slowest. As can be seen in Figure 8 (b), (c), and (d), damping does not improve the results in this case and only leads to a higher number of iterations.

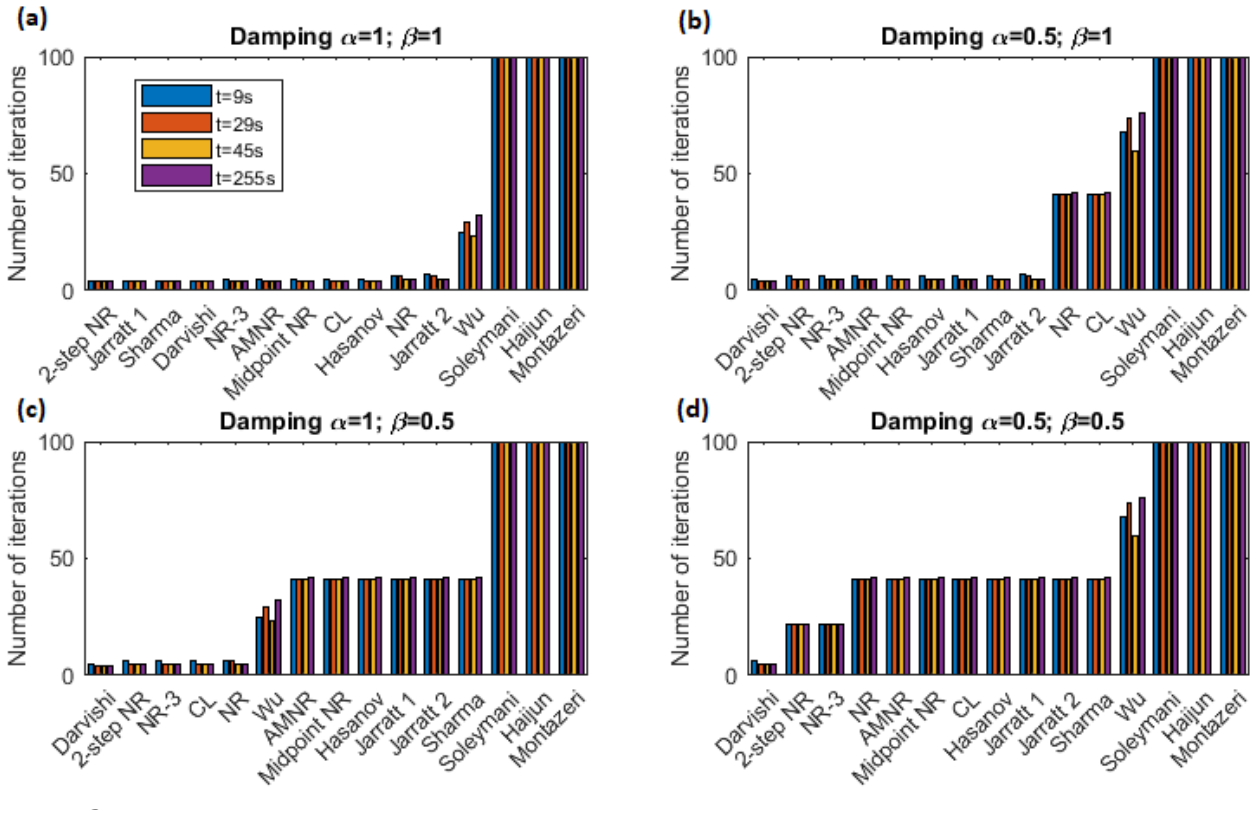

Figure 8 - Number of iterations required for the methods to converge at 4 different time instants for the constitutive law with h=1 with (a) no damping; (b) damping on the 1st step; (c) damping on the 2nd step; (d) damping on 1st and 2nd steps.

The average CPU times obtained for the algorithm that converged are shown in Figure 9. In the case with no damping, NR is the fastest, although it takes more iterations than most methods (see Figure 8 (a)). This is consistent with the results discussed in Section 2.3. NR only requires the computation of one Jacobian per

iteration, contrarily to most of the other methods that require the computation of two or three Jacobians at every iteration. In the case where damping is applied on both steps, the method of Darvishi is the fastest, as well as the one that requires the smallest number of iterations. Once again, NR is among the fastest methods in this case as well.

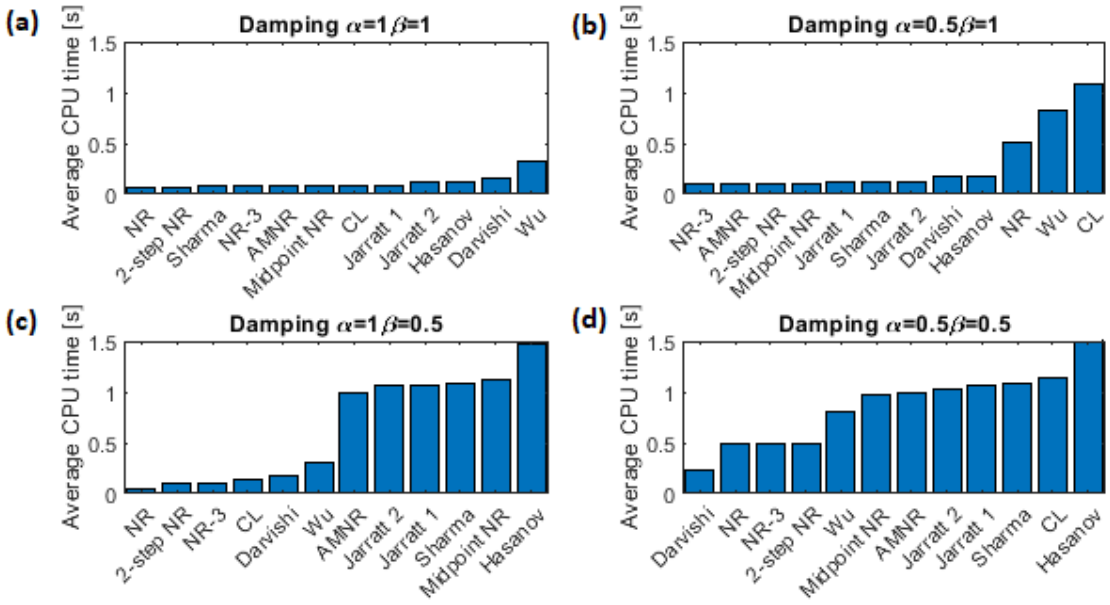

Figure 9 - Average CPU time of the converging algorithm with (a) no damping; (b) damping on the 1st step; (c) damping on the 2nd step; (d) damping on 1st and 2nd steps.

Figure 10 shows the number of iterations calculated for each method in the case where $h < 1$, i.e., when the Jacobian is approximated and the relation between the stress and the elastic strain is not direct. Again, a number of iterations equal to 100 means failure to convergence. The methods of Montazeri, Haijun, Jarratt 1, Soleymani, and Wu never converged, due to a singular Jacobian (Montazeri, Jarratt 1, Soleymani) or divergence (Haijun, Wu). Therefore, the results for these methods are not shown in Figure 10. As shown in section 2.3, the success of each method is highly dependent on the system to solve. These 5 methods are not efficient for the problem considered; however, they can give good results on other equations.

Without damping (see Figure 10 (a)), none of the methods can solve the equations of the studied model at every time instant considered. Sharma and NR are the methods that give the best results, as they lead to a solution in 2 out of the 4 cases.

When damping is applied on the first step, i.e., $\{\alpha;\beta\} = \{0.5; 1\}$, NR and CL become more robust: both these methods converge for the 4 studied cases (see Figure 10 (b)). However, this has a significant impact on the convergence rate, as both methods require around 40 iterations to converge while less than 10 iterations were necessary without damping (for the successful cases). Overall, the use of damping on the 1st step improves the robustness of most algorithms. When damping is applied on the 2nd step ($\{\alpha;\beta\} = \{1; 0.5\}$), several of the 2-step methods become more robust (Figure 10 (c)). Once again, this results in a significant increase of the number of iterations required to reach the solution. When damping is applied on both steps (Figure 10 (d)), all methods prove to be robust, with the exception of the Darvishi method which does not converge for time t=255s. However, it is important to note that the convergence rate of Darvishi is barely affected by damping. While all other methods show a significant increase in the number of iterations when damping is applied on both steps, Darvishi method converges in less than 10 iterations when it is successful.

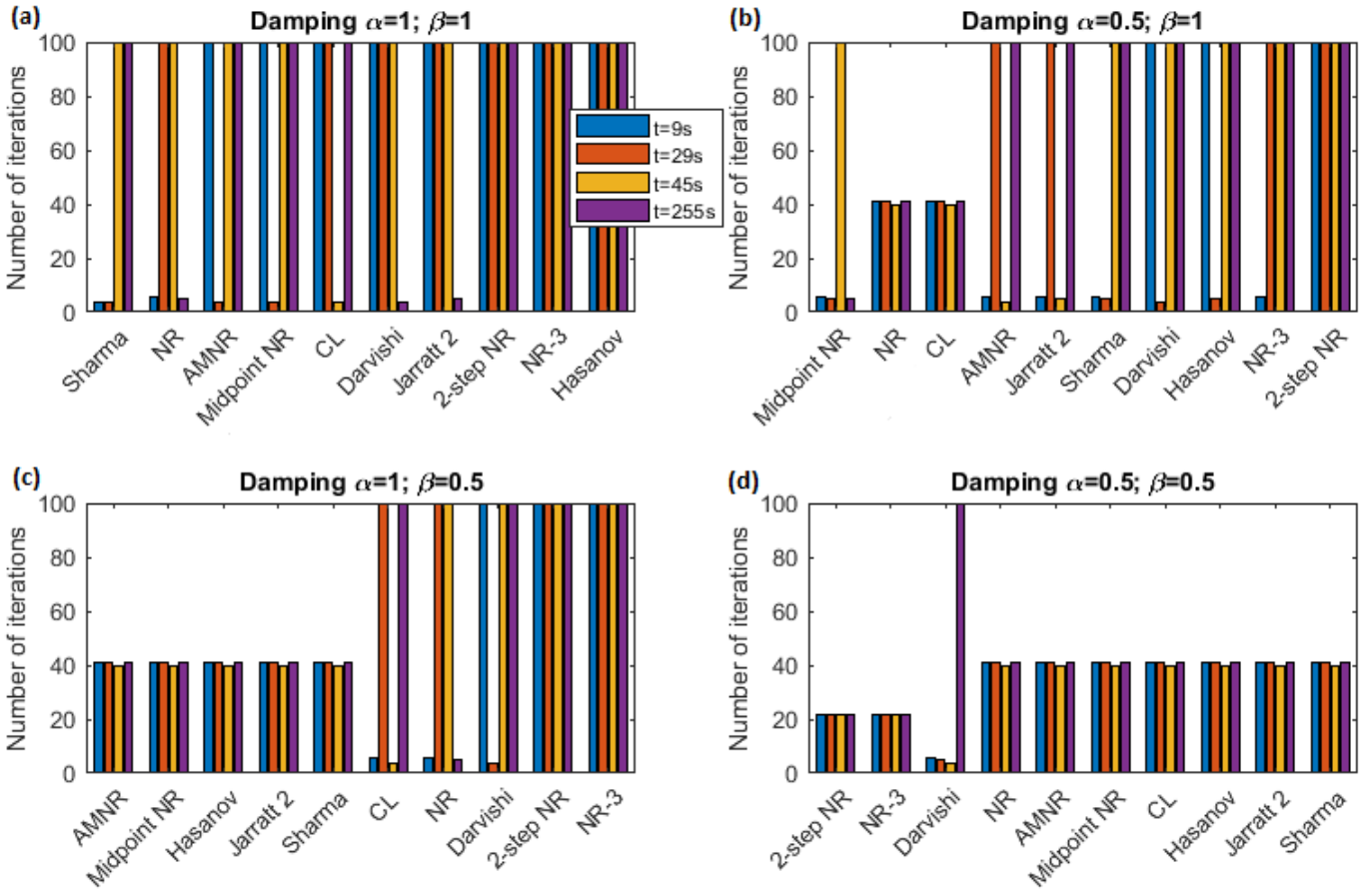

Figure 10- Number of iterations required for the methods to converge at 4 different time instants with (a) no damping; (b) damping on the 1st step; (c) damping on the 2nd step; (d) damping on 1st and 2nd steps.

3.3 Optimal method for the finite-element code

In a finite-element code, robustness is the most important criterion at the level of the integration point. In a simulation with multiple elements, the reduction of the time increment due to a problem of convergence in one element leads to re-doing the calculation for all elements. Multiplying the number of time steps in a finite-element simulation is much more costly than occasionally increasing the number of iterations within the material law for a few elements.

Based on the results of sections 2.3, 2.4, and 3.2.2, an optimal strategy was designed. To benefit from the robustness of the NR method with damping (see Figure 10 (b)) and the efficiency of NR without damping (see Figure 2 and Figure

476 10 (a)), a combined approach was implemented in the finite-element code. By
477 default, the NR method is used within the material law. NR is quite robust and it is
478 the fastest method on average, as shown in Section 2.3 and in Figure 9 (a), which
479 makes it the most adequate default method for solving the equations. However, in
480 the case where NR does not converge in the calculation of a given finite element,
481 the process of solving the nonlinear equations for that element is restarted from the
482 initial value using damped NR algorithm, which is slightly slower but more robust.
483 It is necessary to restart from the initial value and not the value obtained with NR
484 because the latter is likely a local minimum of the residual function which damped
485 NR will also remain stuck at if used as initial value. Although other methods with
486 damping could have been used, NR with damping was chosen as it is expected to
487 be the fastest among the damped methods, given the results obtained with a
488 consistent Jacobian presented in Figure 9 (d).

489 *3.3.1 Application to the single-element model*

490 The 3 different methods (NR, damped NR, mixed NR/damped NR) are compared
491 on the loading case described in Figure 7. The simulation is performed on 100
492 cycles. The results are displayed in Table 4.

493 *Table 4 - Performance of the selected methods for a 100-cycle finite-element simulation*

Method	CPU time	Number of FE time steps
NR	1.515 s	2206
Damped NR	1.812 s	1710
Mixed NR/Damped NR	1.172 s	1710

494 The damped NR method requires fewer time steps than NR. This is because no
495 reduction of the time increment occurs with the damped method thanks to its
496 robustness. However, the high number of iterations required to solve the equations
497 makes it slightly slower. The mixed NR/Damped NR approach performs better

than both these algorithms in terms of speed, as it combines both the robustness of damped NR (no reduction of time increment) and the efficiency of NR (rapid convergence in the material law).

3.3.2 Application to a larger model

The mixed procedure was finally tested on a larger model. The model consists in a slice of a tube from a solar receiver, similar to the test case studied by Logie et al. [29]. This tube contains a heat transfer fluid at a temperature $T_{htf} = 565°C$ and is exposed to a solar flux $q_{sol} = 890$ kW/m² on its front, as shown in Figure 11 (a). The back of the tube is fixed, and both faces of the slice are made to remain parallel during the simulation, to model the fact that the entire tube remains straight. Only half of the tube is modeled due to the symmetry. The half tube is meshed with 500 elements: 1 along the axial length of the tube slice, 5 along the thickness of the tube, and 100 along its circumference.

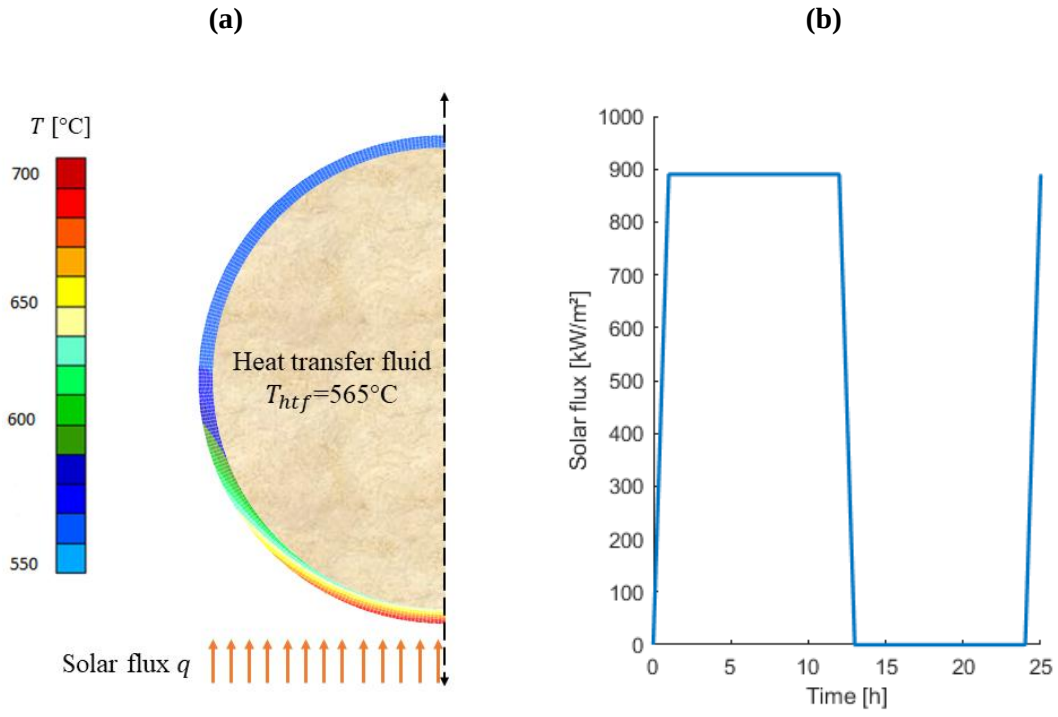

Figure 11 – (a) Temperature field in a tube from a solar receiver; (b) Thermal loading of the tube.

The thermal loading is applied cyclically to simulate the daily heating and cooling of the solar receiver, as shown in Figure 11 (b): at the beginning of the cycle, no

solar flux is applied. Then solar flux increases linearly over a period of 1h until it reaches its value q_{sol}. The solar flux is then maintained for a duration of 11 hours (daytime) before decreasing back to 0 in an hour. Finally, the tube remains with no thermal loading for a duration of 11 hours (nighttime).

Two cycles are simulated using either NR, damped NR, or the mixed method to evaluate the performance of the different methods on a large-scale model.

Table 5 - Performance of the selected methods for a 2-cycle finite-element simulation

Method	**CPU time**		**Number of FE time steps**
	Constitutive law	Total	
NR	217.3s	21m 58s.	697
Damped NR	40s	8m 13s.	469
Mixed NR/Damped NR	28s	8m 6s.	469

Table 5 shows the computation times (time spent in the constitutive law and total for the simulation) and number of steps required for the simulation of 2 cycles of the tube model with the different methods. Damped NR and the mixed method perform much better than NR in this case. However, there is not much difference between damped NR and mixed NR. This is due to the fact that on a large-scale simulation, the time spent for solving the global equations at the finite-element level is more significant than the time spent in the material law of each element. This can be seen from the CPU times: only 28 to 40 seconds are spent in the constitutive law for the simulations with damped NR and the mixed method, whereas the whole simulation lasts around 8 minutes in both cases. Therefore, the reduction of the time increment (robustness) has a greater impact on the large-scale simulation compared to the single-element model seen in section 3.3.1, but the speed of convergence within the material law is not as significant.

4 Conclusions

In this article, we conducted a review of various Newton-type methods for the solution of nonlinear multidimensional equations. The robustness and efficiency of each method was tested on a sample of 10 systems of nonlinear equations, with starting points taken at various distances from the solution. Table 6 sums up the results of the study on nonlinear equations. The Newton-type methods tested are ranked from best to worst on 3 criteria: the number of iterations required to reach the solution, the CPU time to reach the solution, and the robustness, i.e., the capacity to reach a solution. The table also indicates the effect numerical damping can have, considering the most favorable damping coefficients. The following conclusions can be drawn from the study:

- Classic Newton method is the fastest on average on the set of equations tested in this study; compared to other methods, Newton's method is the most robust when the initial point is not too far from the solution.
- Methods that are considered more efficient from a mathematical point of view (fewer iterations required, higher order of convergence) are not the most efficient from a computational point of view. For instance, the Newton method and the Wu method are amongst those that require the highest number of iterations to reach a solution, but among the fastest when considering CPU time. For numerical applications, CPU time is more important than the number of iterations.
- Damping can improve the robustness of some methods (NR, 2-step NR, NR-3, Soleymani) but does not give good results on others (Midpoint-NR, Montazeri, Darvishi). The efficiency of numerical damping also seems to depend on the problem to solve.
- Efficiency and robustness of methods depend on the system to solve and on the distance from the solution of the starting point.

Table 6 - Summary of the results of the study on nonlinear systems of equations: ranking of the Newton-type methods on number of iterations, CPU time, and robustness; improvement of robustness due to numerical damping using the most favorable combination of coefficient for each method.

Method	Number of iterations	CPU time	Robustness	Improvement of robustness from damping (most favorable case)
NR	12	1	5	++
Wu	14	3	8	+
2-step NR	1	2	6	++++
NR-3	8	5	9	++
AMNR	11	8	7	+
Midpoint NR	9	4	4	--
CL	15	14	10	+
Hasanov	3	6	2	+
Haijun	7	11	15	+
Jarratt 1	4	7	3	+
Jarratt 2	13	15	14	+
Sharma	5	12	1	+
Soleymani	10	10	11	+++
Montazeri	6	13	12	/
Darvishi	2	9	13	-

The methods were then compared on an advanced Lemaitre-Chaboche model implemented in a finite-element code. A particularity of this model is that there is no exact analytical expression of Jacobian matrix, therefore an approximation is used. This issue makes it more difficult for Newton-type methods to reach a solution. In the case of the Lemaitre-Chaboche model, damping was shown to significantly improve the robustness of most methods. A mixed method using Newton method with and without damping was finally adopted in the finite element code to improve the overall computation time.

Generally, Newton-type methods of order 2 or higher do not bring much improvement regarding efficiency or robustness compared to Newton method. Other methods which were not tested in this study could still offer further improvement in the convergence of the constitutive law:

- quasi-Newton methods [30] or secant methods [31], which do not require the computation of the Jacobian;
- algorithms using a trust-region approach, such as Levenberg-Marquardt [32] or dogleg method.

Acknowledgements

The authors are grateful to John Cockerill Energy and to the Walloon Region for its financial support. The authors acknowledge the MecaTech Cluster. A.M. Habraken acknowledges the Belgian Fund for Scientific Research FRS-FNRS for its support.

Appendix A: Jacobian matrix for the Lemaitre-Chaboche model

The Jacobian matrix of the system described by equations (7) to (10) can be written as:

$$\left[\frac{\partial \mathcal{R}}{\partial \mathcal{X}}\right] = \begin{bmatrix} \frac{\partial \boldsymbol{R}_{\boldsymbol{\varepsilon}^e}}{\partial \Delta \boldsymbol{\varepsilon}^e} & \frac{\partial \boldsymbol{R}_{\boldsymbol{\varepsilon}^e}}{\partial \Delta r} & \frac{\partial \boldsymbol{R}_{\boldsymbol{\varepsilon}^e}}{\partial \Delta \boldsymbol{\sigma}} & \frac{\partial \boldsymbol{R}_{\boldsymbol{\varepsilon}^e}}{\partial \Delta \boldsymbol{X_1}} & \frac{\partial \boldsymbol{R}_{\boldsymbol{\varepsilon}^e}}{\partial \Delta \boldsymbol{X_2}} & \frac{\partial \boldsymbol{R}_{\boldsymbol{\varepsilon}^e}}{\partial \Delta \boldsymbol{X_3}} \\ \frac{\partial R_r}{\partial \Delta \boldsymbol{\varepsilon}^e} & \frac{\partial R_r}{\partial \Delta r} & \frac{\partial R_r}{\partial \Delta \boldsymbol{\sigma}} & \frac{\partial R_r}{\partial \Delta \boldsymbol{X_1}} & \frac{\partial R_r}{\partial \Delta \boldsymbol{X_2}} & \frac{\partial R_r}{\partial \Delta \boldsymbol{X_3}} \\ \frac{\partial \boldsymbol{R}_{\boldsymbol{\sigma}}}{\partial \Delta \boldsymbol{\varepsilon}^e} & \frac{\partial \boldsymbol{R}_{\boldsymbol{\sigma}}}{\partial \Delta r} & \frac{\partial \boldsymbol{R}_{\boldsymbol{\sigma}}}{\partial \Delta \boldsymbol{\sigma}} & \frac{\partial \boldsymbol{R}_{\boldsymbol{\sigma}}}{\partial \Delta \boldsymbol{X_1}} & \frac{\partial \boldsymbol{R}_{\boldsymbol{\sigma}}}{\partial \Delta \boldsymbol{X_2}} & \frac{\partial \boldsymbol{R}_{\boldsymbol{\sigma}}}{\partial \Delta \boldsymbol{X_3}} \\ \frac{\partial \boldsymbol{R}_{\boldsymbol{X_1}}}{\partial \Delta \boldsymbol{\varepsilon}^e} & \frac{\partial \boldsymbol{R}_{\boldsymbol{X_1}}}{\partial \Delta r} & \frac{\partial \boldsymbol{R}_{\boldsymbol{X_1}}}{\partial \Delta \boldsymbol{\sigma}} & \frac{\partial \boldsymbol{R}_{\boldsymbol{X_1}}}{\partial \Delta \boldsymbol{X_1}} & \frac{\partial \boldsymbol{R}_{\boldsymbol{X_1}}}{\partial \Delta \boldsymbol{X_2}} & \frac{\partial \boldsymbol{R}_{\boldsymbol{X_1}}}{\partial \Delta \boldsymbol{X_3}} \\ \frac{\partial \boldsymbol{R}_{\boldsymbol{X_2}}}{\partial \Delta \boldsymbol{\varepsilon}^e} & \frac{\partial \boldsymbol{R}_{\boldsymbol{X_2}}}{\partial \Delta r} & \frac{\partial \boldsymbol{R}_{\boldsymbol{X_2}}}{\partial \Delta \boldsymbol{\sigma}} & \frac{\partial \boldsymbol{R}_{\boldsymbol{X_2}}}{\partial \Delta \boldsymbol{X_1}} & \frac{\partial \boldsymbol{R}_{\boldsymbol{X_2}}}{\partial \Delta \boldsymbol{X_2}} & \frac{\partial \boldsymbol{R}_{\boldsymbol{X_2}}}{\partial \Delta \boldsymbol{X_3}} \\ \frac{\partial \boldsymbol{R}_{\boldsymbol{X_3}}}{\partial \Delta \boldsymbol{\varepsilon}^e} & \frac{\partial \boldsymbol{R}_{\boldsymbol{X_3}}}{\partial \Delta r} & \frac{\partial \boldsymbol{R}_{\boldsymbol{X_3}}}{\partial \Delta \boldsymbol{\sigma}} & \frac{\partial \boldsymbol{R}_{\boldsymbol{X_3}}}{\partial \Delta \boldsymbol{X_1}} & \frac{\partial \boldsymbol{R}_{\boldsymbol{X_3}}}{\partial \Delta \boldsymbol{X_2}} & \frac{\partial \boldsymbol{R}_{\boldsymbol{X_3}}}{\partial \Delta \boldsymbol{X_3}} \end{bmatrix}$$

Derivatives of $\boldsymbol{R}_{\boldsymbol{\varepsilon}^e}$:

$$\frac{\partial \boldsymbol{R}_{\boldsymbol{\varepsilon}^e}}{\partial \Delta \boldsymbol{\varepsilon}^e} = \boldsymbol{I}_{\mathbf{6x6}} \qquad \frac{\partial \boldsymbol{R}_{\boldsymbol{\varepsilon}^e}}{\partial \Delta r} = \boldsymbol{n} \qquad \frac{\partial \boldsymbol{R}_{\boldsymbol{\varepsilon}^e}}{\partial \Delta \boldsymbol{\sigma}} = \frac{\Delta r}{(1-D)^2}\frac{\partial \boldsymbol{n}^{\boldsymbol{X}}}{\partial \Delta \boldsymbol{s}} \qquad \frac{\partial \boldsymbol{R}_{\boldsymbol{\varepsilon}^e}}{\partial \Delta \boldsymbol{X_i}} = -\frac{\partial \boldsymbol{R}_{\boldsymbol{\varepsilon}^e}}{\partial \Delta \boldsymbol{\sigma}}$$

Derivatives of R_r:

$$\frac{\partial R_r}{\partial \Delta \boldsymbol{\varepsilon}^e} = \mathbf{0}_{1\mathrm{x}6} \qquad \frac{\partial R_r}{\partial \Delta r} = bQ * \exp(-br) \qquad \frac{\partial R_r}{\partial \Delta \boldsymbol{\sigma}} = \{\boldsymbol{n}\}^{\boldsymbol{T}} \qquad \frac{\partial R_r}{\partial \Delta \boldsymbol{X_i}} = \{\boldsymbol{n}^{\boldsymbol{X}}\}^T$$

Derivatives of $\boldsymbol{R}_{\boldsymbol{\sigma}}$:

$\frac{\partial \boldsymbol{R_\sigma}}{\partial \Delta \boldsymbol{\varepsilon^e}} = \boldsymbol{I_{6x6}}$	$\frac{\partial \boldsymbol{R_\sigma}}{\partial \Delta r} = \boldsymbol{0}_{6x1}$	$\frac{\partial \boldsymbol{R_\sigma}}{\partial \Delta \boldsymbol{\sigma}} = -\frac{1+\nu}{E}\left(\frac{\boldsymbol{H_\sigma^+}}{1-D} + \frac{\boldsymbol{H_\sigma^-}}{1-hD}\right) + \frac{\nu}{E}\left(\frac{H(tr(\sigma))}{1-D} + \frac{H(-tr(\sigma))}{1-hD}\right)\boldsymbol{I_{6x1}} \otimes \boldsymbol{I_{6x1}}$	$\frac{\partial \boldsymbol{R_\sigma}}{\partial \Delta \boldsymbol{X_i}} = \boldsymbol{0}_{6x6}$

592 Where $H(x)$ is the Heaviside step function, $\otimes$ is the tensorial product, $\boldsymbol{H_\sigma^+}$ and $\boldsymbol{H_\sigma^-}$

593 are obtained by applying the Heaviside step function to each component of $\langle \boldsymbol{\sigma} \rangle^+$

594 and $\langle \boldsymbol{\sigma} \rangle^-$ respectively. $\boldsymbol{I_{6x6}}$ is the 6-by-6 second-order identity tensor and $\boldsymbol{0}_{mxn}$ is

595 an m-by-n zero tensor.

596 If $h = 1$, the derivative $\frac{\partial \boldsymbol{R_\sigma}}{\partial \Delta \boldsymbol{\sigma}}$ becomes:

597 $$\frac{\partial \boldsymbol{R_\sigma}}{\partial \Delta \boldsymbol{\sigma}} = -\frac{1+\nu}{E}\frac{1}{1-D}\boldsymbol{I_{6x6}} + \frac{\nu}{E}\frac{1}{1-D}\boldsymbol{I_{6x1}} \otimes \boldsymbol{I_{6x1}}$$

598 Derivatives of $\boldsymbol{R_{X_i}}$:

$\frac{\partial \boldsymbol{R_{X_i}}}{\partial \Delta \boldsymbol{\varepsilon^e}} = \boldsymbol{0_{6x6}}$	$\frac{\partial \boldsymbol{R_{X_i}}}{\partial \Delta r} = -\frac{2}{3}C_i \boldsymbol{n^X} + \gamma_i(\boldsymbol{X_i} - \boldsymbol{Y_i})$	$\frac{\partial \boldsymbol{R_{X_i}}}{\partial \Delta \boldsymbol{\sigma}} = -\frac{2}{3}C_i \Delta r \frac{\partial \boldsymbol{n^X}}{\partial \Delta \boldsymbol{\sigma}}$	$\frac{\partial \boldsymbol{R_{X_i}}}{\partial \Delta \boldsymbol{X_j}} = \boldsymbol{0}_{6x6}$ for $j \neq i$

$$\frac{\partial \boldsymbol{R_{X_i}}}{\partial \Delta \boldsymbol{X_i}} = \boldsymbol{I_{6x6}} - \frac{2}{3}C_i \Delta r \frac{\partial \boldsymbol{n^X}}{\partial \Delta \boldsymbol{X_i}} + \gamma_i \Delta r \left(\boldsymbol{I_{6x6}} - \frac{\partial \boldsymbol{Y_i}}{\partial \Delta \boldsymbol{X_i}}\right) + b_i \Delta t \left[\frac{3}{2}(r_i - 1)\|\boldsymbol{X_i}\|^{r_i - 3}\boldsymbol{X_i} \otimes \boldsymbol{X_i} + \|\boldsymbol{X_i}\|^{r_i - 1}\boldsymbol{I_{6x6}}\right] - \frac{1}{C_i}\frac{\partial C_i}{\partial T}\Delta T \boldsymbol{I_{6x6}}$$

599 Where:

600 $$\frac{\partial \boldsymbol{n^X}}{\partial \Delta \boldsymbol{\sigma}} = \frac{1}{1-D}\frac{\partial \boldsymbol{n^X}}{\partial \Delta \boldsymbol{s}} = \frac{1}{1-D}\frac{1}{\|\boldsymbol{s} - \boldsymbol{X}\|}\left[\frac{3}{2}\boldsymbol{I_{6x6}} - \frac{1}{2}\boldsymbol{I_{6x1}} \otimes \boldsymbol{I_{6x1}} - \boldsymbol{n^X} \otimes \boldsymbol{n^X}\right]$$

601 $$\frac{\partial \boldsymbol{n^X}}{\partial \Delta \boldsymbol{X_i}} = \frac{1}{\|\boldsymbol{s} - \boldsymbol{X}\|}\left[\frac{3}{2}\boldsymbol{I_{6x6}} - \boldsymbol{n^X} \otimes \boldsymbol{n^X}\right]$$

602 $$\frac{\partial \boldsymbol{Y_i}}{\partial \Delta \boldsymbol{X_i}} = -\frac{3}{2}\alpha_{b,i}\frac{\|\boldsymbol{X_i}\|^{r_i - 3}}{1 + \alpha_{b,i}\|\boldsymbol{X_i}\|^{r_i}}\left[\Delta t Y_{st,i}\left(\|\boldsymbol{X_i}\|^2 \boldsymbol{I_{6x6}} + \frac{3}{2}(r_i - 1)\boldsymbol{X_i} \otimes \boldsymbol{X_i}\right)\right.$$

603 $$\left. - \frac{r_i \|\boldsymbol{X_i}\|}{1 + \alpha_{b,i}\|\boldsymbol{X_i}\|^{r_i}}\boldsymbol{X_i} \otimes (\boldsymbol{Y_{i,n}} - \frac{3}{2}\Delta t \alpha_{b,i} Y_{st,i}\|\boldsymbol{X_i}\|^{r_i - 1}\boldsymbol{X_i})\right]$$

References

[1] S. Yuan, L. Duchêne, C. Keller, E. Hug, and A. M. Habraken, “Tunable surface boundary conditions in strain gradient crystal plasticity model,” Mech. Mater., vol. 145, no. February, p. 103393, 2020, doi: 10.1016/j.mechmat.2020.103393.

[2] J. L. Chaboche, “A review of some plasticity and viscoplasticity constitutive theories,” Int. J. Plast., vol. 24, no. 10, pp. 1642–1693, 2008, doi: 10.1016/j.ijplas.2008.03.009.

[3] R. Ahmed and T. Hassan, “Constitutive modeling for thermo-mechanical low-cycle fatigue-creep stress–strain responses of Haynes 230,” Int. J. Solids Struct., vol. 126–127, pp. 122–139, 2017, doi: 10.1016/j.ijsolstr.2017.07.031.

[4] J. L. Chaboche and G. Rousselier, “On the plastic and viscoplastic constitutive equations-part I: Rules developed with internal variable concept,” J. Press. Vessel Technol. Trans. ASME, vol. 105, no. 2, pp. 153–158, 1983, doi: 10.1115/1.3264257.

[5] J. Lemaitre and R. Desmorat, Engineering Damage Mechanics. Berlin/Heidelberg: Springer-Verlag, 2005.

[6] J. H. Holland, Adaptation in Natural and Artificial Systems. The MIT Press, 1992.

[7] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in Proceedings of ICNN’95 - International Conference on Neural Networks, 1995, vol. 4, pp. 1942–1948, doi: 10.1109/ICNN.1995.488968.

[8] J. M. Gutiérrez and M. A. Hernández, “A family of chebyshev-halley type methods in banach spaces,” Bull. Aust. Math. Soc., vol. 55, no. 1, pp. 113–130, 1997, doi: 10.1017/s0004972700030586.

[9] X. Wu, “Note on the improvement of Newton’s method for system of

nonlinear equations," Appl. Math. Comput., vol. 189, no. 2, pp. 1476–1479, 2007, doi: 10.1016/j.amc.2006.12.035.

[10] M. T. Darvishi and A. Barati, "Super cubic iterative methods to solve systems of nonlinear equations," Appl. Math. Comput., vol. 188, no. 2, pp. 1678–1685, 2007, doi: 10.1016/j.amc.2006.11.022.

[11] J. M. Ortega and W. C. Rheinboldt, "General Iterative Methods," in Iterative Solution of Nonlinear Equations in Several Variables, 1970, pp. 181–239.

[12] D. K. R. Babajee and M. Z. Dauhoo, "An analysis of the properties of the variants of Newton's method with third order convergence," Appl. Math. Comput., vol. 183, no. 1, pp. 659–684, 2006, doi: 10.1016/j.amc.2006.05.116.

[13] M. Frontini and E. Sormani, "Some variant of Newton's method with third-order convergence," Appl. Math. Comput., vol. 140, no. 2–3, pp. 419–426, 2003, doi: 10.1016/S0096-3003(02)00238-2.

[14] A. . Özban, "Some new variants of Newton's method," Appl. Math. Lett., vol. 17, no. 6, pp. 677–682, Jun. 2004, doi: 10.1016/S0893-9659(04)90104-8.

[15] M. Frontini and E. Sormani, "Third-order methods from quadrature formulae for solving systems of nonlinear equations," Appl. Math. Comput., vol. 149, no. 3, pp. 771–782, 2004, doi: 10.1016/S0096-3003(03)00178-4.

[16] D. K. R. Babajee, M. Z. Dauhoo, M. T. Darvishi, A. Karami, and A. Barati, "Analysis of two Chebyshev-like third order methods free from second derivatives for solving systems of nonlinear equations," J. Comput. Appl. Math., vol. 233, no. 8, pp. 2002–2012, 2010, doi: 10.1016/j.cam.2009.09.035.

[17] V. I. Hasanov, I. G. Ivanov, and G. Nedjibov, "A New Modification of

Newton's Method," in XXVII Summer School Applications of Mathematics in Engineering and Economics, 2002, pp. 278–286, doi: 10.13140/2.1.2805.0561.

[18] W. Haijun, "New third-order method for solving systems of nonlinear equations," Numer. Algorithms, vol. 50, no. 3, pp. 271–282, 2009, doi: 10.1007/s11075-008-9227-2.

[19] P. Jarratt, "Some Fourth Order Multipoint Iterative Methods for Solving Equations," Math. Comput., vol. 20, no. 95, p. 434, 1966, doi: 10.2307/2003602.

[20] J. R. Sharma, R. K. Guha, and R. Sharma, "An efficient fourth order weighted-Newton method for systems of nonlinear equations," Numer. Algorithms, vol. 62, no. 2, pp. 307–323, 2013, doi: 10.1007/s11075-012-9585-7.

[21] F. Soleymani, "Regarding the accuracy of optimal eighth-order methods," Math. Comput. Model., vol. 53, no. 5–6, pp. 1351–1357, 2011, doi: 10.1016/j.mcm.2010.12.032.

[22] H. Montazeri, F. Soleymani, S. Shateyi, and S. S. Motsa, "On a new method for computing the numerical solution of systems of nonlinear equations," J. Appl. Math., vol. 2012, no. May 2014, 2012, doi: 10.1155/2012/751975.

[23] M. T. Darvishi and A. Barati, "A fourth-order method from quadrature formulae to solve systems of nonlinear equations," Appl. Math. Comput., vol. 188, no. 1, pp. 257–261, 2007, doi: 10.1016/j.amc.2006.09.115.

[24] M. T. Heath, *Scientific Computing*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2018, doi: 10.1137/1.9781611975581

[25] H. Morch, L. Duchêne, R. Harzallah, V. Tuninetti, and A. M. Habraken, "Efficient temperature dependence of parameters for thermo-mechanical finite element modeling of alloy 230," Eur. J. Mech. A/Solids, vol. 85, no.

September 2020, 2021, doi: 10.1016/j.euromechsol.2020.104116.

[26] University of Liege, “Lagamine FE code.” http://www.lagamine.uliege.be/dokuwiki/doku.php/.

[27] C. O. Frederick and P. J. Armstrong, “A mathematical representation of the multiaxial Bauschinger effect,” Mater. High Temp., vol. 24, no. 1, pp. 1–26, 2007, doi: 10.3184/096034007X207589.

[28] M. Yaguchi, M. Yamamoto, and T. Ogata, “A viscoplastic constitutive model for nickel-base superalloy, part 1: Kinematic hardening rule of anisotropic dynamic recovery,” Int. J. Plast., vol. 18, no. 8, pp. 1083–1109, 2002, doi: 10.1016/S0749-6419(01)00029-8.

[29] W. R. Logie, J. D. Pye, and J. Coventry, “Thermoelastic stress in concentrating solar receiver tubes: A retrospect on stress analysis methodology, and comparison of salt and sodium,” Sol. Energy, vol. 160, no. November 2017, pp. 368–379, 2018, doi: 10.1016/j.solener.2017.12.003.

[30] R. Fletcher, “Newton-like Methods,” in Practical Methods of Optimization, Wiley, Ed. 1987.

[31] J. E. Dennis and R. B. Schnabel, “Secant Methods for Systems of Nonlinear Equations,” in Numerical Methods for Unconstrained Optimization and Nonlinear Equations, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1983, pp. 168–193.

[32] K. Levenberg and F. Arsenal, “A Method for the Solution of Certain Non-Linear Problems in Least Squares,” Q. Appl. Math., vol. 1, no. 278, pp. 536–538, 1943.