Levenberg-Marquardt vs Powell's dogleg method for Gurson-Tvergaard-Needleman plasticity model

A Shterenlikht*, NA Alexander

Faculty of Engineering, University of Bristol, Bristol BS8 1TR, UK

Abstract

The GTN continuous damage model is very popular in academia and industry for structural integrity assessment and ductile fracture simulation. Following Aravas' influential 1987 paper, Newton's method has been used widely to solve the GTN equations. However, if the starting point is far from the solution, then Newton's method can fail to converge. Hybrid methods are preferred in such cases. In this work we translate the GTN equations into a non-linear minimization problem and then apply the Levenberg-Marquardt and Powell's 'dogleg' hybrid methods to solve it. **The methods are tested for accuracy and robustness on two simple single finite element models and two 3D models with complex deformation paths. In total nearly 137,000 different GTN problems were solved.** We show that the Levenberg-Marquardt method is more robust than Powell's method. Our results are verified against the Abaqus' own solver. The superior accuracy of the Levenberg-Marquardt method allows for larger time increments in implicit time integration schemes.

Keywords:
GTN model, Levenberg-Marquardt method, Powell dogleg method, Slatec

1. Introduction

Gurson-Tvergaard-Needleman (GTN) [1] is a popular pressure dependent (or porous metal) plasticity model. The GTN model is widely used in academia and industry. Typical applications include structural integrity assessments of nuclear reactor pressure vessels [2] and welded joints [3]; optimization of impact resistance of marine steels [4]; forming of Aluminium alloys in automotive industry [5] and steel forming [6], etc.

The GTN flow potential depends on the first and the second stress invariants, p and q, and a set of state variables. Mathematically the GTN model results in a system of non-linear PDEs, or in finite differences - a set of non-linear algebraic equations.

Aravas [7] successfully used the Newton's method for the solution of the GTN equations. Newton's method is still used successfully for this problem, see e.g. [8, 9] for recent examples.

However, Newton's method may not converge at all, or converge to a local minimum, if the starting point is far from the solution. This is a well known weakness of the Newton's method [10, 11, 12, 13]. Hence, great care should be taken when selecting a starting point for the Newton's method. In practice, if the deformation path is complex, there might be no good guess for a starting point, so the Newton's method would fail.

To overcome this problem Beardsmore et al [14] successfully applied Powell's 'dogleg' (DL) method [15] to solving the GTN equations for an implicit time integration case. The DL method is a combination of the Newton's

*Corresponding author
Email address: `mexas@bris.ac.uk` (A Shterenlikht)

and the steepest descent (SD) methods. When the approximated solution is far from the global minimum, the SD step is taken. The Newton's step is taken when the approximated solution appears to be close enough to the global minimum. Simple single finite element problems and an axisymmetric tensile problem were analysed in [14] with the results validated against the Abaqus [16].

However, in this work we show an example of a rod under tension and shear, in which the DL method fails to solve the GTN equations.

The main purpose of this paper is to find a solution method that is accurate and robust enough so that it can be used to solve large numbers of GTN problems, with no modification of the parameters of the **solution method, such as e.g. the starting point, or the scaling**.

We therefore suggest looking at the solution of the GTN equations as a non-linear optimization problem. We apply the Levenberg-Marquardt (LM) minimization algorithm [17], which is recommended for general non-linear least squares problems in optimization literature, see e.g. [18, p.228,233], and demonstrate that it outperforms the DL method in all cases under analysis.

The plan for the paper is as follows. Section 2 gives a summary of the GTN equations and discusses factors complicating their numerical minimization. Section 3 gives a summary of the LM and the DL computational routines used. Scaling is discussed in section 4. Section 5 shows convergence results for the LM and the DL methods on four test problems and compares them with the Abaqus implementation. We show that for simple deformation paths, the LM method converges quicker than the DL. **We then test the robustness of the DL and the LM methods on problems where different material points undergo vastly different deformation histories.** For the rod under tension and shear model the LM method can be used for the whole of the deformation path, while the DL method fails half way. For a 3D tensile model, the LM method seems to be more robust than the Abaqus' own GTN solver. This means that substantially larger time steps can be taken with implicit time integration scheme when using the LM method, compared to when using the Abaqus' GTN solver. Finally, in section 6 we discuss potential pitfalls of the DL method when solving the GTN equations.

2. The GTN equations

2.1. The GTN model

The mechanical justification of the GTN model is given in detail elsewhere [1, 7, 19, 8]. Here we are only concerned with the numerical implementation, hence we only present the resulting set of GTN equations.

The GTN model is an extension of the classical von Mises plasticity. The von Mises flow potential depends only on the second stress invariant, equivalent stress, q, whereas the GTN potential adds dependence on pressure, p, as well. The GTN model has two state variables: f - the void volume fraction and ε_q^m - the equivalent plastic strain in the fully dense matrix. The GTN potential can be written as:

$$\left(\frac{q}{\sigma_0}\right)^2 + 2q_1 f \cosh\left(\frac{3q_2 p}{2\sigma_0}\right) - (1 + q_3 f^2) = 0 \qquad (1)$$

where q_1, q_2, q_3 are fitted model parameters introduced to help the model agree better with experiments. Following [7] we use $q_1 = 1.5, q_2 = 1.0, q_3 = q_1^2$.

$$\sigma_0 = \sigma_0(\varepsilon_q^m) \qquad (2)$$

is the flow stress in the fully dense matrix. Typically, this is either a piece-wise linear function based on raw experimental data, see Fig. 1, or some fitted smooth function, e.g. Ramberg-Osgood or a power law.

p and q are calculated using the classical radial return algorithm [7]:

$$p = p^e + K\Delta\varepsilon_p \qquad (3)$$

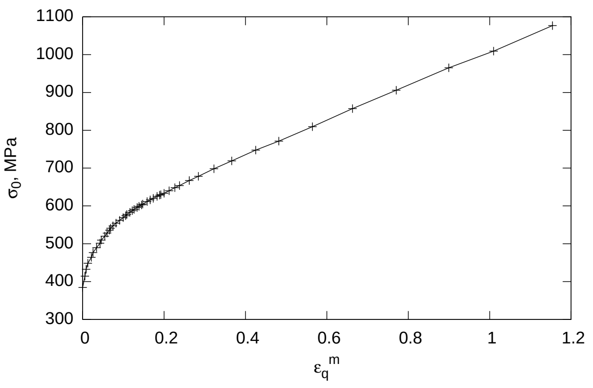


Figure 1: A piece-wise linear function $\sigma_0(\varepsilon_q^m)$ used in this work.

$$q = q^e - 3G\Delta\varepsilon_q \qquad (4)$$

where p^e and q^e are the elastic predictors, calculated on the assumption that the whole of strain increment is elastic, $\Delta\varepsilon_p$ is the volumetric plastic strain increment, $\Delta\varepsilon_q$ is the equivalent plastic strain increment and K and G are the bulk and the shear moduli respectively. In this work we use the Young's modulus of 200 GPa and the Poisson's ratio of 0.3.

Following [20] the GTN model includes the void nucleation mechanism following a normal distribution, which is active only for $p \leq 0$:

$$A = \begin{cases} \frac{f_N}{s_N\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{\varepsilon_q^m - \varepsilon_N}{s_N}\right)^2\right] & p \leq 0 \\ 0 & p > 0 \end{cases} \qquad (5)$$

where f_N, s_N, ε_N are parameters of the normal distribution. Following [7] we use $f_N = 0.04, s_N = 0.1, \varepsilon_N = 0.3$.

The GTN model is completed by the condition for the associated plastic flow (function g_1 below) and by two rules for updating the state variables, ε_q^m and f (functions g_3 and g_4 below).

Thus the GTN model is a set of 4 PDEs, which can be written in the finite differences as:

$$g_1 = \Delta\varepsilon_p \frac{2q}{\sigma_0} + 3q_1q_2f\Delta\varepsilon_q \sinh\left(\frac{3q_2p}{2\sigma_0}\right) = 0 \qquad (6)$$

$$g_2 = \left(\frac{q}{\sigma_0}\right)^2 + 2q_1f\cosh\left(\frac{3q_2p}{2\sigma_0}\right) - (1 + q_3f^2) = 0 \qquad (7)$$

$$g_3 = \Delta\varepsilon_q^m(1-f)\sigma_0 + p\Delta\varepsilon_p - q\Delta\varepsilon_q = 0 \qquad (8)$$

$$g_4 = \Delta f - (1-f)\Delta\varepsilon_p - A\Delta\varepsilon_q^m = 0 \qquad (9)$$

At the start of deformation $\varepsilon_q^m(t = 0) = 0$ and $f(t = 0) = 0.01$ - the initial void volume fraction.

Eqns. (3) and (4) clearly show the key role of the time step for the success of the Newton's method. If the time increment is small, the elastic predictors, p^e, q^e are very close to the new flow surface, and the required plastic corrections, $\Delta\varepsilon_p, \Delta\varepsilon_q$ are very small. In such cases a starting point of (0,0) for $\Delta\varepsilon_p, \Delta\varepsilon_q$ is close enough to the solution and Newton's method works well. This typically is the case in explicit time integration scheme [16, 21, 22].

On the other hand, an implicit time integration schemes usually use a significantly larger time increment [16]. This results in the elastic predictor, p^e, q^e, going significantly beyond the new flow surface, requiring large plastic correction. In such cases $\Delta\varepsilon_p, \Delta\varepsilon_q$ could be so far from (0,0), that the Newton's method might fail to converge [10, 12, 13]. In such cases a hybrid solution method, such as the DL or the LM seems to be most suitable.

If a finite element model and the deformation path are sufficiently simple, then it might be possible to choose a better starting search point for $(\Delta\varepsilon_p, \Delta\varepsilon_q, \Delta\varepsilon_q^m, \Delta f)$ than (0,0,0,0). For example if the time increment is fixed and the deformation is monotonic, then solution from the previous time increment might be a better choice for a starting point. However, in general this is not possible. If the time increment changes significantly from one increment to another, or if the deformation changes rapidly from e.g. tension to compression, then the solutions of the GTN problem for two consecutive time increments might be very different. In such cases using a solution from the previous time increment as the starting point for the current time increment is counter productive. Indeed, for the problems analysed in this paper we obtain better convergence results when using (0,0,0,0) as the starting point.

2.2. The least squares GTN problem

Matrix notation is helpful here: $\boldsymbol{x} \equiv (x_1, x_2, x_3, x_4)^T \equiv (\Delta\varepsilon_p, \Delta\varepsilon_q, \Delta\varepsilon_q^m, \Delta f)^T$, $\boldsymbol{g} \equiv (g_1, g_2, g_3, g_4)^T$, $\boldsymbol{J} \equiv J_{ij} \equiv \partial g_i / \partial x_j$, $||.||$ is L2 norm. If we introduce the objective function

$$F(\boldsymbol{x}) = ||\boldsymbol{g}|| \tag{10}$$

then the problem of solving the system (6)-(9) can be rewritten in the minimization framework as

$$\min_x F(\boldsymbol{x}) \tag{11}$$

and if the system (6)-(9) has a solution at all, then the only acceptable solution of (11), $\boldsymbol{x}^*$, is such that $F(\boldsymbol{x}^*) = 0$ (within machine tolerance, more on this later).

Eqns. (2)-(9) possess several features, which present at least six potential problems for most numerical solution algorithms.

F has very narrow valleys, Fig. 2. This means F is a lot more sensitive to some variables than to others, which presents problems for all gradient based methods, like SD, the DL and the LM.

The final drop to the the global minimum is extremely steep and very localized, Fig. 3. The iterative process must be robust enough in the initial stages, to descend towards this very localized region.

$g_2 \gg g_1, g_3, g_4$ for large time increments. This fact complicates choosing a good iteration step. There is a strong suggestion in optimization literature to scale the functions in such cases [12, 11, 23]. However, the complexity of the GTN functions does not allow to choose a reasonably simple scaling strategy.

σ_0 is discontinuous at 0, which might create problems during the initial stages of the deformation, where $\Delta\varepsilon_q^m < 0$ will result in $\varepsilon_q^m < 0$, for which σ_0 is undefined. In addition, if a piece-wise form of σ_0 is chosen, its derivative will be discontinuous in multiple points.

Finally, function A is discontinuous, Eqn. (5). Accordingly $A' \equiv \mathrm{d}A/\mathrm{d}\varepsilon_q^m$ is discontinuous too. However, such discontinuity only occurs if p oscillates about 0 in a single time increment, which is rare in practice. Together this and the previous point result in discontinuity of the Jacobian, specifically components $J_{i3}, i = 1, \ldots, 4$.

The non-linear minimization framework naturally helps overcoming another complication of the Aravas' original approach [7]. He treated x_1, x_2 as the two primary unknowns and solved (6)-(7) as a system of 2 equations, where x_3, x_4 were fixed parameters [7]. Following that he separately solved (8)-(9), again as a system of 2 equations, to update x_1, x_2. This two stage process was repeated until convergence. As shown in [14], this approach has several drawbacks. Up to 400 iterations of sub-problem (8)-(9) might be required, until convergence of (6)-(7) is achieved. The Jacobian for sub-problem (6)-(7) is complicated, involving a matrix inverse on each iteration, because all implicit dependencies from (8)-(9) must be taken into account. Finally, for some trial x_1, x_2, sub-problem (8)-(9) might not have a solution.

In [14] Eqns. (6)-(9) were solved as four simultaneous equations, which significantly improved the robustness of the solution algorithm. In this work we solve the GTN equations as a non-linear minimization problem, as defined by (11).

3. Implementation

The Abaqus' [16] own solver and it's implementation of the GTN model were taken as a reference, against which we validated our work. Numerical integration of the structural equilibrium equations (the outer loop) in the Abaqus was done with the Newton's method [16]. As far as we can tell from the Abaqus Theory manual, integration of material

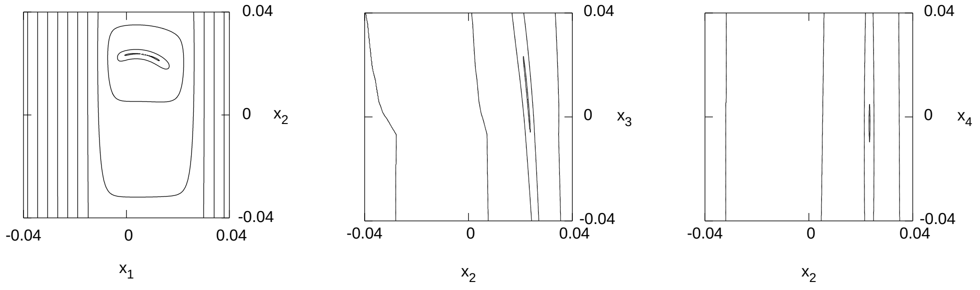


Figure 2: **Typical** contour plots of F around $\boldsymbol{x} = 0$. The contour lines are spaced by a factor of 10.

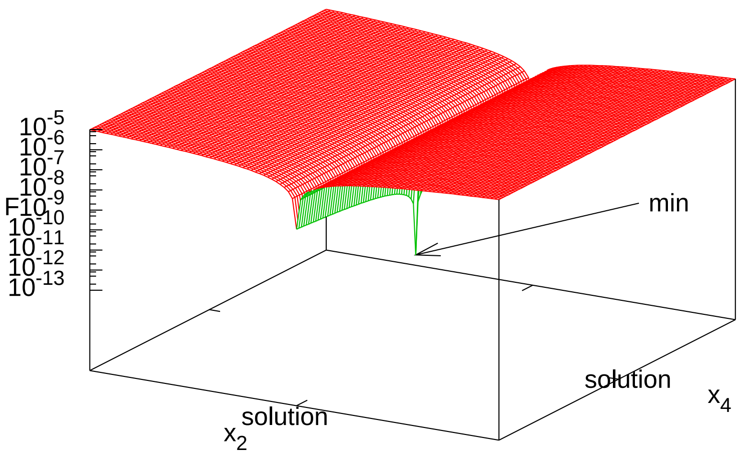


Figure 3: **A typical** 3D section of F around a solution. The bounds are $\boldsymbol{x}^* \pm 10^{-7}$. Note the extremely steep descent to the minimum.

equations (the inner loop) by Abaqus' own solver is also done using Newton's method [16]. However, there is no indication in the manuals as to the choice of the staring point.

In all cases we used automatic time incrementation scheme for the outer loop, based on the maximum force residuals [16].

We chose freely available Fortran routines of the Slatec library (http://netlib.org/slatec): DNLS1 routine for the Levenberg-Marquardt method [17, 24] and DNSQ routine for the Powell's dogleg method [15, 25]. There are two reasons for this choice: (a) Fortran routines were required because our GTN code is written as a user material subroutine for the Abaqus code [16]; (b) we needed to have access to the source code to examine the exact implementation of the algorithms.

Both the LM and the DL routines use the same convergence criterion:

$$\Delta \leq \epsilon_x ||\boldsymbol{D}\boldsymbol{x}|| \tag{12}$$

where Δ is the step bound (trust region size), $\boldsymbol{D}$ is the diagonal matrix of scaling factors and ϵ_x is the user specified maximum relative error in $\boldsymbol{x}$ at the solution.

Initially, $\Delta = k||\boldsymbol{D}x^{(0)}||$ if $||\boldsymbol{D}x^{(0)}|| \neq 0$ and otherwise $\Delta = k$, where k is a user specified factor, and $\boldsymbol{x}^{(0)}$ is the initial guess. As we mentioned before, for lack of a better guess, $\boldsymbol{x}^{(0)} = 0$ was used in this work. Therefore $\Delta = k$. As we show below, the DL algorithm is very sensitive to k.

On the following iterations, Δ is updated based on the success of the previous iteration, as measured by the gain function [17, 15].

Unless specified otherwise, we used $\epsilon_x = \sqrt{\epsilon}$, where ϵ is the machine epsilon, the largest relative spacing. We use double precision (IEEE 64 bit), so $\epsilon \approx 2.22 \times 10^{-16}$, $\sqrt{\epsilon} \approx 1.49 \times 10^{-8}$.

We note briefly that both the LM and the DL routines use one extra convergence criteria each. In the LM routine **it is** the criterion based on the predicted and actual reduction in F being smaller than a prescribed tolerance. In the DL routine **it is** the criterion that the exact zero is found, $F = 0$. However, in none of our numerical experiments were either of these two extra criteria satisfied, so we don't discuss them further.

4. Scaling

It is well known that good scaling of $\boldsymbol{x}$ is vital for the success of an iterative solver [12, 11, 23, 17, 15, 10].

4.1. Auto (internal) scaling

Both the LM and the DL Slatec routines implement adaptive auto scaling [17], based on the norms of the columns of the Jacobian. Initial Jacobian is used on the first iteration. On the following iterations, either the current Jacobian is used or the previous scaling factor, whichever is greater:

$$D_j^{(0)} = ||J_{ij}^{(0)}|| \tag{13}$$

$$D_j^{(k)} = \max\left(D_j^{(k-1)}, ||J_{ij}^{(k)}||\right) \tag{14}$$

However, our results (section 5 below) show that such auto scaling doesn't work well for the GTN problem in many cases, because of the fast changing Jacobian.

4.2. Manual scaling

Preliminary results showed, see Fig. 4, that x_1, x_4 are in the order of 10^{-4} to 10^{-2}, and x_2, x_3 are in the order of 10^{-2} to 10^{-1}. **The scaling was chosen so that the scaled variables, $\boldsymbol{Dx}$, are of the same order. This simple strategy works well for many practical problems [26].** Hence we used $\boldsymbol{D} = \mathrm{diag}(10^4, 10^2, 10^2, 10^4)$ as a default scaling matrix.

5. Results

5.1. Two single finite element models

The behaviours of the LM and the DL routines were first validated on simple single finite element models, one loaded under uniaxial tension, and another loaded under tension+shear, which are. shown schematically in Fig. 5. **The elements are cubes initially, with side length, $L = 1$mm. In the uniaxial tension case the applied displacement is $u = 2L$. In the tension+shear case the applied displacement is $u = L$. Note that in the tension+shear model the two right bottom nodes are constrained only in direction 3 and are free to move in directions 1 and 2. The prescribed motion of the top nodes in direction 2 induces motion of the bottom right nodes to the right too, which creates tensile stress along direction 2.** Reduced integration element was used in both cases (single integration point, C3D8R [16]).

The models were analysed with both the DL and the LM routines with these parameters: $k = 100$, $\boldsymbol{D} = \mathrm{diag}(10^4, 10^2, 10$ $\epsilon_x = 10^{-5} \times \sqrt{\epsilon}$. Note the very small ϵ_x value. For these simple models, i.e. the numerical routines must deal with only a single deformation path, it was possible to find the solution of the GTN model with very high accuracy.

The LM, the DL and the Abaqus reference implementation agree perfectly in both cases, as shown in Figs. 6,7. However, in both cases the DL routine progressed in smaller time increments than the LM, see Tab. 1.

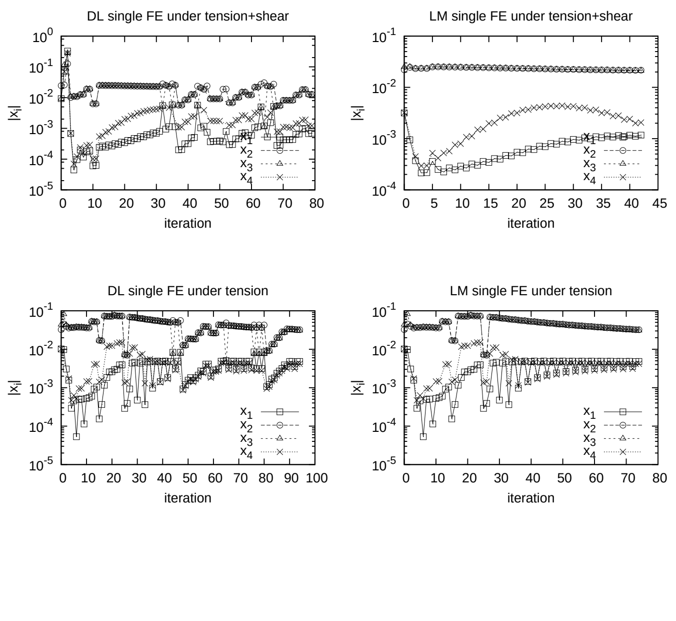


Figure 4: Magnitudes of solution values, $|x_i|$, for two single finite element problems (Fig. 5) with the DL and the LM routines. The abscissa is the finite element time increment number. Note that $|x_2|, |x_3| \approx 10^{-1}$ to 10^{-2}, while $|x_1|, |x_4| \approx 10^{-2}$ to 10^{-4}.

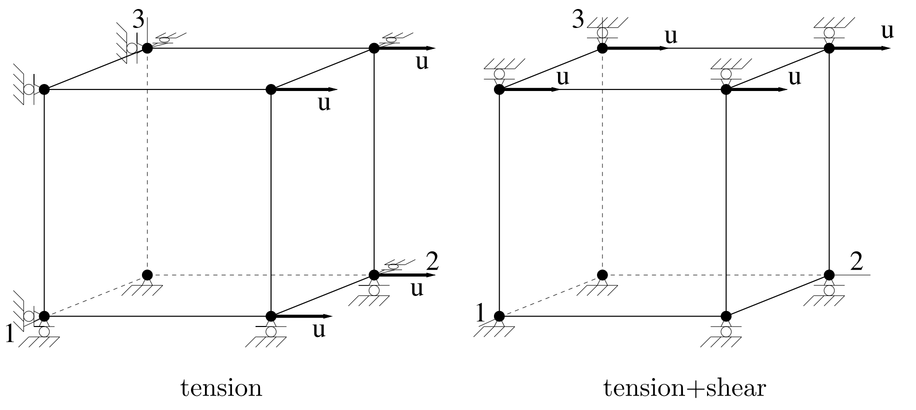


Figure 5: Schematic of the two single finite element models, the first is loaded only in tension, and the second is loaded in tension and shear. Numbers in brackets show constrained degrees of freedom at the nodes. u is prescribed displacement.

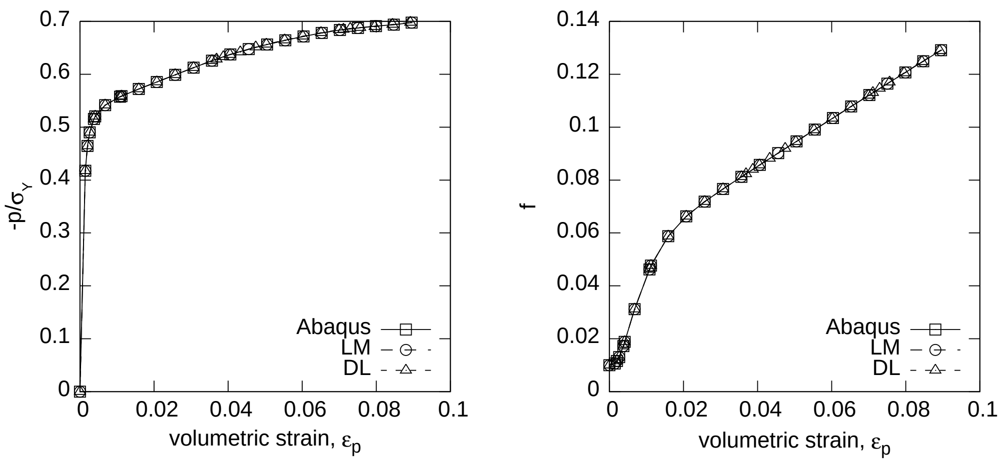


Figure 6: Pressure, p, and void volume fraction, f, plotted against the volumetric strain ε_p for a single FE under uniaxial tension. Note that the Abaqus' reference implementation and the LM routine took 24 increments, while the DL routine took 29 increments.

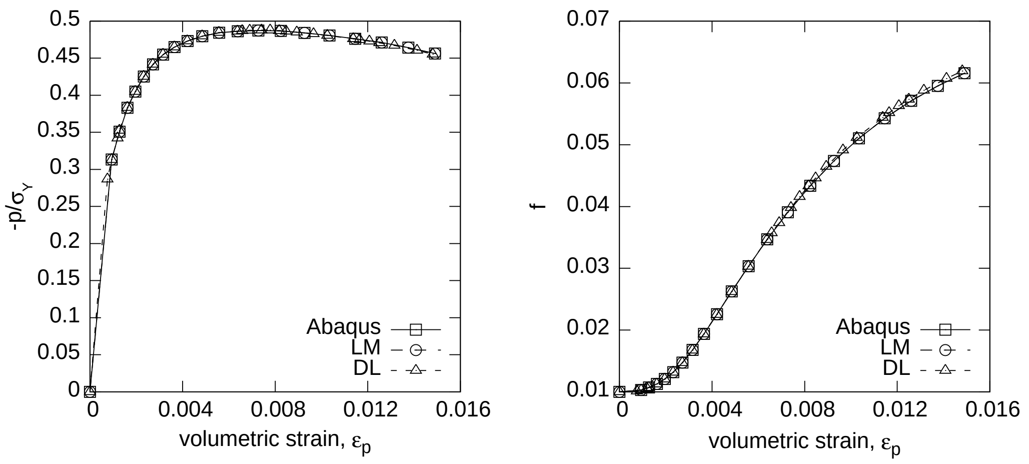
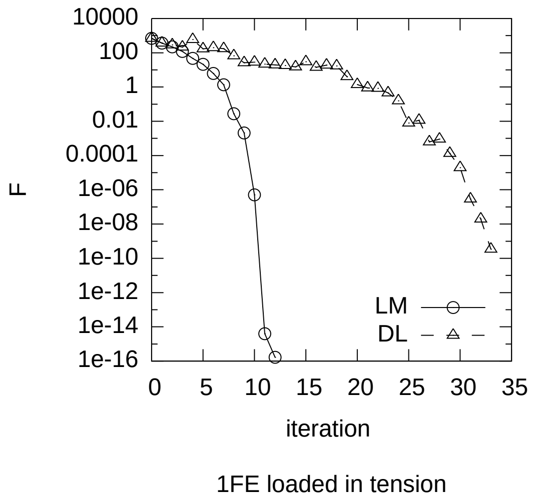


Figure 7: Pressure, p, and void volume fraction, f, plotted against the volumetric strain ε_p for a single FE under tension+shear. Note that the Abaqus' reference implementation and the LM routine took 20 increments, while the DL routine took 30 increments.



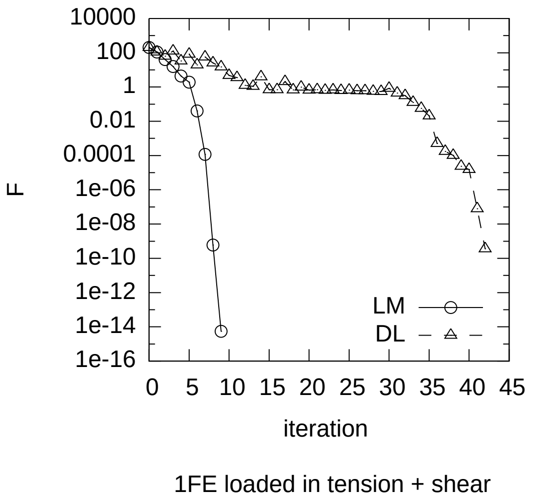


Figure 8: Reduction of the objective function in increment 1 for the single FE models. The LM routine show much better convergence than the DL.

	time steps		rel. avg time inc.	
	LM	DL	LM	DL
tension	25	28	0.040	0.036
tension+shear	20	22	0.050	0.045

Table 1: Total number of time steps and relative average time increment (average time increment divided over the total time) for single FE models.

In addition, for each time increment, the DL routine took many more iterations to converge than the LM routine. Two typical examples are shown in Fig. 8.

We could not find a combination of k and $\boldsymbol{D}$ (or internal auto scaling) which would lead to the DL algorithm converging at least as fast as the LM algorithm. **In our experience, the GTN problem always converges faster with the LM than with the DL, no matter the scaling.** Therefore, on the basis of these two simple examples, we conclude that the DL algorithm [15], or at least its implementation in Slatec [25], are less robust than the LM method for the solution of the GTN equations.

5.2. Rod in tension+shear

The example of a cylindrical rod loaded in tension and shear was chosen because very different deformation paths are obtained in different parts within the same FE model. The model and the boundary conditions are shown in Fig. 9. **The rod is 100 mm long and 20 mm in diameter, i.e. the length to diameter ratio is 5.** The top end of the rod is fully constrained. The bottom end is constrained in the axial direction, z, and a prescribed displacement **of 50 mm** is applied to all bottom nodes along x. **The mesh consists of 1944 8-node first order reduced integration elements C3D8R [16].** The deformed shape and the contour plot of the von Mises equivalent stress, q, are also shown in Fig. 9.

Figs. 10 and 11 compare the evolution of pressure, p, and void volume fraction, f, with volumetric strain, ε_p, between the reference Abaqus implementation and the LM for points A and B respectively. There is a perfect agreement between the Abaqus' own implementation and the LM method. Note that the DL routine could not complete the full deformation path.

The complex deformation paths in both point A and B illustrate our earlier point that using the previous time

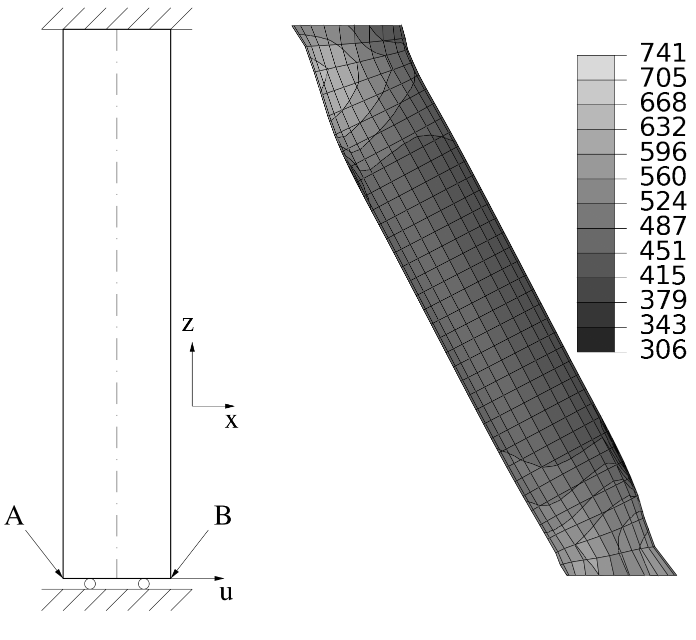


Figure 9: Cylindrical rod under tension and shear showing the boundary conditions and the contour plot of equivalent stress, q, at the end of deformation.

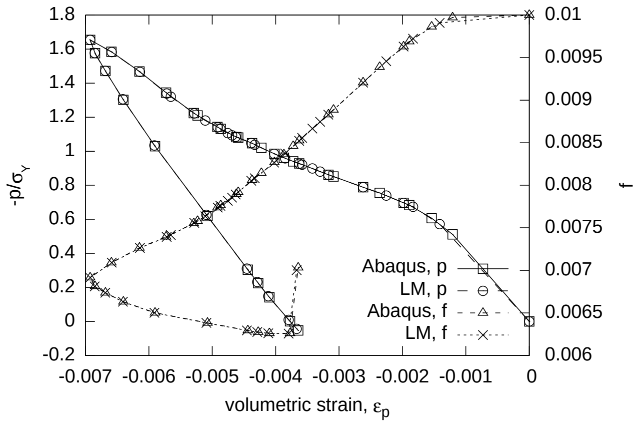


Figure 10: Pressure, p, and void volume fraction, f, evolution for point A of the rod under shear+tension model, see Fig. 9. Note a complex deformation path.

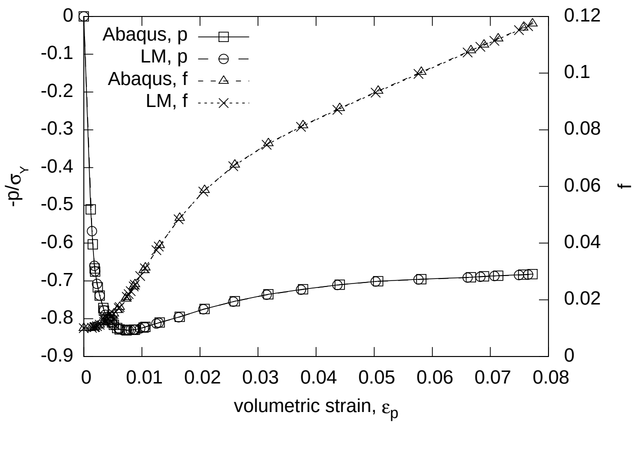


Figure 11: Pressure, p, and void volume fraction, f, evolution for point B of the rod under shear+tension model, see Fig. 9. Note a complex deformation path.

increment solution, $\boldsymbol{x}(t)$, as the starting point for the current time increment solution of the GTN equations is a poor choice.

There are 1944 finite elements in the whole model. The deformation proceeds in 36 steps with the LM and the Abaqus automatic implicit time incrementation scheme, based on mid-point force residuals [16]. Thus the GTN system will have to be solved 69984 times, with different

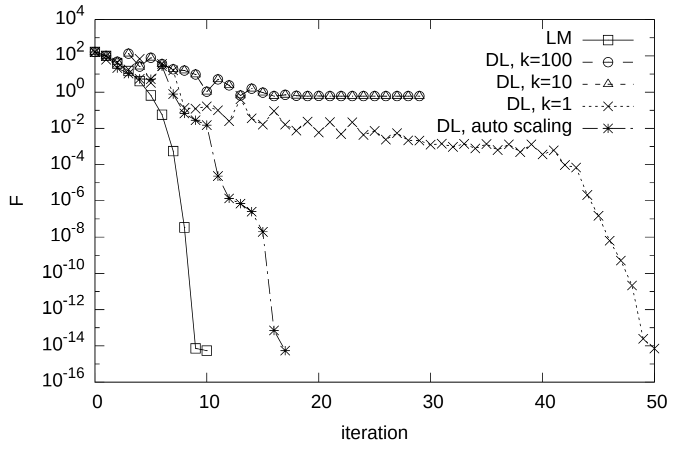


Figure 12: DL problem case 1. 4% of the deformation path. Convergence of the DL an the LM methods for a GTN problem with $p^e = -1.263043927832757 \times 10^3$, $q^e = 5.392240455236696 \times 10^3$, $\varepsilon_q^m(t) = 6.765219019610775 \times 10^{-3}$, $f(t) = 9.855657798939831 \times 10^{-3}$.

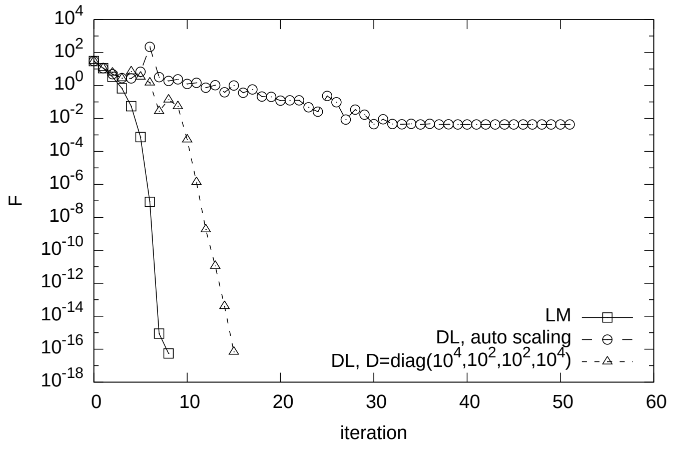


Figure 13: DL problem case 2. 2% of the deformation path. Convergence of DL an LM with $p = -4.338322236568409 \times 10^2$, $q = 2.159184632950240 \times 10^3$, $\varepsilon_q^m(t) = 0$, $f(t) = 0.01$.

set of parameters $p^e, q^e, \varepsilon_q^m(t), f(t)$. It is very awkward and time consuming to change the parameters of the solution algorithm, i.e. $k, \boldsymbol{D}, \epsilon_x$, during the deformation. Therefore, a solution routine is required which is robust enough to solve tens of thousands GTN systems with no manual intervention.

The LM routine can do this with $\boldsymbol{x}^{(0)} = 0$, $k = 100$, $\boldsymbol{D} = \text{diag}(10^4, 10^2, 10^2, 10^4)$, $\epsilon_x = 10^{-2}\sqrt{\epsilon}$. In other words the LM method solves the GTN equations with accuracy that is two orders of magnitude better than the generally accepted value, $\sqrt{\epsilon}$ [24, 23]. Indeed the LM solution is so accurate that Eqns. (5)-(9) are satisfied to 10^{-16} or even to 10^{-17}, whereas Kojic et al [19] claim that using Aravas' original method [7] these equations cannot be satisfied to below 10^{-8}. The fact that the LM method can solve very large sample of GTN problems very accurately, with no change in either $k, \boldsymbol{D}, \epsilon_x$, shows its robustness.

The DL routine with the same parameters works for only 4% of the deformation path. After that it fails to converge. Attempts to find a solution at this point, i.e. a particular set of $p^e, q^e, \varepsilon_q^m(t), f(t)$ GTN parameters, are shown in Fig. 12. By reducing the initial trust region size from 100 down to 1, the DL method converges, although exhibiting very undesirable oscillatory behaviour, i.e. the objective function, F, increases in many iterations. On the contrary, the LM method delivers a steady reduction of F with each iteration. A little more trial and error work shows that for this particular problem auto scaling option delivers the fastest convergence of the DL routine, yet still slower than that of the LM method.

However, switching to auto scaling for *all* 70,000 cases makes the matters worse, and the DL method fails to converge only after 2% of the deformation path, as shown in Fig. 13. In this case switching back to $\boldsymbol{D} = \text{diag}(10^4, 10^2, 10^2, 10^4)$ leads to a fast convergence of the DL method, yet not as fast as that of the LM method.

Fig. 12 indicated that using a smaller initial trust region step, k, is beneficial for the success of the DL method. However, our experiments showed that auto scaling was not successful for a complete deformation path, no matter how small k was.

After extensive trial and error work we found that the most robust combination of parameters for the DL method was $\boldsymbol{D} = \text{diag}(10^3, 10^2, 10^2, 10^3), k = 0.01$, allowing solution of the GTN problems for up to 67% of the full deformation path, Fig. 14. At that point the DL method could not converge and a change of the parameters of the

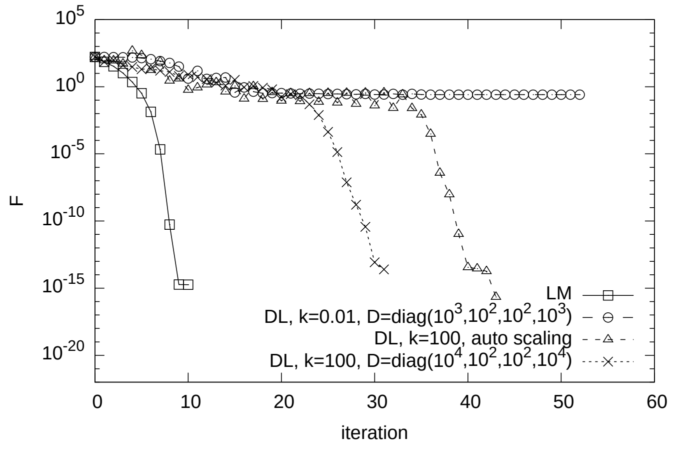


Figure 14: DL problem case 3. 67% of the deformation path. $p = -2.383678148938705 \times 10^3$, $q = 7.044527220580412 \times 10^3$, $\varepsilon_q^m(t) = 9.441005826620887 \times 10^{-2}$, $f(t) = 1.201198152892517 \times 10^{-2}$.

method was required. Fig. 14 shows that switching back to either auto scaling or $\boldsymbol{D} = \mathrm{diag}(10^4, 10^2, 10^2, 10^4)$ was sufficient.

We could not find a set of the DL parameters that would work for the whole of the deformation path. Thus the conclusion we make is that the DL method is less robust than the LM method for the GTN problem.

5.3. 3D tensile model

In this example the extra accuracy of the LM method allows the use of larger time increments in the implicit time integration scheme, compared to the Abaqus' own solver.

The model is a 3D circular cylinder of length L and radius R, see Fig. 15. Following Aravas [7] we use L/R=4. Prescribed vertical displacement is applied to the top surface, while vertical motion of all bottom nodes is constrained. To promote necking at the top of the model, the section A has a smaller radius of 0.995R. The mesh consists of 2478 8-node first order reduced integration elements C3D8R [16]. Also shown in Fig. 15 is the axial cross section showing the deformed shape and the contour plots of pressure, p, at the end of the deformation. The pressure reaches maximum at point C and minimum at point B.

Points B and C undergo very different deformation paths, as shown in Figs. 16 and 17. As in the rod un-

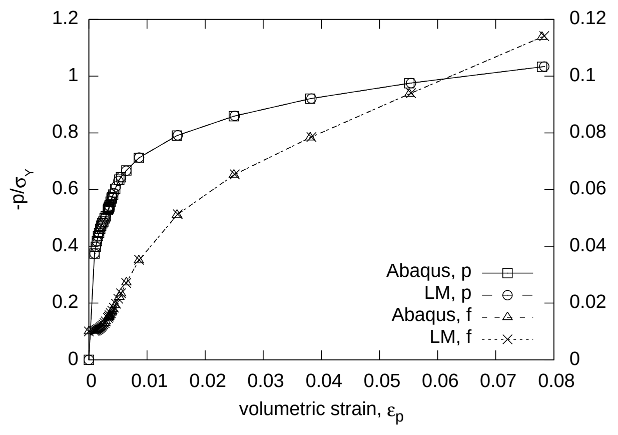


Figure 16: Pressure, p, and void volume fraction, f, evolution in point B of the 3D tensile model, the point with the minimum final p value, see Fig. 15.

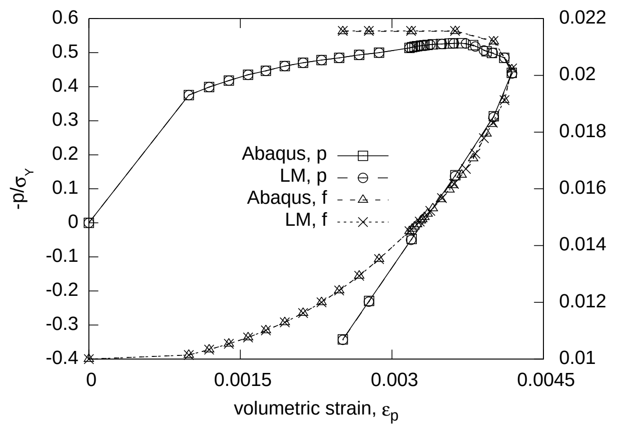


Figure 17: Pressure, p, and void volume fraction, f, evolution in point C of the 3D tensile model, the point with the maximum final p value, see Fig. 15.

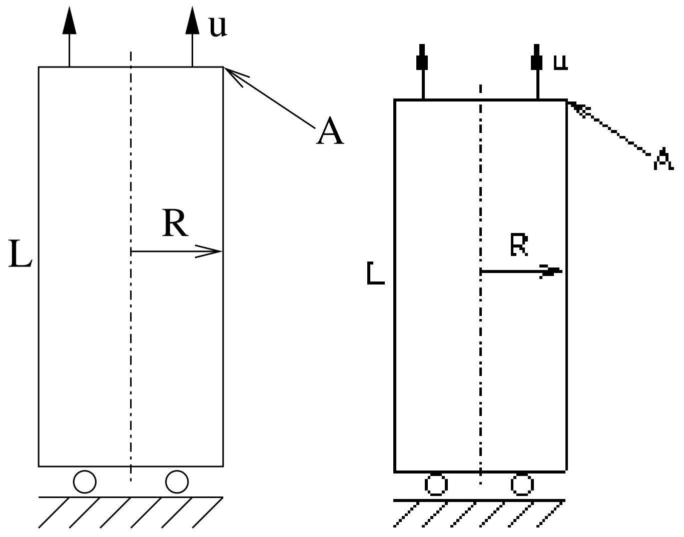


Figure 15: 3D tensile model showing the basic dimensions and the boundary conditions on the left and the contour plot of pressure, p, drawn on the axial cross section. R is radius. Note that at section A, the radius is reduced to $0.995R$, to promote necking.

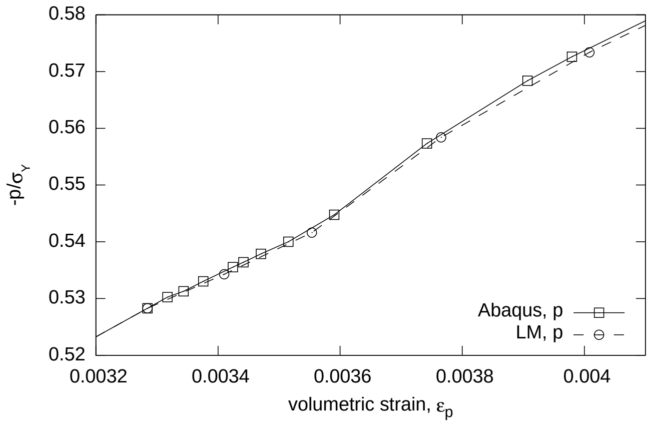


Figure 18: Pressure, p, evolution in point B. Note that the Abaqus solver progresses in much smaller time increments compared to using the LM method. **In this part of the deformation path the minimum relative time increments (time increment divided by the total time) were 1.25×10^{-2} for the LM method and 1.76×10^{-3} for the DL method.**

der shear+tension example, there is a perfect agreement between the Abaqus' own implementation and the LM method.

As in example 5.2, no combination of the DL model parameters could be found for this model to complete the whole of the deformation path.

For the LM routine we used the same values of the parameters here as in section 5.2: $\boldsymbol{x}^{(0)} = 0$, $k = 100$, $\boldsymbol{D} = \mathrm{diag}(10^4, 10^2, 10^2, 10^4)$, $\epsilon_x = 10^{-2}\sqrt{\epsilon}$.

Very low tolerance, $\approx 10^{-10}$, leads to very accurate evaluations of the GTN functions, $\boldsymbol{g}$, and of the Jacobian, $\boldsymbol{J}$. This leads to lower maximum force residuals in the implicit method, hence allowing for larger time increments. As a result the whole of the deformation path is completed within 34 time increments with the Abaqus' own solver, while with the LM method only 27 time increments are required. The larger time increments used by the LM method are clearly seen in Fig. 18, which is a fragment from Fig. 16. **Here the minimum relative time increments (time increment divided by the total time) were 1.25×10^{-2} for the LM method and 1.76×10^{-3} for the DL method.**

Hence, the conclusion for this example is that not only is the LM method more robust than the DL approach, but also that the LM method is significantly more accurate than the Abaqus' own solver, allowing for substantially larger time increments in the implicit time integration scheme.

6. Discussion

The DL method is designed to enforce decrease in F with each increment [15, 27]. Hence, the oscillatory behaviour of the objective function, F, in the DL method, i.e. its increase and decrease on the following iteration, Figs. 8, 12-14, is surprising. We put forward a possible explanation of the observed behaviour.

In most cases the oscillatory behaviour occurs while the current iterate is still far from the global minimum, see Figs. 12-14, where the DL steps are likely to be scaled SD steps. The oscillatory behaviour of the SD method when F is a narrow steep valley is well understood [23, 12, 11, 13]. As shown in section 2.2, Figs. 2 and 3, the GTN objective function indeed contains regions with very dissimilar components of Jacobian, i.e. very narrow valleys.

This problem might become worse if the trust region size, Δ, also oscillates from one increment to another, i.e. if poor and good steps alternate. A good step leads to an increase of Δ, which leads to a bigger next step, which, if F is locally in the shape of a narrow valley, might be too big, and leads to a point where F is greater than in the previous step.

Importantly, both arguments rest on the fact that the objective function for the GTN problem possesses narrow valleys, where the optimal behaviour of the DL method is very sensitive to the method's parameters, primarily the initial trust region size, k, and the diagonal scaling matrix $\boldsymbol{D}$.

Finally, we note that Powell himself remarks that his method 'is less elegant than the one used by Marquardt' and is only preferred 'because it economizes on the number of computer evaluations, when J is approximated numerically' [15]. In this work the exact Jacobian is used, so even this advantage of the DL method is lost.

However, it is still not clear why the LM method performs so much better in narrow valleys. One possible explanation is that in the DL method the switch from SD to Newton's (or dogleg) steps is an abrupt, 'if-then' algorithm. In contrast, in the LM method all steps are somewhere between the SD and Newton's steps, controlled by the damping parameter [17]. Importantly, the dumping parameter controls both the size and the direction of the step. So, we speculate that in the LM method, even when the objective function is a deep valley, the step is leading away from SD direction, due to the damping factor. This helps the LM method converge in such cases. However, more work is needed to clarify this point conclusively.

7. Concluding remarks

Using the freely available Fortran library, we have demonstrated that the Levenberg-Marquardt (LM) method is more robust for the GTN problem than the Powell's dogleg (DL) method. For simple single finite element models, the LM converges faster than the DL. Furthermore, for a 3D problem of a rod under combined tensile and shear loading, no set of the DL parameters could be found that would complete the whole of the deformation path. **In addition, for a 3D uniaxial tension problem using the LM method allows for the use of larger time increments, compared to the Abaqus' Newton's solver, in implicit time integration with automatic time incrementation scheme based on maximum force residuals. This means that using the LM method in the inner loop (GTN equations) leads to lower force residuals in the outer loop (equilibrium equations), which means that the LM method can solve the inner loop more accurately then the Newton's**

method. This leads us to conclude that using the LM method for the inner loop via the Abaqus user material subroutine is more accurate then using Abaqus' own implementation of Newton's method.

Directions for further investigation are clear.

It would be interesting to check whether the conclusions of this work hold as well for other pressure dependent models, e.g. Rousselier's [28].

Finally, the physical meaning of some of the GTN parameters puts limits on their range. q and σ_0 cannot decrease during a plastic loading step, hence $\Delta\varepsilon_q \equiv x_2 \geq 0, \Delta\varepsilon_q^m \equiv x_3 \geq 0$. Hence, formally the GTN problem should be treated as a non-linear minimization with inequality constrains. It would be interesting to explore whether adding these constraints, e.g. using a penalty function [12, 11, 23, 13], would lead to a more robust algorithm or to a faster convergence.

8. Acknowledgments

A Shterenlikht acknowledges financial support from the UK Engineering and Physical Sciences Research Council (EPSRC), grant number EP/H010947/1.

References

[1] V. Tvergaard, A. Needleman, Analysis of the cup-cone fracture in a round tensile bar, Acta Metallurgica 32 (1984) 157–169.

[2] B. K. Dutta, S. Guin, M. K. Sahu, M. K. Samal, A phenomenological form of the q(2) parameter in the Gurson model, International Journal of Pressure Vessels and Piping 85 (2008) 199–210.

[3] M. K. Samal, M. Seidenfuss, E. Roos, K. Balani, Investigation of failure behavior of ferritic-austenitic type of dissimilar steel welded joints, Engineering Failure Analysis 18 (2011) 999–1008.

[4] J. Choung, S.-R. Cho, K. S. Kim, Impact test simulations of stiffened plates using the micromechanical porous plasticity model, Ocean Engineering 37 (2010) 749–756.

[5] N. Le Maout, S. Thuillier, P. Y. Manach, Aluminum alloy damage evolution for different strain paths - Application to hemming process, Engineering Fracture Mechanics 76 (2009) 1202–1214.

[6] P. D. Wu, J. D. Embury, D. J. Lloyd, Y. Huang, K. W. Neale, Effects of superimposed hydrostatic pressure on sheet metal formability, International Journal of Plasticity 25 (2009) 1711–1725.

[7] N. Aravas, On the numerical integration of a class of pressure-dependent plasticity models, International journal for numerical methods in engineering 24 (1987) 1395–1416.

[8] M. Ben Bettaieb, X. Lemoine, L. Duchene, A. M. Habraken, On the numerical integration of an advanced Gurson model, International journal for numerical methods in engineering 85 (2011) 1049–1072.

[9] Z. L. Zhang, On the accuracies of numerical-integration algorithms for Gurson-based pressure-dependent elastoplastic constitutive models, Computer methods in applied mechanics and engineering 121 (1995) 15–28.

[10] P. Rabinowitz (Ed.), Numerical Methods for Non-Linear Algebraic Equations, Gordon and Breach, 1970.

[11] L. E. Scales, Introduction to Non-Linear Optimization, Macmillan, 1985.

[12] J. Nocedal, S. J. Wright, Numerical Optimization, Springer, 2 edition, 2006.

[13] D. G. Luenberger, Y. Ye, Linear and Nonlinear Programming, Springer, 3 edition, 2008.

[14] D. W. Beardsmore, M. A. Wilkes, A. Shterenlikht, An implementation of the Gurson-Tvergaard-Needleman plasticity model for Abaqus Standard using a trust region method, in: Proceedings of ASME Pressure Vessels and Piping Conference/ICPVT-11, 23-27 July 2006, Vancouver, BC, Canada, American Society of Mechanical Engineers, pp. 615–624. ISBN: 0-79184-757-8.

[15] M. J. D. Powell, A hybrid method for non-linear equations, in: P. Rabinowitz (Ed.), Numerical Methods for Non-Linear Algebraic Equations, Gordon and Breach, 1970, pp. 87–114.

[16] Abaqus, User's manual, Version 6.10, Dassault Systèmes, 2010.

[17] J. J. Moré, The Levenberg-Marquardt algorithm: implementation and theory, in: G. A. Watson (Ed.), Lecture Notes in Mathematics, No. 630, - Numerical Analysis, Springer-Verlag, 1978, pp. 105–116.

[18] J. E. Dennis, R. B. Schnabel, Numerical methods for unconstrained optimization and nonlinear equations, SIAM, 1996.

[19] M. Kojic, I. Vlastelica, M. Zivkovic, Implicit stress integration procedure for small and large strains of the Gurson material model, International journal for numerical methods in engineering 53 (2002) 2701–2720.

[20] C. C. Chu, A. Needleman, Void nucleation effects in biaxially stretched sheets, Journal of engineering materials and technology 102 (1980) 249–256.

[21] T. Belytschko, W. K. Liu, B. Moran, Nonlinear Finite Elements for Continua and Structures, John Wiley & Sons, 2000.

[22] J. C. Simo, T. J. R. Hughes, Computational Inelasticity,

Springer, 2000.

[23] P. E. Gill, W. Murray, M. H. Wright, Practical Optimization, Academic Press, 1981.

[24] K. L. Hiebert, DNLS1, FORTRAN subroutine, 1980. Minimize the sum of the squares of M nonlinear functions in N variables by a modification of the Levenberg-Marquardt algorithm, netlib.org/slatec/src/dnls1.f.

[25] K. L. Hiebert, DNSQ, FORTRAN subroutine, 1980. Find a zero of a system of a N nonlinear functions in N variables by a modification of the Powell hybrid method, netlib.org/slatec/src/dnsq.f.

[26] M. H. Wright, Discussion on "The Current State of Software", in: M. J. D. Powell (Ed.), Nonlinear Optimisation 1981, Academic Press, London, 1982, pp. 409–410. Proceedings of the NATO Advanced Research Institute held at Cambridge in July 1981.

[27] M. J. D. Powell, A fortran subroutine for solving systems of nonlinear algebraic equations, in: P. Rabinowitz (Ed.), Numerical Methods for Non-Linear Algebraic Equations, Gordon and Breach, 1970, pp. 115–161.

[28] G. Rousselier, J.-C. Devaux, G. Mottel, G. Devesa, A methodology of ductile fracture analysis based on damage mechanics: an illustration of a local approach of fracture, in: J. D. Landes, A. Saxena, J. G. Merkle (Eds.), Nonlinear fracture mechanics: Volume II - Elastic-Plastic Fracture, ASTM STP 995, 1989, pp. 332–354.