

T ROD

 **UPC**
UNIVERSITAT
POLITÈCNICA
DE CATALUNYA

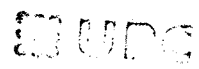
UNIVERSITAT POLITÈCNICA DE CATALUNYA
ESCOLA TÈCNICA SUPERIOR D'ENGINYERS
DE CAMINS, CANALS I PORTS

DEPARTAMENT DE MATEMÀTICA APLICADA III

**ARBITRARY LAGRANGIAN-EULERIAN FORMULATION
OF QUASISTATIC NONLINEAR PROBLEMS**

by

ANTONIO RODRÍGUEZ FERRAN


BIBLIOTECA RECTOR GABRIEL FERRATE
Campus Nord

ADVISOR: ANTONIO HUERTA CEREZUELA

BARCELONA, JULY 1996

4.5.7 Test NECKING

The necking problem is a well-known benchmark test in nonlinear solid mechanics, (Simo, 1988; García-Garino, 1993). A circular bar, with a radius of 6.413 mm and 53.334 mm length, is subjected to uniaxial tension. A slight geometric imperfection (1% radius reduction) induces necking in the central part of the bar. Because of symmetry, only a quarter of the specimen is modelled. The uniaxial constitutive law may be found in Simo (1988).

A mesh of 50 eight-noded quadrilaterals with 2×2 Gauss points is employed to simulate an axial elongation of 14 mm (26% of initial length). To assess the time-increment sensitivity of each algorithm, the computation is performed with three different time-increments (100, 200, and 1000 steps) for both algorithms.

The final deformed shape of the specimen is shown, for the reference solution (second algorithm with 1000 time-steps), in Figure 4.25. Since an Updated Lagrangian formulation is employed, a good shape definition in the necking zone is not obtained. This test will be reproduced in Chapter 5 with an Arbitrary Lagrangian-Eulerian formulation, but the objective here is to compare the two stress update algorithms.

The influence of the time-increment can be seen in Figure 4.26, where radius reduction is plotted versus elongation. The differences are important for the first algorithm, see Figure 4.26a, starting at around 10% elongation and leading to significantly different final values of the neck radius (21%). On the other hand, the second algorithm shows much less sensitivity to the time-increment: the three curves, see Figure 4.26b, are much closer together, with the 200 and 1000 time-step solutions almost superimposed and with a discrepancy in the final neck radius of only 3%. Moreover, these last curves are very similar to those portrayed in Simo (1988).

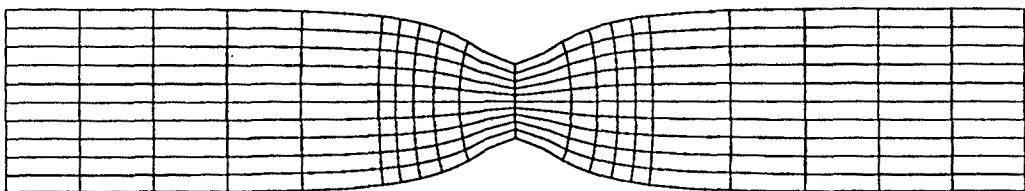


Figure 4.25 Test NECKING. Final deformed shape. Second algorithm with 1000 time-steps

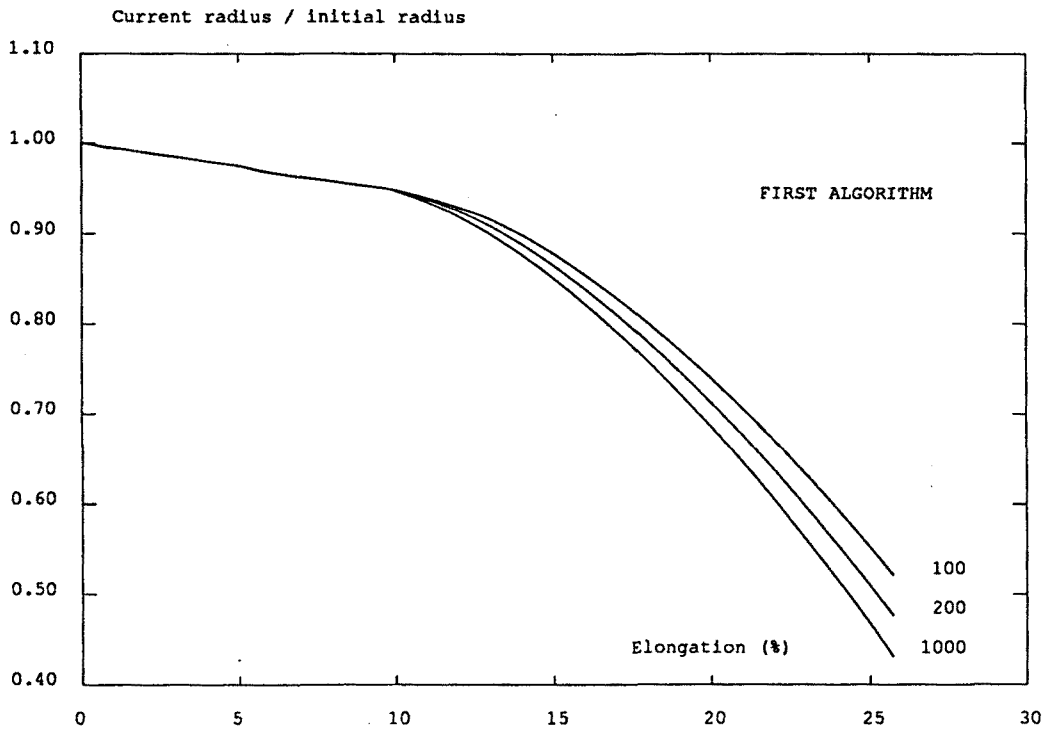


Fig. 4.26a

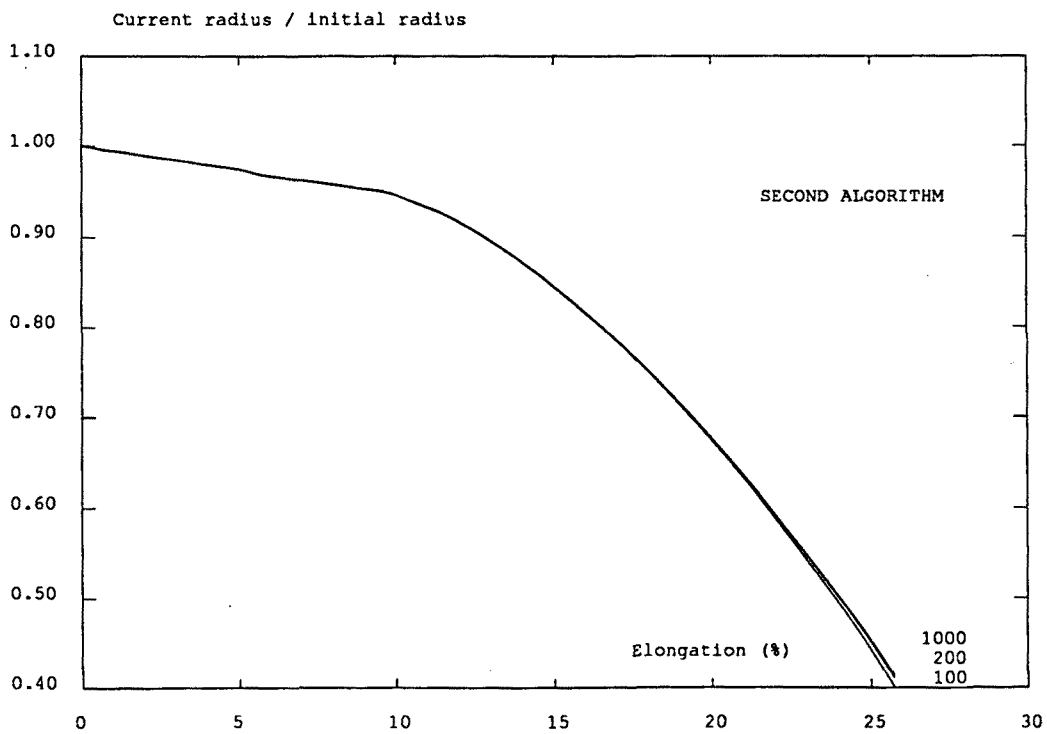


Fig. 4.26b

Figure 4.26 Test NECKING. Radius reduction vs. bar elongation curves computed with 100, 200 and 1000 time-steps. a) First algorithm. b) Second algorithm

4.5.8 Test SHELL

An axisymmetric shell made of an elastic material is clamped at its border, and a ring load P is applied to the shell, causing a deflection v of the apex. This test was employed in Chapter 3 to compare various nonlinear solvers, see Subsection 3.5.2 and Figure 3.1, and will be employed here to assess the performance of the two stress update algorithms for structural finite elements.

This problem is solved in Zienkiewicz & Taylor (1991), with a displacement-controlled technique (Crisfield, 1991), on the central point (i.e., by prescribing increasing values of v), and for different values of the load eccentricity $e = r/R_h$. For comparison purposes, the maximum value of $e = 0.42$ employed in Zienkiewicz & Taylor (1991) has been chosen. The analysis has been performed with two values of the increment of v : $\Delta v = 0.005\text{in}$ and $\Delta v = 0.001\text{in}$.

Figure 4.27 shows the load-deflection curves for the two stress update algorithms. It can be seen that the response for the first algorithm, Figure 4.27a, is much more dependent on the size of Δv than for the second algorithm, Figure 4.27b. This shows again the superior accuracy of the second algorithm. It must be remarked that the solution with the first algorithm and $\Delta v = 0.001\text{in}$ converges to the solution with the second algorithm which, moreover, shows good agreement with that of Zienkiewicz & Taylor (1991) with a load peak around $P = 75\text{lb}$.

For larger values of the eccentricity e , a displacement-controlled method is no longer valid, because the load-deflection curve shows a snap-back behaviour. An arc-length technique, (Crisfield, 1991), is then necessary, see Appendix C or Vila *et al.* (1996). A cylindrical arc-length formulation, combined with the full Newton-Raphson method, has been used to solve the problem with $e = 0.60$. The arc length is automatically updated at every step as proposed by Crisfield (1991), by prescribing a desired number of four iterations per step. An initial arc length of $\Delta \ell = 0.01\text{in}$ and a lower bound of $\Delta \ell = 0.005\text{in}$ have been employed.

The load-deflection curves are presented in Figure 4.28. The computational cost for the two algorithms is shown in Table 4.1. It can be seen that the required number

of both time steps and iterations for the first algorithm is about twice as much as for the second algorithm. Recalling that the computational cost per iteration of the first algorithm is about half the cost per iteration for second algorithm, it can be concluded that the stress update costs approximately the same in both computations. It must be remarked, however, that the overall cost per iteration includes, apart from the stress update, other important and expensive features (evaluation of residuals, resolution of the system of equations, etc.). Therefore, the overall cost for the second algorithm is indeed lower than for the first one.

It is interesting to note that the average number of iterations per step is 4.1 for the first algorithm and 4.2 for the second one, thus indicating a very good performance of the automatic arc-length control, (Crisfield, 1991). Because of its second-order accuracy, the second algorithm shows superior global convergence properties, thus allowing for larger arc lengths. This point is clear from Figure 4.29, which depicts the arc length versus the accumulated length ℓ . Except for a few steps, the first algorithm generally demands smaller $\Delta\ell$ to satisfy the requirement of four iterations per step. As a result, the total number of iterations is higher for the first algorithm. Figure 4.30 shows the accumulated number of iterations versus the accumulated length ℓ .

	Load steps	Iterations	Iterations per step
First algorithm	418	1719	4.1
Second algorithm	198	822	4.2

Table 4.1 Test SHELL. Computational cost of the two stress update algorithms

As a final remark, it is worth mentioning that the second algorithm yields a more accurate load-deflection response in Figure 4.28. This fact can be verified by reproducing the test with the first algorithm and a very small constant $\Delta\ell = 0.005\text{in}$ (i.e., the lower bound previously used, with no automatic update of the arc length). A total of 1098 steps and 3643 iterations are required. As shown in Figure 4.31, the solution then converges to that obtained with the second algorithm with only 198 steps and 822 iterations. In conclusion, the second algorithm provides a clearly better solution, both in terms of accuracy and computational cost.

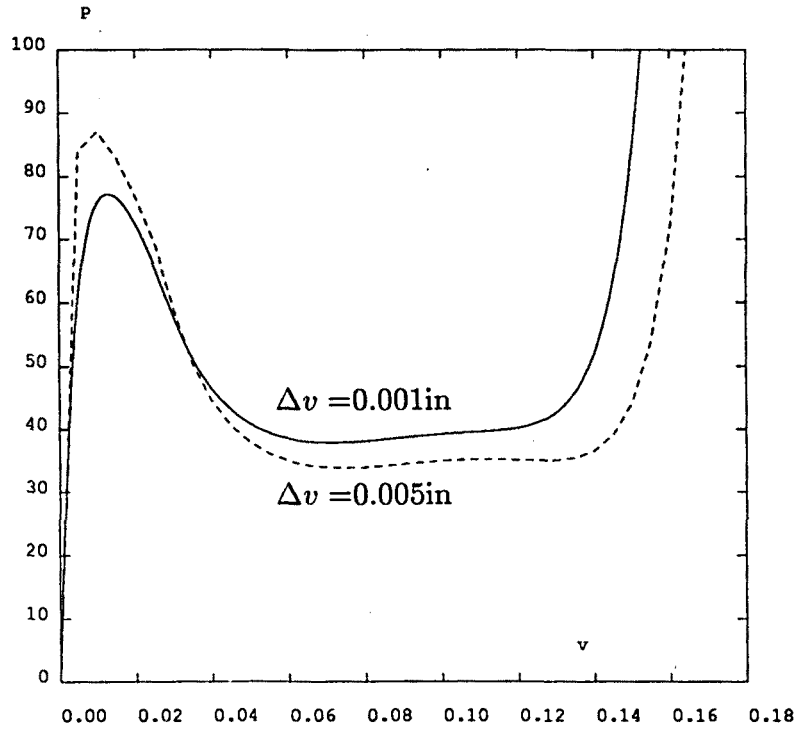


Fig. 4.27a

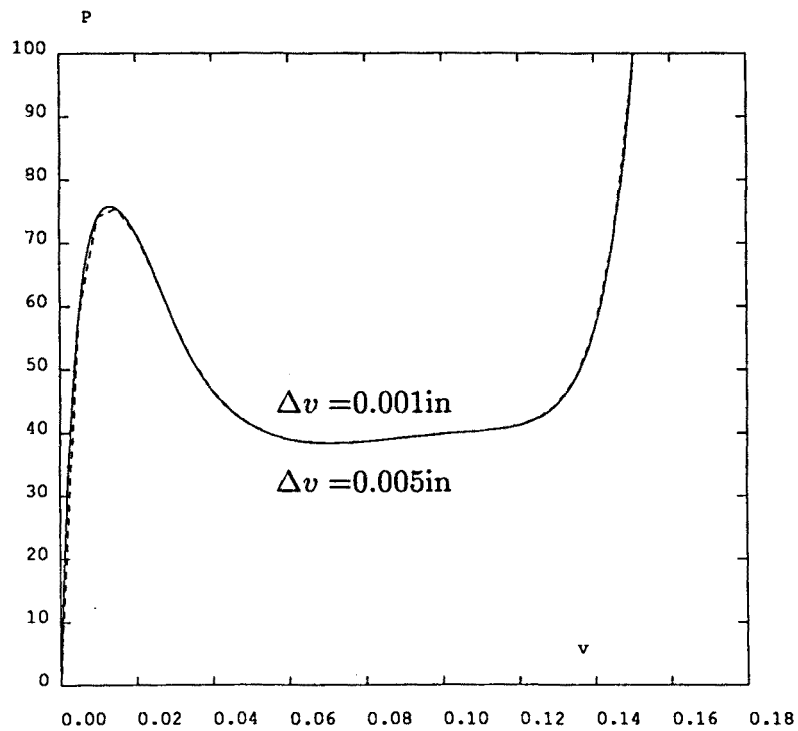


Fig. 4.27b

Figure 4.27 Test SHELL. Load vs. deflection curves for an eccentricity $e = 0.42$. Displacement-control solutions with $\Delta v = 0.005\text{in}$ and $\Delta v = 0.001\text{in}$. a) First algorithm. b) Second algorithm

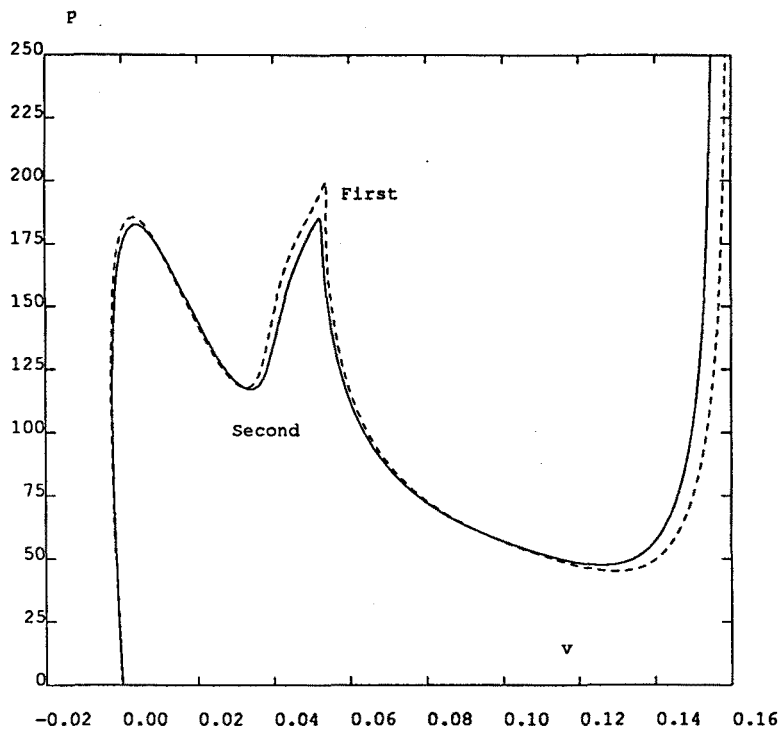


Figure 4.28 Test SHELL. Load vs. deflection curves for an eccentricity $e = 0.60$. Arc-length solutions with automatic arc-length control

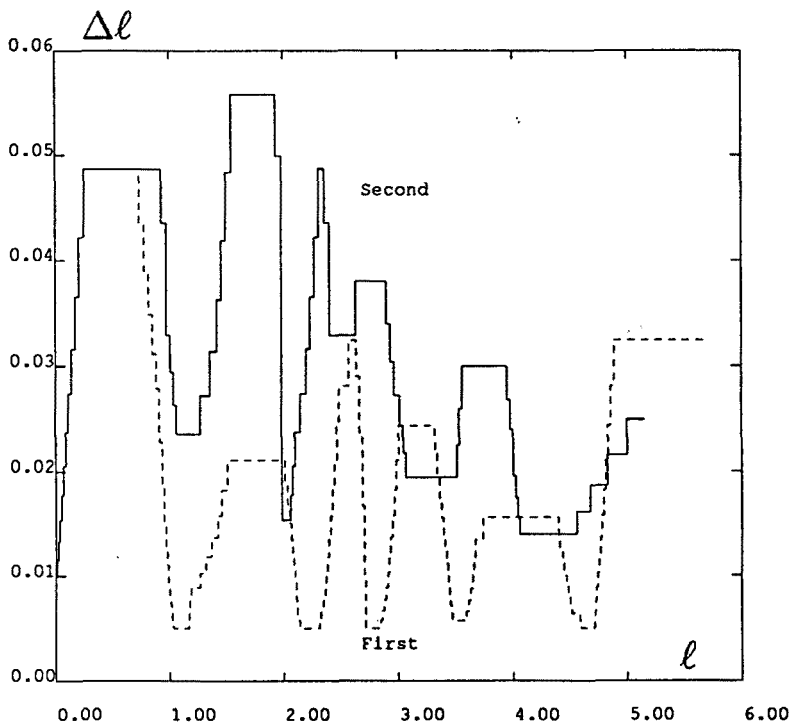


Figure 4.29 Test SHELL. Arc length vs. accumulated length

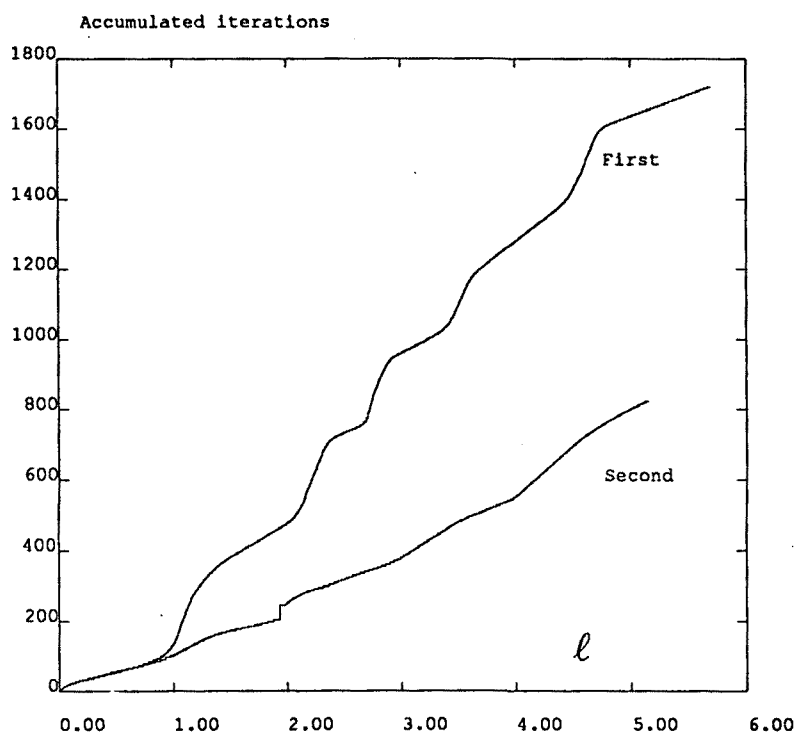


Figure 4.30 Test SHELL. Accumulated iterations vs. accumulated length

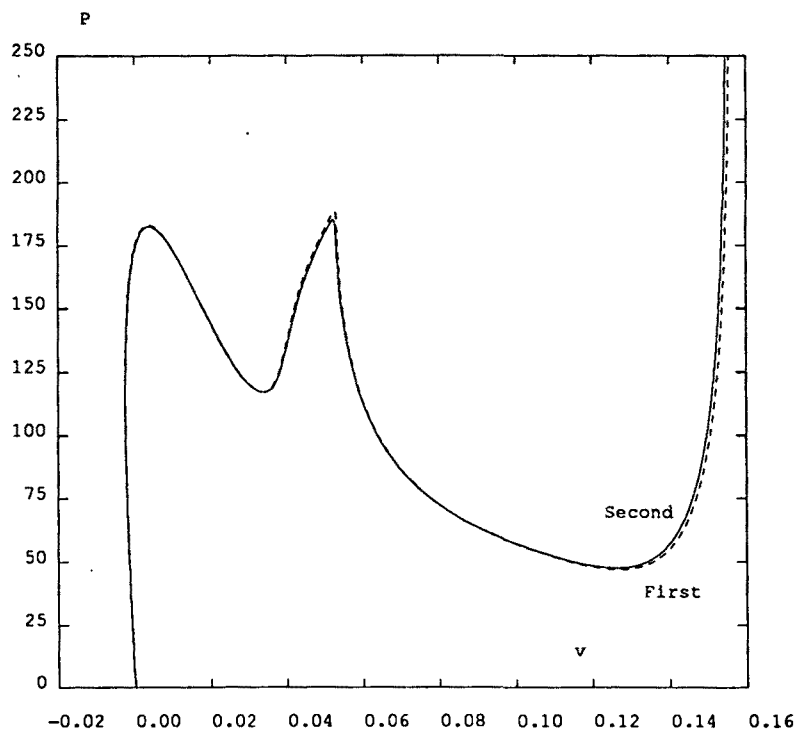


Figure 4.31 Test SHELL. Load vs. deflection curves. Constant $\Delta l = 0.005$ in for the first algorithm

4.6 Concluding remarks

A methodology to compare stress update algorithms for large strains from an analytical point of view has been presented and applied to two simple algorithms for hypoelastic constitutive laws.

An accuracy analysis has been performed to deduce the order of the truncation error associated to each numerical algorithm. This provides an a priori knowledge on the accuracy for each algorithm. Therefore the performance of the algorithms can be studied from an analytical point of view apart from the usual numerical experiments.

The basic ingredient of this approach is the use of convected frames. This choice enables standard techniques of numerical analysis in finite differences to evaluate the accuracy of the numerical time integration.

The convected frame formalism also allows a unified derivation of both algorithms. The first algorithm, (Bathe *et al.*, 1975), which is first-order accurate, uses the full incremental Lagrange strain tensor as the strain measure. The second one measures strain with the usual small strain tensor, but computed in the midstep configuration. This algorithm, (Pinsky *et al.*, 1983), is second-order accurate.

Moreover, it has been shown that the main difference, from an accuracy point of view, resides in the elastic modulus tensor (in particular: when it is evaluated). Because both algorithms use the exact value for the strain increment.

After the error analysis, the two algorithms have been adapted to a fixed frame. With this adaptation, both algorithms can be employed to add large strain capabilities to a small-strain finite element code in a simple way.

Regarding the computational efficiency, the second algorithm is superior for the quasi-static problems considered here, where an implicit integration is performed. This conclusion, however, cannot be readily extended to explicit algorithms for fast-transient dynamics. In those problems, the time step is usually restricted by stability requirements, so first-order accuracy may be sufficient. Moreover, since no iterations are performed and no consistent systems of equations are solved, the stress update has a relevant impact on the overall computational cost.

A set of simple deformation paths (simple shear, uniaxial extension, extension and compression, dilatation, extension and rotation) have been used to assess the relative performance of the two algorithms, both for large-strain elastic and elastoplastic analysis. The general outcome of these numerical tests is in good agreement with the accuracy analysis: the predicted solutions are much more dependent on the time-step for the first-order algorithm than for the second-order one. The extension and rotation test, however, illustrates that the first-order algorithm can have a superior performance for certain stress components in shear-free deformation paths. Finally, two well documented benchmark tests, a necking analysis and a shell under a ring load, confirm the previous conclusions and show the superior performance of the second-order algorithm with continuum and structural elements.

Appendix 4.A The midstep material components of the modulus tensor

It is shown in this Appendix that the constant-velocity assumption yields a second-order approximation to the exact midstep material components of the modulus tensor. This property, Eq. (4.32), is essential for the second-order accuracy of the second stress update algorithm.

It is assumed that the modulus tensor C depends on the deformation (Pinsky *et al.*, 1983), represented in a material setting by the metric tensor G . That is, the dependence of C with respect to time is not explicit, but associated to the deformation of the body, and can be written as

$$C^{ijkl}(t) = f(g_{mn}(t)), \quad (4.A.1)$$

where f is a function of the components of the metric tensor g_{mn} . The exact midstep components of the modulus tensor are then

$${}^{n+\frac{1}{2}}C^{ijkl} = f({}^{n+\frac{1}{2}}g_{mn}), \quad (4.A.2)$$

where ${}^{n+\frac{1}{2}}g_{mn}$ are the exact midstep components of the metric tensor, while the approximate midstep components of C are

$${}^{n+\frac{1}{2}}\bar{C}^{ijkl} = f({}^{n+\frac{1}{2}}\bar{g}_{mn}), \quad (4.A.3)$$

with \bar{g}_{mn} the constant-velocity approximation of g_{mn} . Recalling the definition of g_{mn} , Eq. (4.7), and the relation between exact and approximate midstep orthonormal coordinates, ${}^{n+\frac{1}{2}}z^\alpha$ and ${}^{n+\frac{1}{2}}\bar{z}^\alpha$, Eq. (4.39), it can be easily concluded that

$${}^{n+\frac{1}{2}}g_{mn} = {}^{n+\frac{1}{2}}\bar{g}_{mn} + \mathcal{O}(\Delta t^2). \quad (4.A.4)$$

As previously shown for other quantities, the constant-velocity assumption also provides a second-order approximation to the components of the metric tensor. Substituting Eq. (4.A.4) into Eq. (4.A.2) and performing a first-order Taylor's expansion finally renders, assuming that the first derivative of f is bounded,

$${}^{n+\frac{1}{2}}C^{ijkl} = {}^{n+\frac{1}{2}}\bar{C}^{ijkl} + \mathcal{O}(\Delta t^2),$$

which is Eq. (4.32).

Appendix 4.B Stress update for shear-free deformation paths

Here, a particular behaviour of the first stress update algorithm is demonstrated. This algorithm correctly predicts null shear stresses for any shear-free deformation path. The second algorithm, on the contrary, only provides similar results for some particular deformation paths of this type.

The general expression for shear-free deformation paths is

$$\left. \begin{aligned} x(t) &= Xa_x(t)\cos\theta(t) - Ya_y(t)\sin\theta(t) \\ y(t) &= Xa_x(t)\sin\theta(t) + Ya_y(t)\cos\theta(t) \end{aligned} \right\}, \quad (4.B.1)$$

where $a_x(t)$ and $a_y(t)$ represent the axial deformations in the x and y directions respectively, and $\theta(t) = 2\pi t$ accounts for the rigid rotation. Taking $a_x(t) = 1+t$ and $a_y(t) = 1$ yields the extension-rotation test of Subsection 4.5.6. An extension-compression-rotation test with no change in volume can be obtained with $a_x(t) = \frac{1}{a_y(t)} = 1+t$, and choosing $a_x(t) = a_y(t) = 1+t$ renders a dilatation-rotation test.

The deformation gradient of motion (4.B.1) is

$$\mathbf{F} = \begin{bmatrix} a_x(t)\cos\theta(t) & -a_y(t)\sin\theta(t) \\ a_x(t)\sin\theta(t) & a_y(t)\cos\theta(t) \end{bmatrix}, \quad (4.B.2)$$

and the Jacobian is $J = a_x(t)a_y(t)$.

To assess the behaviour of the two stress update algorithms, it is enough to study the stress predicted after one time-step, and to check if the shear stress component (in rotated axes, to account for the rigid rotation) is null or not.

First stress update algorithm

For the first time-step Δt , the incremental Lagrange strain tensor is, Eq. (4.42),

$${}^n\Delta\varepsilon = \frac{1}{2} \begin{bmatrix} (A_x^2 - 1) & 0 \\ 0 & (A_y^2 - 1) \end{bmatrix}, \quad (4.B.3)$$

with $A_x = a_x(\Delta t)$ and $A_y = a_y(\Delta t)$. Recalling the expression of the first algorithm, Eq. (4.46a), the stress after one time-step is

$$\boldsymbol{\sigma} = \frac{E}{2A_xA_y} \begin{bmatrix} (B_x\cos^2\Theta + B_y\sin^2\Theta) & (B_x\sin\Theta\cos\Theta - B_y\sin\Theta\cos\Theta) \\ (B_x\sin\Theta\cos\Theta - B_y\sin\Theta\cos\Theta) & (B_x\sin^2\Theta + B_y\cos^2\Theta) \end{bmatrix}, \quad (4.B.4)$$

with $\Theta = 2\pi\Delta t$, $B_x = A_x^2(A_x^2 - 1)$ and $B_y = A_y^2(A_y^2 - 1)$.

If the stress $\boldsymbol{\sigma}$ is transformed into the rotated stress $\boldsymbol{\sigma}' = \mathbf{R}\boldsymbol{\sigma}\mathbf{R}^T$, where \mathbf{R} is a orthogonal tensor that accounts for the rigid rotation, the result is

$$\boldsymbol{\sigma}' = \frac{E}{2} \begin{bmatrix} \frac{A_x}{A_y}(A_x^2 - 1) & 0 \\ 0 & \frac{A_y}{A_x}(A_y^2 - 1) \end{bmatrix}. \quad (4.B.5)$$

Equation (4.B.5) shows that null shear stresses $\sigma_{x'y'}$ are predicted, independently of the axial deformations A_x and A_y . It can be easily checked that the analysis just performed for the first time-step can be extended to the successive increments. In conclusion, the first algorithm predicts null shear stresses (in the rotated axes) $\sigma_{x'y'}$ for any number of time-steps and for any choice of $a_x(t)$ and $a_y(t)$.

Second stress update algorithm

A similar analysis has been performed for the second algorithm. Since the midstep configuration is employed, the required computations are rather cumbersome in this case. The incremental displacements for the first time-step can be computed from Eq. (4.B.1). After that, the midstep coordinates are obtained following Eq. (4.24). Then the incremental displacements are written in terms of the midstep coordinates, so the strain increment can be written as, Eq. (4.47),

$${}^{n+\frac{1}{2}}\overline{\Delta\varepsilon} = \frac{2}{A_x A_y + (A_x + A_y)\cos\Theta + 1} \begin{bmatrix} A_x A_y + (A_x - A_y)\cos\Theta - 1 & (A_x - A_y)\sin\Theta \\ (A_x - A_y)\sin\Theta & A_x A_y - (A_x - A_y)\cos\Theta - 1 \end{bmatrix} \quad (4.B.6)$$

Once the strain increment is known, the stress σ is computed according to Eq. (4.51a) and then transformed into the rotated stress σ' . The final result is that the shear stress $\sigma_{x'y'}$ is proportional to a factor that depends on axial deformations A_x and A_y :

$$\sigma_{x'y'} \sim -A_x^3 A_y^2 + A_x^2 A_y^3 + A_x^2 A_y - A_x A_y^2. \quad (4.B.7)$$

It can be seen from Eq. (4.B.7) that $\sigma_{x'y'}$ is not null for any choice of $a_x(t)$ and $a_y(t)$. Taking $a_x(t) = 1 + t$ and $a_y(t) = 1$, for instance, yields a non-null $\sigma_{x'y'}$, as shown for the extension-rotation test of Subsection 4.5.6. On the contrary, for $a_x(t) = a_y(t)$ (dilatation-rotation test) and for $a_x(t)a_y(t) = 1$ (extension-compression-rotation test, with no volume change), null values for $\sigma_{x'y'}$ are predicted. In conclusion, the second stress update algorithm predicts correct null shear stresses (in the rotated axes) for some particular deformation paths of the form (4.B.1), but not for every shear-free path.

Chapter 5

ARBITRARY LAGRANGIAN-EULERIAN QUASISTATIC ANALYSIS

5.1 Introduction

The Arbitrary Lagrangian-Eulerian (ALE) formulation was developed with the goal of overcoming the limitations of the Lagrangian and Eulerian descriptions, the two classical formulations of continuum mechanics (Malvern, 1969).

In a Lagrangian formulation the reference system is attached to the continuum and moves with it. A particle can be readily identified by its material coordinates, although it occupies different points in space as it moves. If a numerical method is used to solve the problem, the mesh nodes represent the same particles throughout the computation. As there is no relative motion between the continuum and the mesh, there are no convective terms in the conservation laws that govern the problem; this eases considerably the task of solving them. Moreover, boundary conditions can be treated in a straightforward manner as the contour is always represented by the same nodes. On the other hand, a Lagrangian formulation may result inappropriate in a large deformation analysis, where elements may become too distorted or even entangled. Lagrangian formulations are common in solid and structural mechanics, see Bathe (1982), Crisfield

(1991), Zienkiewicz & Taylor (1991).

In an Eulerian formulation the reference is fixed in the laboratory and the material flows through it. Attention is put in points in space, which are constantly identified by their spatial coordinates even though different particles occupy them at different times. Eulerian formulations are mostly used in fluid mechanics, because the fixed mesh can handle arbitrarily large distortions of the continuum. It has two important drawbacks: 1) the relative motion between mesh and material leads to convective terms in the governing equations which present numerical difficulties and 2) it is cumbersome to describe large boundary motion.

In an attempt to combine the strong points of both formulations, the ALE formulation, (Donéa, 1983), was initially developed for fluid dynamics, both in Finite Differences (Noh, 1964; Trulio, 1966; Hirt *et al.*, 1974) and in the Finite Element Method (Donéa *et al.*, 1977; Belytschko & Kennedy, 1978; Hughes *et al.*, 1978, 1981; Huerta & Liu, 1988). In the ALE formulation, the reference system is neither attached to the particles nor fixed in space; it moves with respect to the laboratory with an arbitrary velocity, independent from that of the continuum. A node in the reference mesh corresponds, as time passes, to different points in space (because the mesh is not fixed in space) and also to different particles (because the mesh motion is not that of the material). The basic idea is to choose the mesh movement in such a way that 1) material surfaces can be correctly tracked (see, for instance, Sarrate (1996)) and 2) the mesh elements are kept as regular as possible.

More recently, the ALE formulation has been extended to nonlinear solid mechanics (Schreurs, 1983; Huétink, 1986; Liu *et al.*, 1986; Benson, 1989; Huerta & Casadei, 1991; Ghosh & Kikuchi, 1991). Compared to ALE fluid dynamics, the main additional difficulty is then the stress update (i.e., the time-integration of the constitutive equation). Indeed, the rate-form constitutive equation of ALE nonlinear solid mechanics contains a convective term that accounts for the relative motion between mesh and material. Because of this, the stress update cannot be performed at the Gauss-point level, as simply as in a Lagrangian analysis, and more involved procedures are required. In fact, the correct treatment of the convective term in the constitutive equation is a key point

in ALE nonlinear solid mechanics, see Benson (1992).

This Chapter is organized as follows. The basic ALE equations are reviewed in Section 5.2. Special emphasis is put in stressing the similarities and differences between ALE transient and quasistatic analyses. Section 5.3 presents the state-of-the-art in ALE stress update, a crucial aspect of ALE nonlinear analysis. The two basic issues are 1) explicit versus implicit algorithms and 2) split versus unsplit algorithms. Unsplit algorithms treat the ALE governing equations as a whole, while split algorithms perform an operator split to handle material and convective terms separately. It is shown that any combination of explicit/implicit, split/unsplit can be found in the literature.

Section 5.4 is dedicated to ALE stress update for quasistatic analysis. After assessing the various options, it is concluded that the most natural choice is to use an split formulation. By doing so, the material terms in the governing equations can be handled just like in an Updated Lagrangian analysis, so the nonlinear equation solvers of Chapter 3 and the stress update algorithms of Chapter 4 are directly applicable. After that, the convection terms are handled with explicit algorithms. Some numerical examples are employed in Section 5.5 to assess the performance of the various stress update algorithms. Finally, some concluding remarks are made in Section 5.6.

5.2 Basic ALE equations

5.2.1 ALE kinematics

Two domains are commonly employed in continuum mechanics: the material domain R_X , made up of material particles X , and the spatial domain R_x , consisting of spatial points x . Neither of these two is taken as the reference in the ALE description, so a third domain is needed: the referential domain R_χ , formed by reference (or grid) points χ .

One-to-one transformations between these three domains are needed. The referential domain R_χ is mapped into the material and spatial domains by Ψ and Φ respectively. The particle motion η may then be put as $\eta = \Phi \circ \Psi^{-1}$, clearly showing that, of course,

the three mappings Ψ , Φ and η are not independent.

The particle motion η can be represented as

$$\begin{aligned} \eta: \quad R_X \times [0, \infty) &\longrightarrow R_x \times [0, \infty) \\ (X, t) &\longmapsto (x, t) \end{aligned} \quad (5.1a)$$

with

$$x_i = x_i(X, t) \quad ; \quad t = t. \quad (5.1b)$$

Equation (5.1b) explicitly states the particular nature of η : 1) the spatial coordinates x depend both of the material particle X and time t , and 2) physical time is measured by the same variable t in both material and spatial domains. For every fixed instant t , the mapping

$$\begin{aligned} \eta_t: \quad R_X &\longrightarrow R_x \\ X &\longmapsto x \end{aligned} \quad (5.1c)$$

defines a configuration Ω of the spatial domain R_x . It is convenient to employ a matrix representation for the gradient of η ,

$$\frac{\partial \eta}{\partial (X, t)} = \begin{bmatrix} \frac{\partial x}{\partial X} & v \\ \mathbf{0}^T & 1 \end{bmatrix}, \quad (5.2)$$

where $\mathbf{0}^T$ is a null row-vector and the material velocity v is

$$v = \left. \frac{\partial x}{\partial t} \right|_X, \quad (5.3)$$

where $\left|_X$ means "holding X fixed".

In a similar way, the mapping Φ from the referential domain to the spatial domain is represented by

$$\begin{aligned} \Phi: \quad R_\chi \times [0, \infty) &\longrightarrow R_x \times [0, \infty) \\ (\chi, t) &\longmapsto (x, t) \end{aligned} \quad (5.4)$$

and its gradient is

$$\frac{\partial \Phi}{\partial(\chi, t)} = \begin{bmatrix} \frac{\partial x}{\partial \chi} & \hat{v} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (5.5)$$

where now the mesh velocity \hat{v} is involved,

$$\hat{v} = \left. \frac{\partial x}{\partial t} \right|_{\chi}. \quad (5.6)$$

Note that, since both the material and the mesh move with respect to the laboratory, corresponding material and mesh velocities have been defined, Eqs. (5.3) and (5.6), by derivating with respect to time the equations of material motion and mesh motion respectively.

Finally, regarding Ψ , it is convenient to represent directly its inverse Ψ^{-1} ,

$$\begin{aligned} \Psi^{-1}: \quad R_X \times [0, \infty) &\longrightarrow R_\chi \times [0, \infty) \\ (X, t) &\longmapsto (\chi, t) \end{aligned} \quad (5.7)$$

and its gradient

$$\frac{\partial \Psi^{-1}}{\partial(X, t)} = \begin{bmatrix} \frac{\partial \chi}{\partial X} & w \\ \mathbf{0}^T & 1 \end{bmatrix}, \quad (5.8)$$

where velocity w is defined as

$$w = \left. \frac{\partial \chi}{\partial t} \right|_X, \quad (5.9)$$

and can be interpreted as the particle velocity in the referential domain, since it measures the time variation of the referential coordinate χ holding the material particle X fixed.

The relation between velocities v , \hat{v} and w can be obtained by derivating $\eta = \Phi \circ \Psi^{-1}$,

$$\frac{\partial \eta}{\partial(X, t)}(X, t) = \frac{\partial \Phi}{\partial(\chi, t)}(\Psi^{-1}(X, t)) \frac{\partial \Psi^{-1}}{\partial(X, t)}(X, t) = \frac{\partial \Phi}{\partial(\chi, t)}(\chi, t) \frac{\partial \Psi^{-1}}{\partial(X, t)}(X, t), \quad (5.10)$$

or, in matrix format:

$$\begin{bmatrix} \frac{\partial \mathbf{x}}{\partial \mathbf{X}} & \mathbf{v} \\ \mathbf{0}^T & 1 \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathbf{x}}{\partial \boldsymbol{\chi}} & \hat{\mathbf{v}} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \frac{\partial \boldsymbol{\chi}}{\partial \mathbf{X}} & \mathbf{w} \\ \mathbf{0}^T & 1 \end{bmatrix}, \quad (5.11)$$

which yields, after block-multiplication,

$$\mathbf{v} = \hat{\mathbf{v}} + \frac{\partial \mathbf{x}}{\partial \boldsymbol{\chi}} \mathbf{w}. \quad (5.12)$$

Equation (5.12) may be written as

$$\mathbf{c} := \mathbf{v} - \hat{\mathbf{v}} = \frac{\partial \mathbf{x}}{\partial \boldsymbol{\chi}} \mathbf{w}, \quad (5.13)$$

thus defining the convective velocity \mathbf{c} , that is, the relative velocity between the material and the mesh.

Remark 5.1: The convective velocity \mathbf{c} , Eq. (5.13), should not be confused with \mathbf{w} , Eq. (5.9). As stated before, \mathbf{w} is the particle velocity as seen from the referential domain $R_{\boldsymbol{\chi}}$, whereas \mathbf{c} is the particle velocity relative to the mesh as seen from the spatial domain $R_{\mathbf{x}}$ (both \mathbf{v} and $\hat{\mathbf{v}}$ are variations of coordinate \mathbf{x}). In fact, Eq. (5.13) implies that $\mathbf{c} = \mathbf{w}$ if and only if $\partial \mathbf{x} / \partial \boldsymbol{\chi} = \mathbf{I}$ (where \mathbf{I} is the identity tensor), that is, when the mesh motion is purely translational, without rotations or deformations of any kind.

After the fundamentals on ALE kinematics have been presented, it should be remarked that both Lagrangian or Eulerian formulations may be obtained as particular cases. With the choice $\boldsymbol{\Psi} = \mathbf{I}$, Eq. (5.7) reduces to $\mathbf{X} \equiv \boldsymbol{\chi}$ and a Lagrangian description results: the mesh and material velocities, Eqs. (5.3) and (5.6), coincide, and the convective velocity \mathbf{c} , Eq. (5.13), is null.

If, on the other hand, $\boldsymbol{\Phi} = \mathbf{I}$, Eq. (5.4) simplifies into $\mathbf{x} \equiv \boldsymbol{\chi}$, thus implying an Eulerian description: a null mesh velocity is obtained from Eq. (5.6) and the convective velocity \mathbf{c} is simply the material velocity \mathbf{v} .

In order to express the conservation laws in an ALE framework, the relation between time derivatives are needed. Let F be a scalar physical quantity, described by $f(\mathbf{x}, t)$, $f^*(\chi, t)$ and $f^{**}(X, t)$ in the spatial, referential and material domains respectively. Stars are employed to emphasize that the functional forms are, in general, different.

With the help of the mapping Ψ , the transformation from $f^*(\chi, t)$ to $f^{**}(X, t)$ can be put as

$$f^{**} = f^* \circ \Psi^{-1}, \quad (5.14)$$

and derivation with respect to time yields

$$\frac{\partial f^{**}}{\partial(X, t)}(X, t) = \frac{\partial f^*}{\partial(\chi, t)}(\chi, t) \frac{\partial \Psi^{-1}}{\partial(X, t)}(X, t). \quad (5.15)$$

Equation (5.15) is amenable to the matrix form

$$\left[\frac{\partial f^{**}}{\partial X} \quad \frac{\partial f^{**}}{\partial t} \right] = \left[\frac{\partial f^*}{\partial \chi} \quad \frac{\partial f^*}{\partial t} \right] \begin{bmatrix} \frac{\partial \chi}{\partial X} & \mathbf{w} \\ \mathbf{0}^T & 1 \end{bmatrix}, \quad (5.16)$$

which renders, after block-multiplication,

$$\frac{\partial f^{**}}{\partial t} = \frac{\partial f^*}{\partial t} + \frac{\partial f^*}{\partial \chi} \mathbf{w}. \quad (5.17)$$

By employing Eq. (5.13), Eq. (5.17) may be rearranged into

$$\frac{\partial f^{**}}{\partial t} = \frac{\partial f^*}{\partial t} + \frac{\partial f}{\partial \mathbf{x}} \mathbf{c}. \quad (5.18)$$

Dropping the stars to ease the notation, the fundamental ALE relation between material time derivatives, referential time derivatives and spatial derivatives is finally cast as

$$\frac{\partial f}{\partial t} \Big|_X = \frac{\partial f}{\partial t} \Big|_\chi + c_i \frac{\partial f}{\partial x_i} \quad (5.19)$$

which can be interpreted in the usual way: the variation of the physical magnitude F for a given particle X is the local variation (i.e., with respect to the reference χ) plus a convective term taking into account the relative motion between the material and the reference system.

Mesh acceleration plays no role in the ALE formulation, so only the material acceleration \mathbf{a} is needed, which may be expressed in the Lagrangian, Eulerian or ALE formulation respectively as

$$\mathbf{a} = \left. \frac{\partial \mathbf{v}}{\partial t} \right|_X; \quad (5.20a)$$

$$\mathbf{a} = \left. \frac{\partial \mathbf{v}}{\partial t} \right|_{\mathbf{x}} + \mathbf{v} \frac{\partial \mathbf{v}}{\partial \mathbf{x}}; \quad (5.20b)$$

$$\mathbf{a} = \left. \frac{\partial \mathbf{v}}{\partial t} \right|_{\chi} + \mathbf{c} \frac{\partial \mathbf{v}}{\partial \mathbf{x}}. \quad (5.20c)$$

It must be noted that the ALE expression of acceleration, Eq. (5.20c), is simply a particularization of the fundamental relation (5.19), taking material velocity as the physical quantity F .

Since acceleration is the material derivative (i.e., for a fixed particle X) of velocity, Eq. (5.20a), two terms are needed to represent it in both the Eulerian and the ALE formulations, Eqs. (5.20b) and (5.20c): the local acceleration, $(\partial \mathbf{v} / \partial t)|_{\mathbf{x}}$ or $(\partial \mathbf{v} / \partial t)|_{\chi}$, and the convective acceleration, $\mathbf{v}(\partial \mathbf{v} / \partial \mathbf{x})$ or $\mathbf{c}(\partial \mathbf{v} / \partial \mathbf{x})$. These convective terms reflect the fact that particle X is neither attached to spatial point \mathbf{x} nor to grid point χ . One of the benefits of the ALE formulation in fluid dynamics is the reduction of convective terms in the governing equations: if the mesh motion is properly selected, the convective velocity \mathbf{c} is smaller than the material velocity \mathbf{v} , Eq. (5.13), so the ALE convective acceleration $\mathbf{c}(\partial \mathbf{v} / \partial \mathbf{x})$ is smaller than the Eulerian convective acceleration $\mathbf{v}(\partial \mathbf{v} / \partial \mathbf{x})$.

5.2.2 ALE transient analysis

Conservation laws

Equation (5.19) is the starting point to deduce the three fundamental conservation laws of continuum mechanics (mass, momentum and energy) in the ALE description. If mechanical effects are uncoupled from thermal effects, then the mass and momentum equations can be solved independently from the energy equation. The ALE version of these two equations is, (Donéa, 1983),

Balance of mass

$$\frac{\partial \rho}{\partial t} \Big|_{\mathcal{X}} + c_j \frac{\partial \rho}{\partial x_j} = -\rho \frac{\partial v_j}{\partial x_j}, \quad (5.21a)$$

Balance of momentum

$$(\rho a_i) = \rho \frac{\partial v_i}{\partial t} \Big|_{\mathcal{X}} + \rho c_j \frac{\partial v_i}{\partial x_j} = \frac{\partial \sigma_{ij}}{\partial x_j} + b_i. \quad (5.21b)$$

where ρ is the density, σ is the Cauchy stress tensor and b is the force per unit volume.

A standard procedure in nonlinear solid mechanics is dropping the equation of mass balance (5.21a), which is not explicitly accounted for, thus solving only the momentum balance (5.21b). In solid mechanics, the domain boundary is typically composed of material surfaces. Since these surfaces are accurately tracked by the Lagrangian description commonly employed in solid mechanics (and also by the ALE description, as commented later), the balance of mass is verified at the global level without explicitly stating it.

Of course, Eq. (5.21a) must also hold at the local level. A common assumption is taking the density ρ a constant. The mass balance then becomes

$$\frac{\partial v_j}{\partial x_j} = 0, \quad (5.22)$$

which is the well-known incompressibility equation. This simplified version of the mass balance is also commonly neglected in solid mechanics. This is due to the fact that *i*) elastic deformations typically induce very small changes in volume and *ii*) plastic deformations are isochoric or volume preserving, (Khan & Huang, 1995). That means that changes in density are negligible, and that Eq. (5.22) automatically holds to sufficient approximation without need of adding it to the set of governing equations.

In conclusion, if the two common assumptions of *i*) uncoupled thermal and mechanical effects and *ii*) constant density are made, then the only conservation law that needs to be solved is the momentum balance (5.21b).

Since the ALE formulation has been chosen, the ALE expression of acceleration, Eq. (5.20c), is employed to represent the inertia forces $\rho \mathbf{a}$ (which are taken into account in the general, i.e. transient, case). Because of this, Eq. (5.21b) contains convective terms associated to the convective velocity \mathbf{c} . These convective terms are similar to those encountered in the Eulerian formulation and reflect the relative motion between mesh and material.

Equation (5.21b) must be complemented with appropriate boundary conditions, which are imposed on the spatial domain. The boundary Γ_x of R_x is composed of two portions Γ_x^g and Γ_x^h with prescribed velocities and tractions respectively:

$$v_i = g_i \quad \text{in } \Gamma_x^g \quad (5.23a)$$

$$\sigma_{ij}n_j = h_i \quad \text{in } \Gamma_x^h. \quad (5.23b)$$

In Eqs. (5.23), \mathbf{g} are prescribed velocities, \mathbf{h} are prescribed tractions and \mathbf{n} is the outward unit normal to Γ_x . A detailed discussion of the implementation of boundary conditions in the ALE formulation can be found in Huerta & Liu (1990) and Huerta & Casadei (1994).

Constitutive equations for nonlinear solid mechanics

In nonlinear solid mechanics, material behaviour is often described by a rate-form constitutive equation

$$\dot{\sigma}^* = f(\mathbf{v}, \boldsymbol{\sigma}), \quad (5.24)$$

relating an objective rate of stress $\dot{\sigma}^*$ to stress, stretching (rate-of-deformation) and velocity, see Malvern (1969) and Appendix B. The material rate of stress, $\dot{\sigma} = (\partial\sigma/\partial t)|_{\mathbf{X}}$ is not employed in Eq. (5.24) to measure the stress variation because it is not an objective tensor. Infinitely many objective rates of stress may be defined; this is typically done by adding to $\dot{\sigma}$ some additional terms that counteract the non-objectivity of $\dot{\sigma}$, so that an objective $\dot{\sigma}^*$ results. After doing so, Eq. (5.24) may be rearranged into

$$\dot{\sigma} = q(\mathbf{v}, \boldsymbol{\sigma}), \quad (5.25)$$

where q contains both f and the terms in $\dot{\sigma}^*$ additional to $\dot{\sigma}$ which ensure the objectivity of $\dot{\sigma}^*$.

In the ALE context, referential time derivatives, not material time derivatives, are employed to represent evolution in time. Specializing the general relationship (5.19) to the stress tensor yields

$$\dot{\sigma} = \frac{\partial\sigma}{\partial t}\Big|_{\mathbf{X}} = \frac{\partial\sigma}{\partial t}\Big|_{\boldsymbol{\chi}} + c_j \frac{\partial\sigma}{\partial x_j}, \quad (5.26)$$

which can be combined with Eq. (5.25) to get a rate-form constitutive equation for ALE nonlinear solid mechanics

$$\frac{\partial\sigma}{\partial t}\Big|_{\boldsymbol{\chi}} + c_j \frac{\partial\sigma}{\partial x_j} = q, \quad (5.27)$$

where, again, a convective term reflects the motion of material particles relative to the mesh.

Remeshing

To define completely the problem posed by Eqs. (5.21b), (5.23) and (5.27), it is necessary to take Eq. (5.13) into account. Since the mesh moves independently from the material, it is necessary to specify the mesh velocity \hat{v} so that the convective velocity c can be determined.

Various remeshing strategies have been proposed, see Donéa (1983), Benson (1989), Huerta & Casadei (1991), Ponthot (1995) and Sarrate (1996). The basic common features are: *i*) boundary nodes are required to remain in the boundary (i.e., they are forced to move with the material along the normal to the surface, with relative motion allowed along the tangent), thus ensuring an accurate tracking of the boundaries; *ii*) mesh distortion is controlled by moving inner nodes in an appropriate way.

In conclusion, if the density is assumed constant and the mass equation is neglected, a transient process is modelled by the ALE (transient) momentum balance (5.21b), and the ALE constitutive equation (5.27), see Box 5.1, complemented with boundary conditions (5.23) and a remeshing strategy to select the mesh velocity.

$$\rho \frac{\partial v_i}{\partial t} \Big|_{\mathcal{X}} + \rho c_j \frac{\partial v_i}{\partial x_j} = \frac{\partial \sigma_{ij}}{\partial x_j} + b_i$$

$$\frac{\partial \sigma}{\partial t} \Big|_{\mathcal{X}} + c_j \frac{\partial \sigma}{\partial x_j} = q$$

Box 5.1 ALE transient analysis

5.2.3 ALE quasistatic analysis

A process is termed **quasistatic** if the inertia forces ρa are negligible with respect to the other forces in the RHS of Eq. (5.21b). In this case, the momentum balance becomes

$$\frac{\partial \sigma_{ij}}{\partial x_j} + b_i = 0, \quad (5.28)$$

that is, a static equilibrium equation where time and velocity play no role.

Since inertia forces have been neglected, the different descriptions of acceleration, Eqs. (5.20), do not appear in Eq. (5.28), which is valid in either the Eulerian or the ALE formulations. In conclusion, there are no convective terms in the ALE momentum balance for quasistatic processes.

A process is called **steady**, on the other hand, if the material velocity \mathbf{v} in every spatial point \mathbf{x} is constant in time. In the Eulerian description, this results in a null local acceleration $(\partial \mathbf{v} / \partial t)|_{\mathbf{x}}$, and only the convective acceleration is present in the momentum balance, which reads

$$(\rho a_i) = \rho v_j \frac{\partial v_i}{\partial x_j} = \frac{\partial \sigma_{ij}}{\partial x_j} + b_i. \quad (5.29)$$

It must be remarked that **quasistatic** and **steady** have clearly different meanings. Nothing is said about the local acceleration in a quasistatic process, which is in general non-null. On the other hand, inertia forces associated to the convective acceleration are in general non-negligible and included in the steady momentum balance, see Eq. (5.29).

There is a further difference between **quasistatic** and **steady**. The local acceleration is assumed to be exactly zero in a steady process. Many flows of practical interest in fluid dynamics are steady, and typically described by a constant spatial velocity field $\mathbf{v}(\mathbf{x})$.

On the contrary, the inertia forces are **not** equaled to zero in a **quasistatic** process, they are just neglected in comparison with other forces. In fact, prescribing the inertia forces to be exactly zero results in a very particular body motion. Indeed, the inertia forces $\rho \mathbf{a}$ only equal zero if the material acceleration \mathbf{a} is null, that is, if the material velocity \mathbf{v} of every material particle \mathbf{X} is constant. This only happens if all the material particles move along parallel straight lines at constant speed.

Remark 5.2: In a quasistatic process, inertia forces are neglected, but this does not imply that the acceleration is assumed to be zero. A process may have relevant, non-null local, convective and total accelerations and still be

correctly modelled as quasistatic, if stress variations and/or body forces are much larger than inertia forces. This is a common situation in solid mechanics (encompassing, for instance, various metal forming processes, see NUMIFORM (1995)).

Remark 5.3: In the ALE context, it is also possible to assume that a process is steady with respect to a grid point χ and neglect the ALE local acceleration $(\partial \mathbf{v} / \partial t)|_{\chi}$, (Ghosh & Kikuchi, 1991). The momentum balance then becomes

$$(\rho a_i) = \rho c_j \frac{\partial v_i}{\partial x_j} = \frac{\partial \sigma_{ij}}{\partial x_j} + b_i \quad (5.30)$$

However, the physical meaning of a null ALE local acceleration is not completely clear, because a grid point holds different material particles and occupies different spatial points at different instants (recall that the mesh moves with respect to the particles and the laboratory).

Summarizing, the momentum balance in ALE quasistatic analysis is the classical equilibrium equation (5.28). Regarding the ALE constitutive equation for quasistatic processes, it should be remarked that the source term \mathbf{q} in Eq. (5.27) measures the stress variation of a certain particle X . Because of this, the convective term $c_j (\partial \sigma / \partial x_j)$ is needed to reflect the fact that the grid point χ is occupied, due to the arbitrary mesh motion, by different particles at different times. This remark is inherent to the ALE kinematics and also applies for quasistatic processes, because the assumption of negligible inertia has no effect on it. In conclusion, **convective terms are present in the ALE constitutive equation for quasistatic processes.**

To conclude: a quasistatic process is modelled by the ALE (quasistatic) momentum balance (5.28), the ALE constitutive equation (5.27), see Box 5.2, complemented with boundary conditions (5.23) and the definition of mesh velocity.

$$\frac{\partial \sigma_{ij}}{\partial x_j} + b_i = 0$$

$$\frac{\partial \sigma}{\partial t} \Big|_{\mathcal{X}} + c_j \frac{\partial \sigma}{\partial x_j} = q$$

Box 5.2 ALE quasistatic analysis

5.2.4 Finite element discretization

The weak form of Eq. (5.21b) is obtained by multiplying by the test functions δv_i and employing the divergence theorem to account for the traction force on the boundary Γ_x^h ,

$$\int_{R_x} \rho \delta v_i \frac{\partial v_i}{\partial t} \Big|_{\mathcal{X}} dR_x + \int_{R_x} \rho \delta v_i c_j \frac{\partial v_i}{\partial x_j} dR_x + \int_{R_x} \frac{\partial \delta v_i}{\partial x_j} \sigma_{ij} dR_x = \int_{R_x} \delta v_i b_i dR_x + \int_{\Gamma_x^h} \delta v_i h_i d\Gamma_x. \quad (5.31)$$

A finite element method proceeds by dividing the domain R_x into elements. When an appropriate set of shape and test functions are chosen to interpolate the velocity, the matrix equation corresponding to Eq. (5.31) is (Liu *et al.*, 1986),

$$M \mathbf{v}' + N \mathbf{v} + \mathbf{f}_{\text{int}} = \mathbf{f}_{\text{ext}} \quad (5.32)$$

where M is the mass matrix, N the convective matrix, \mathbf{v}' the vector of nodal values of referential rate of velocity, \mathbf{v} the velocity vector, \mathbf{f}_{int} the internal force vector and \mathbf{f}_{ext} the external load vector. Equations (5.31) and (5.32) simply state the equilibrium of forces, including the inertia terms caused by acceleration. The relative velocity \mathbf{c} is included in the definition of N .

If the process is assumed to be steady with respect to a grid point, (Ghosh & Kikuchi, 1991), the first term in Eqs. (5.31) and (5.32) is disregarded. However, a more common

and physically meaningful hypothesis (see Remark 5.3) is to neglect all inertia terms, and a quasistatic process is obtained. The resulting weak form is then

$$\int_{R_x} \frac{\partial \delta v_i}{\partial x_j} \sigma_{ij} dR_x = \int_{R_x} \delta v_i b_i dR_x + \int_{\Gamma_x^h} \delta v_i h_i d\Gamma_x, \quad (5.33)$$

which yields, after finite element discretization,

$$f_{\text{int}} = f_{\text{ext}}. \quad (5.34)$$

Equation (5.24) is the standard equilibrium equation of nonlinear mechanics, see Eq. (3.2). As commented above, there are no convective terms in the momentum balance of ALE quasistatic analysis.

5.3 State-of-the-art in ALE stress update

5.3.1 Preliminaries

Due to the semi-discretization (in space) associated to the FEM, Eqs. (5.32) and (5.34) are systems of ordinary differential equations to be solved by numerical time-integration, in conjunction with the ALE constitutive equation (5.27). As previously remarked, the convective term in this equation is needed to represent relative motion between mesh and material, an essential feature of the ALE formulation.

Remark 5.4: The only difference in the ALE constitutive equation for transient and quasistatic cases lies in the meaning of variable t . In a transient analysis, t represents physical time, which appears explicitly in the momentum balance, see Box 5.1. For quasistatic analyses, however, physical time is usually not employed as an independent variable, because it is not present in the momentum balance, Box 5.2. Thus, the variable t in the constitutive equation should be interpreted as the non-physical, pseudo-time parameter typically employed as a load factor in quasistatic simulations.

The methods for the numerical time-integration of Eqs. (5.32) and (5.34) can be classified into **implicit** and **explicit**.

If an implicit time-integration method is chosen, velocities and accelerations are written, by means of difference formulas, as functions of displacements, (Gerardin *et al.*, 1983; Belytschko, 1983). The fundamental unknowns are then the incremental material displacements $\Delta \mathbf{u}$ from one (known) equilibrium configuration Ω_n at time t_n to a new (unknown) equilibrium configuration Ω_{n+1} at time t_{n+1} , a time increment Δt later, see Subsection 4.3.1. It is very important to note that, because of the ALE description, the incremental mesh displacements $\Delta \hat{\mathbf{u}}$ are selected according to a remeshing strategy and are different from $\Delta \mathbf{u}$; this fact must be properly dealt with when checking equilibrium at time t_{n+1} . On the other hand, when explicit time-integration methods are employed, (Belytschko, 1983), accelerations drawn from the momentum balance are integrated to obtain velocities and then displacements. Time is advanced incrementally with no iterations over the time-step.

Whether an implicit or explicit method is chosen, stresses must be updated from Ω_n to Ω_{n+1} to compute the internal forces ${}^{n+1}f_{\text{int}}$, which are required either to check equilibrium (implicit methods, see Chapter 3) or to compute the acceleration at time t_{n+1} (explicit methods).

The time-integration of the ALE constitutive equation (5.27) cannot be performed at the Gauss-point level, as typically done in Lagrangian formulations, because a certain Gauss point is occupied by different particles at times t_n and t_{n+1} . This is reflected by the convective term $c_j \partial \sigma / \partial x_j$ in Eq. (5.27).

In fact, the convective term in the constitutive equation makes the ALE stress update rather more involved than in the Lagrangian case. Various strategies have been proposed to cope with the annoying convective term in Eq. (5.27). Following Benson (1992), they will be classified here into **split** and **unsplit** methods.

If an **unsplit** method is employed, (Liu *et al.*, 1986; Ghosh & Kikuchi, 1991) the fully coupled equations are solved. Coupling refers here to the combination of mate-

rial and convective effects, see Boxes 5.1 and 5.2. Note that in a quasistatic analysis, the convective matrix N of Eq. (5.32) is no longer present and coupling is only due to the constitutive equation (5.27). For transient analysis, on the contrary, both the momentum balance and the constitutive equation exhibit coupling.

Split methods (or fractional-step methods), on the other hand, proceed by performing an operator split on the equations, (Huétink, 1986; Huerta & Casadei, 1991; Baaijens, 1993). The basic idea is to treat material and convective terms separately, by dividing each time-increment into two phases: a material (or Lagrangian) phase, where convective effects in the momentum balance (transient process) and constitutive equation are neglected, and a convection (or transport) phase.

Any combination of explicit/implicit, split/unsplit may be made. This point will be illustrated in the next Subsection, where some choices in the literature are reviewed.

5.3.2 Literature review

Implicit split ALE formulations

An implicit split ALE formulation is employed in Huétink (1986) and Huétink *et al.* (1990) to model quasistatic metal forming processes. The material displacement increment $\Delta \mathbf{u}$ is computed exactly as in the Updated Lagrangian method, (Bathe, 1982). The mesh displacement increment $\Delta \hat{\mathbf{u}}$ is selected following a remeshing strategy.

The Lagrangian stress update involves the time-integration of the material term \mathbf{q} in Eq. (5.27). This process may be symbolically expressed as

$${}^{n+1}\boldsymbol{\sigma}(\mathbf{X}) = {}^n\boldsymbol{\sigma}(\mathbf{X}) + \Delta\boldsymbol{\sigma}_m, \quad (5.35)$$

where the subscript m emphasizes that $\Delta\boldsymbol{\sigma}_m$ is the stress increment of a material particle \mathbf{X} . After that, the convective term $c_j\partial\boldsymbol{\sigma}/\partial x_j$ has to be integrated so that ${}^{n+1}\boldsymbol{\sigma}(\boldsymbol{\chi})$, the stress in configuration Ω_{n+1} of a certain grid point $\boldsymbol{\chi}$, is obtained.

The stress gradient $\partial\boldsymbol{\sigma}/\partial x_j$ is especially troublesome, because it cannot be computed at element level. In finite elements with linear interpolation of the displacements, for

instance, the element stress gradient vanishes because the stress is a discontinuous, element-wise constant field.

The method proceeds by regularizing this discontinuous field, following an interpolation procedure typical of postprocessing and error estimation techniques. First, local smoothing (least-squares interpolation at element level) provides nodal values of stresses. Then, mean nodal values are employed to define, via the shape functions, a continuous stress field ${}^n\hat{\sigma}$. The stress update is then

$${}^{n+1}\sigma(\chi) = {}^n\sigma(\chi) + \Delta\sigma_m - (\Delta u_j - \Delta \hat{u}_j) \frac{\partial {}^n\hat{\sigma}}{\partial x_j}, \quad (5.36)$$

where the continuous stress ${}^n\hat{\sigma}$ enables the gradient determination. Note that, contrary to Eq. (5.35), Eq. (5.36) provides the new stress in a certain grid point χ .

This algorithm is prone to numerical instabilities in some cases, that can be solved by a process of global smoothing: not only the gradient, but also the stress at time t_n is evaluated with the regularized, continuous stress ${}^n\hat{\sigma}$, instead of with the known discontinuous stress ${}^n\sigma$. The algorithm is then

$${}^{n+1}\sigma(\chi) = {}^n\hat{\sigma}(\chi) + \Delta\sigma_m - (\Delta u_j - \Delta \hat{u}_j) \frac{\partial {}^n\hat{\sigma}}{\partial x_j} \quad (5.37)$$

This global smoothing suppresses the instabilities, but tends to be excessively diffusive. The method finally adopts a weighted global smoothing: the stress at time t_n is a weighted sum of the discontinuous and continuous stress fields. This results in

$${}^{n+1}\sigma(\chi) = (1 - \beta) {}^n\hat{\sigma}(\chi) + \beta {}^n\sigma(\chi) + \Delta\sigma_m - (\Delta u_j - \Delta \hat{u}_j) \frac{\partial {}^n\hat{\sigma}}{\partial x_j} \quad (5.38)$$

where β is the weight parameter, which is proportional to the ratio of the relative displacement between grid and material point to the element size. This parameter is estimated by numerical experiments.

A more sophisticated approach is that of Baaijens (1993). Profiting from the body of knowledge in fluid mechanics, a Time-Discontinuous/Galerkin-Least-Squares formulation is employed to handle the convective parts in the governing set of equations.

A comparison is made between the standard split ALE formulation, where convection is performed after the Lagrangian step in every iteration within the time-increment, and the Updated-ALE formulation, where the iterative process is carried on a purely Lagrangian fashion and the convection is performed just once, at the end of the time-step. The conclusion is that, for a certain type of problems, this latter strategy makes computation considerably cheaper while the decrease in accuracy is minimum.

Explicit split ALE formulations

An example of explicit split formulation may be found in Huerta & Casadei (1991). In the context of fast transient dynamics, they employ a centered second-order scheme for the Lagrangian phase, classical of the Lagrangian formulation of such problems. After that, the convection phase is treated in two different ways: a Lax-Wendroff technique and a Godunov-type technique. These two techniques have been employed in this work in the context of quasistatic analysis, and are discussed in detail in Subsections 5.4.2 and 5.4.3 respectively.

Implicit unsplit ALE formulations

Ghosh & Kikuchi (1991) employ an implicit unsplit ALE formulation. As mentioned in Remark 5.3, the process is assumed to be steady with respect to a grid point; the momentum balance is then

$$Nv + f_{\text{int}} = f_{\text{ext}} \quad (5.39)$$

and the convective matrix N results in the non-symmetry of the tangent stiffness matrix K .

Since the formulation is unsplit, there is no separation between Lagrangian and convection phases. The momentum balance is verified by employing the referential mesh in the midstep configuration $\Omega_{n+\frac{1}{2}}$, so the computed material displacement increments Δu correspond to the material particles X that occupy the grid points χ at time $t_{n+\frac{1}{2}}$. The fact that these particles are not the same that the grid points holded at time t_n is

carefully taken into account: through a process of particle tracing, the location of the material particles at the beginning of the step is found.

The stress ${}^n\sigma$ of these particles is then found by interpolating the Gauss-point values. The integration of the rate constitutive equation yields the material stress increment $\Delta\sigma_m$. The stress at the midstep configuration is then simply computed as

$${}^{n+1/2}\sigma(X) = {}^n\sigma(X) + \frac{1}{2}\Delta\sigma_m \quad (5.40)$$

where m emphasizes, as in Eq. (5.35), that the various quantities refer to the same material particles, not the same grid points. Finally, the stresses are updated to the grid configuration at t_{n+1} using the fundamental ALE relation, Eq. (5.19).

Explicit unsplit ALE formulations

To end this review, the explicit unsplit ALE formulation of Liu *et al.* (1986) will be presented. To account for all inertia effects, Eq. (5.21b) is employed as momentum balance. The treatment of the stress gradient in the constitutive equation is based in the idea of the definition of the stress-velocity product,

$$Y = \tau c, \quad (5.41)$$

where τ is a component of the stress tensor σ . With the help of Y , the ALE constitutive equation (5.27) is rewritten as an equation for every component τ of σ and q of tensor q ,

$$\frac{\partial \tau}{\partial t} + \frac{\partial Y_j}{\partial x_j} = q + \tau \frac{\partial c_j}{\partial x_j}. \quad (5.42)$$

The weak forms of Eqs. (5.41) and (5.42) are obtained with the help of appropriate test functions. After finite element discretization, the corresponding matrix equations are

$$M^\sigma \sigma' + G^T Y - D\sigma = q \quad (5.43)$$

$$M^Y Y = N^Y \sigma \quad (5.44)$$

where M^σ and M^Y are the generalized mass matrices for the corresponding variables, N^Y the generalized convection matrix, G is the divergence operator matrix, D is the generalized diffusion matrix, and σ' (referential rate of stress), σ , Y and q represent nodal vectors instead of fields.

Equation (5.44) may be eliminated by incorporating it into Eq. (5.43),

$$M^\sigma \sigma' + G^T (M^Y)^{-1} N^Y \sigma - D\sigma = q \quad (5.45)$$

Equations (5.32) and (5.45) are integrated by an explicit predictor-corrector method. Lumped mass matrices are used to enhance the computational efficiency.

5.4 ALE stress update in quasistatic analysis

5.4.1 Preliminaries

Choice of the strategy for ALE stress update

It can be seen in Boxes 5.1 and 5.2 (Section 5.2) that the ALE momentum balance for transient processes, Eq. (5.21b), and for quasistatic processes, Eq. (5.28), are different: the former accounts for inertia forces, while the latter does not (a similar situation is found in the Lagrangian analysis of transient and quasistatic processes). Because of this, radically different techniques are employed for its time-integration. The transient momentum balance is typically handled, both in the Lagrangian and ALE formulations, by an **explicit**, velocity-based scheme, see Halleux & Casadei (1987) and Pijaudier-Cabot *et al.* (1995), where stability requirements impose a tight upper bound on the time-step Δt . The quasistatic momentum balance, on the other hand, is commonly time-integrated with an **implicit**, incremental-iterative, displacement-based algorithm, which allows for larger time-steps.

Regarding the ALE constitutive equation (5.27), it is valid for both transient and quasistatic processes, see Boxes 5.1 and 5.2. Since the constitutive equation does not

change, it is a natural requirement that the numerical schemes employed for the stress update apply to both transient and quasistatic analyses. It is also reasonable to demand that these schemes be explicit. Indeed, in the transient case an explicit stress update is needed to be consistent with the explicit time-integration of the momentum balance: an implicit stress update procedure to be used in every time-step within an hydrocode, (Benson, 1992), would be completely unaffordable. As for the quasistatic case, the stress update should be cheap –which, in practice, means explicit–, if the ALE formulation is to be competitive with alternative techniques, such as a Lagrangian formulation combined with an adaptive remeshing strategy, (Peraire *et al.*, 1991).

In exchange for a certain loss of accuracy, split methods offer a generic benefit: the original equation is split into simpler equations, and that means simpler, more robust algorithms, specifically designed for each equation. Regarding the ALE equations, an added advantage results: profit is taken from the experience in solid mechanics and fluid dynamics in handling material and convective terms respectively.

The simplification associated to a split approach is especially interesting for the ALE constitutive equation. Indeed, the stress field σ is typically discontinuous across element edges, so its gradient cannot be reliably computed at the element level. Moreover, stress values are known at the Gauss points —where they are needed to compute internal forces—, not at the nodes. These two difficulties are easier to circumvent with a splitting technique.

This splitting technique may be regarded as the natural choice for ALE quasistatic analysis: since the momentum balance has no convective term, this equation is completely handled by the material phase, so the transport phase need only deal with the convective term in the constitutive equation. The implementation of ALE capabilities in a Lagrangian code for quasistatic analysis is then a straightforward matter, see Rodríguez-Ferran & Huerta (1995).

The material (or Lagrangian) phase

In the first phase within every time-step, the convective terms are neglected and only

material effects are accounted for. The constitutive equation then reads

$$\frac{\partial \sigma}{\partial t} \Big|_{\chi} = q, \quad (5.46)$$

and it must be integrated in time to update stresses from ${}^n\sigma$ (stress at time t_n) to ${}^L\sigma$ (stress after the Lagrangian phase). Neglecting the convective terms is equivalent to assuming that grid points χ move attached to material particles X . Because of this, the Lagrangian phase can be performed with the same stress update algorithms used in Lagrangian simulations, which handle the constitutive equation at the Gauss-point level. In this work, the two algorithms discussed in Chapter 4 have been employed for the Lagrangian phase of the ALE stress update.

The convection (or transport) phase

The time-step is completed with a convection phase, which handles the convective terms not taken into account during the material phase. The constitutive equation is then

$$\frac{\partial \sigma}{\partial t} \Big|_{\chi} + c_j \frac{\partial \sigma}{\partial x_j} = 0, \quad (5.47)$$

and its time-integration is required to update stresses from ${}^L\sigma$ to ${}^{n+1}\sigma$ (i.e., the stress at time t_{n+1}).

Remark 5.5: The fractional-step strategy has resulted in the splitting of the constitutive equation (5.27) into the parabolic equation (5.46) of the material phase and the hyperbolic equation (5.47) of the convection phase. This enables the use of specific algorithms, specially designed to handle each type of equation.

Since it contains a convective term, all the known numerical difficulties associated to convection (typically encountered in computational fluid mechanics) will appear when handling Eq. (5.47). It must be remarked, however, that the main source of trouble in the convective term of Eq. (5.47) is the stress gradient, not the convective velocity itself.

Indeed, the convective velocity \mathbf{c} is usually moderate (in practice, the mesh velocity $\hat{\mathbf{v}}$ is often selected by modifying the material velocity \mathbf{v} in order to reduce element distortion). The stress gradient, on the contrary, is a troublesome term, because it cannot be computed properly at the element level.

A possible solution to this problem is that of Huétink (1986), see Subsection 5.3.2. The main disadvantage of this approach is that it results in an implicit algorithm (for the least-squares interpolation), which may be acceptable for quasistatic analyses (although explicit schemes are preferred, as commented previously) but not for transient analyses, where the momentum balance is treated explicitly. More recently, various explicit procedures for the convection phase of the ALE stress update have appeared, see Huerta & Casadei (1991), Akkerman *et al.* (1995).

In this work, three explicit algorithms to integrate Eq. (5.47) will be presented and discussed (Huerta & Casadei, 1991; Rodríguez-Ferran & Huerta, 1995; Huerta *et al.*, 1995). To avoid computing gradients of the discontinuous stress field at the element level, two different approaches have been taken: either *i*) use an **explicit** smoothing procedure (Lax-Wendroff update) or *ii*) employ algorithms that circumvent the computation of the stress gradient (Godunov-like technique and simple interpolation procedure). In any case, the starting point is noting that Eq. (5.47) contains a scalar equation for every stress component τ

$$\frac{\partial \tau}{\partial t} \Big|_{\chi} + c_j \frac{\partial \tau}{\partial x_j} = 0. \quad (5.48)$$

In fact, the internal variables commonly employed in nonlinear mechanics must also be convected following Eq. (5.48), so τ can be any **stress-related component** (a component of the stress tensor or an internal variable).

Remark 5.6: The term stress-related component does not imply that only stress measures are employed as internal variables. In isotropic plasticity, for instance, it is common to choose the equivalent plastic strain as an internal variable, see Khan & Huang (1995). Stress-related component is just a convenient way to refer to all the variables that must be updated according to Eq. (5.48).

5.4.2 Algorithm 1: Lax-Wendroff update

As classically done in Lax-Wendroff or Taylor-Galerkin schemes, (Lax & Wendroff, 1960 and 1964; Donéa, 1984; Donéa & Quartapelle, 1992), a Taylor series expansion of Eq. (5.48) is followed by substitution of the original equation into the expansion. This yields

$${}^{n+1}\tau = L_\tau - \Delta t \, {}^{n+\frac{1}{2}}c_j \frac{\partial L_\tau}{\partial x_j} + \frac{\Delta t^2}{2} \, {}^{n+\frac{1}{2}}c_i \, {}^{n+\frac{1}{2}}c_j \frac{\partial}{\partial x_i} \left(\frac{\partial L_\tau}{\partial x_j} \right), \quad (5.49)$$

where ${}^{n+\frac{1}{2}}c$ is the convective velocity evaluated at the midstep. For a transient analysis, ${}^{n+\frac{1}{2}}c$ is computed directly as the difference of material and mesh velocity,

$${}^{n+\frac{1}{2}}c = {}^{n+\frac{1}{2}}v - {}^{n+\frac{1}{2}}\hat{v}, \quad (5.50)$$

where ${}^{n+\frac{1}{2}}v$ is obtained in the time-integration of the momentum balance and ${}^{n+\frac{1}{2}}\hat{v}$ is provided by the ALE remeshing procedure, see Halleux & Casadei (1987) and Pijaudier-Cabot *et al.* (1995). For a quasistatic analysis, on the other hand, solving the momentum balance yields an increment of material displacements Δu and the ALE remeshing an increment of mesh displacements $\Delta \hat{u}$. The convective velocity, which is assumed to be constant within the time-step, is then computed as

$$c = \frac{\Delta u - \Delta \hat{u}}{\Delta t}. \quad (5.51)$$

In Eq. (5.49), both the stress gradient, which will be denoted by γ ($\gamma_i = \partial L_\tau / \partial x_i$), and its spatial derivatives are required. An explicit interpolation procedure, based originally in a classical least-squares projection, is employed to obtain a smoothed field of stress gradient, (Huerta & Casadei, 1991). The starting point is the integral relation

$$\int_V f_1 \nabla f_2 dV = - \int_V f_2 \nabla f_1 dV + \int_{S=\partial V} f_1 f_2 \mathbf{n}_x dS, \quad (5.52)$$

where f_1 and f_2 are scalar functions defined on volume V , \mathbf{n}_x is the outward unit normal to the boundary S of volume V , and ∇ denotes the gradient operator. For two-dimensional problems, Eq. (5.52) reduces to

$$\int_\Omega f_1 \nabla f_2 d\Omega = - \int_\Omega f_2 \nabla f_1 d\Omega + \int_{\Gamma=\partial\Omega} f_1 f_2 \mathbf{n}_x d\Gamma, \quad (5.53)$$

where Ω is the two-dimensional domain associated to volume V , and $d\Gamma$ is its boundary. Equation (5.53) is valid both for plane stress/strain and axisymmetrical problems.

Equation (5.53) can be specialized into

$$\int_{R_x^e} N_a \gamma dV = - \int_{R_x^e} \tau \nabla N_a dV + \int_{\partial R_x^e} N_a \tau \mathbf{n}_x dS, \quad (5.54)$$

where R_x^e and ∂R_x^e represent element e and its boundary, \mathbf{n}_x is the outward unit normal in the current configuration and N_a is the shape function of node a . The assembly of Eq. (5.54) results in the linear set of equations

$$M \Gamma = \Phi, \quad (5.55)$$

where M is a consistent pseudo-mass matrix, Γ is a vector of nodal smoothed values of the stress gradient, and the independent vector $\Phi = [\Phi_a]$ is defined as

$$\Phi_a = \sum_e \left[- \int_{R_x^e} \tau \nabla N_a dV + \int_{\partial R_x^e} N_a \tau \mathbf{n}_x dS \right]. \quad (5.56)$$

To compute the integrals along element boundaries ∂R_x^e , the scalar field τ is extrapolated from Gauss points to nodes with the aid of Gauss-point shape functions.

Since an explicit procedure is sought, the consistent matrix M of Eq. (5.55) is substituted by the lumped matrix $M^{lp} = [M_a^{lp}]$, with

$$[M_a^{lp}] = \sum_e \int_{R_x^e} N_a dV \quad (5.57)$$

After doing so, Γ is explicitly computed by solving a trivial system of equations, with diagonal matrix M^{lp} . Then Eq. (5.49) can be solved. Since ${}^{n+1}\tau$ is required at the Gauss points, a collocation technique is used to handle the weak form of Eq. (5.49). This results in

$${}^{n+1}\tau(\xi) = {}^L\tau(\xi) - \Delta t \sum_j c_j(\xi) \gamma_j(\xi) + \frac{\Delta t^2}{2} \sum_i c_i(\xi) \sum_j c_j(\xi) \frac{\partial \gamma_j}{\partial x_i}(\xi) \quad (5.58)$$

for each Gauss point ξ .

A special treatment is required for symmetric boundary conditions. Since the stress-related component τ is symmetric with respect to an axis or plane of symmetry, the normal component of its gradient γ must be set to zero. Since the smoothed nodal gradients are obtained by solving a diagonal set of equations, the cancellation of a certain component γ_i is equivalent to the cancellation of the corresponding independent term Φ_i , prior to the solution of the trivial system. This is the approach taken in the algorithm.

A flowchart of the Lax-Wendroff update technique is presented in Box 5.3.

- 1.— Update the mesh configuration from t_n to t_{n+1} taking into account the mesh velocity
 - 2.— Compute convective velocity c according to Eq. (5.50) (transient processes) or Eq. (5.51) (quasistatic processes)
 - 3.— Compute the lumped mass matrix M^{lp} , Eq. (5.57)
- FOR EVERY STRESS-RELATED COMPONENT τ :
- 4.— Compute the independent vector Φ , Eq. (5.56)
 - 5.— Treatment of symmetry: for nodes in an axis/plane of symmetry, set to zero the normal component of Φ_a
 - 6.— Solve the trivial linear set $M^{lp}\Gamma = \Phi$ (diagonal matrix M^{lp}) to get the vector of smoothed nodal gradients Γ
 - 7.— Interpolate stress gradient γ and its spatial derivatives from nodes to Gauss points with the aid of shape functions
 - 8.— Employ the one-step Lax-Wendroff method to find ${}^{n+1}\tau$ at the Gauss-point level, Eq. (5.58)

Box 5.3 Lax-Wendroff stress update

5.4.3 Algorithm 2: Godunov-like update

The second algorithm for the convection phase is based on Godunov's method for conservation laws, (Le Veque, 1990). With the help of the stress-velocity product $\mathbf{Y} = \tau \mathbf{c}$, Eq. (5.41), Eq. (5.47) is rewritten as the conservation equation

$$\frac{\partial \tau}{\partial t} \Big|_{\chi} + \frac{\partial Y_j}{\partial x_j} = \tau \frac{\partial c_j}{\partial x_j}, \quad (5.59)$$

which is similar to Eq. (5.42) but without the material term q , because only convective effects are considered now.

One-point quadratures

Godunov's method, which was developed in the context of finite volumes, assumes a piecewise constant field τ , (Le Veque, 1990). In a finite element framework, this is the situation if one-point quadratures are employed; Godunov's method is then directly applicable. However, to allow for a subsequent generalization to multiple-point quadratures, a residual weak formulation of the method is preferred, (Huerta & Casadei, 1991). Since the test functions ω are also piecewise constant, the integral equation is valid at the element level:

$$\int_{R_x^e} \omega \frac{\partial \tau}{\partial t} dV = \int_{R_x^e} \omega \tau \frac{\partial c_j}{\partial x_j} dV - \int_{\partial R_x^e} \omega (\mathbf{Y} \cdot \mathbf{n}_x) dS. \quad (5.60)$$

Since both τ and ω are constant within finite element e , Eq. (5.60) results in

$$\frac{\partial \tau}{\partial t} = -\frac{1}{2V} \sum_{s=1}^{N_s} f_s (\tau_s^c - \tau) [1 - \text{sign}(f_s)], \quad (5.61)$$

where τ is the stress-related component of element e , which has volume V and N_s faces, τ_s^c is the stress-related component in the contiguous element across face s , and f_s is the flux of convective velocity \mathbf{c} across face s , $f_s = \int_s \mathbf{c} \cdot \mathbf{n}_x dS$.

For two-dimensional problems, Eq. (5.61) can be simplified into

$$\frac{\partial \tau}{\partial t} = -\frac{1}{2A} \sum_{\Gamma=1}^{N_\Gamma} f_\Gamma (\tau_\Gamma^c - \tau) [1 - \text{sign}(f_\Gamma)], \quad (5.62)$$

where N_Γ is the number of edges of element e and τ_Γ^c is the stress-related component in the contiguous element across edge Γ . For plane strain and plane stress problems, A is the area of element e and f_Γ is the flux across edge Γ , $f_\Gamma = \int_\Gamma \mathbf{c} \cdot \mathbf{n}_x d\Gamma$. For axisymmetric problems, on the other hand, axisymmetry is accounted for by taking $A = \int_{R_x^e} r dS$ and $f_\Gamma = \int_\Gamma (\mathbf{c} \cdot \mathbf{n}_x) r d\Gamma$, where r is the radial coordinate.

Explicit time-integration of Eq. (5.62) yields

$${}^{n+1}\tau = L_\tau - \frac{\Delta t}{2A} \sum_{\Gamma=1}^{N_\Gamma} f_\Gamma (\tau_\Gamma^c - \tau) [1 - \text{sign}(f_\Gamma)] \quad (5.63)$$

for each element.

Multiple-point quadratures

Two different strategies have been tested to extend the Godunov-like update to multiple-point quadratures, (Huerta *et al.*, 1995; Casadei *et al.*, 1995). The first approach is a generalization of the residual formulation of Eq. (5.60) which takes into account that the stress-related component τ and the test function ω are no longer piecewise constant. The second one is a simpler, more straightforward technique which is directly based on the algorithm for one-point quadratures just presented.

If the first strategy is chosen, τ and ω are interpolated via Gauss-point shape functions L_i , which equal 1 for the i -th Gauss point and zero for the other Gauss points. If quadrilaterals with 2×2 integration points are employed, for instance, these shape functions are

$$L_i(\xi, \eta) = \frac{1}{4} \left(1 + \frac{\xi}{\xi_i} \right) \left(1 + \frac{\eta}{\eta_i} \right), \quad (5.64)$$

where ξ and η are the natural coordinates ranging from -1 to 1 and (ξ_i, η_i) are the coordinates of the i -th Gauss point.

A Galerkin formulation of Eq. (5.60) followed by numerical time-integration results in a set of linear equations for every element,

$$M_{ij} {}^{n+1}\tau = M_{ij} L_\tau + \Delta t [C_{ij} L_\tau - S_i] \quad i, j = 1, \dots, N_G. \quad (5.65)$$

In Eq. (5.65), N_G is the number of Gauss points in the quadrature, M is a pseudo-mass matrix, C is a matrix accounting for spatial variations of the stress-velocity product and S is a vector associated to the flux integral, see Huerta *et al.* (1995).

It must be remarked that the pseudo-mass matrix in Eq. (5.65) is diagonal. Indeed, since it is computed as

$$M_{ij} = \int_{R_x} L_i L_j dV \quad (5.66)$$

and because of the choice of shape functions L_i , Eq. (5.64), off-diagonal terms are null. Because of this, an explicit algorithm for the stress update is obtained.

The second approach is an engineering-like extension of the algorithm for piecewise constant fields. The basic idea is to divide every finite element into N_G subelements, each of them corresponding to the influence domain of a Gauss point, (Huerta *et al.*, 1995). If quadrilaterals with 2×2 integration points are employed, for instance, every element is divided into four subelements. In every subelement, τ is assumed to be constant, and represented by the Gauss-point value. Because of this, τ is a piecewise constant field with respect to the mesh of subelements, and Eq. (5.63) can be employed to update the value of τ for every subelement. A very simple and efficient algorithm is obtained, because the submeshing of the original mesh into subelements need only be performed once, at the beginning of the analysis.

This second technique, which has shown better results than the first one, is reflected in Box 5.4.

- 1.- Update the mesh configuration from t_n to t_{n+1} taking into account the mesh velocity
- 2.- Compute convective velocity \mathbf{c} according to Eq. (5.50) (transient processes) or Eq. (5.51) (quasistatic processes)
- 3.- Transfer convective velocity from original mesh to auxiliary mesh of subelements

FOR EVERY SUBELEMENT:

- 4.- Compute area A

LOOP ON SIDES OF THE SUBELEMENT

- 5.- Compute the flux of convective velocity across side, f_s

LOOP ON STRESS-RELATED COMPONENTS

- 6.- Compute jump across side, $(\tau_s^c - \tau)$, and add up the RHS of Eq. (5.62)

END OF LOOP ON STRESS-RELATED COMPONENTS

END OF LOOP ON SIDES OF THE SUBELEMENT

- 7.- Employ Godunov's method to find ${}^{n+1}\tau$ at the subelement level, Eq. (5.63)

Box 5.4 Godunov-like stress update

5.4.4 Algorithm 3: simple interpolation procedure

The computation of stress gradients may be circumvented by recalling the relationship between material and referential time derivatives, Eq. (5.19), and by rewriting the equation of stress transport (5.47) simply as

$$\frac{\partial \tau}{\partial t} \Big|_{\mathbf{X}} = 0. \quad (5.67)$$

The splitting strategy represented by Eqs. (5.46) and (5.67) is amenable to the following geometrical interpretation, (Benson, 1989): the mesh velocity is assumed to equal

the material velocity –thus resulting in null convective effects– during the Lagrangian phase, which yields stress-related components ${}^L\tau$ in a fictitious, Lagrangian mesh; in the convection phase, the material is fixed in space and the mesh is allowed to move to its final configuration, so τ must be remapped from the fictitious Lagrangian mesh to the true ALE mesh. From this viewpoint, Eq. (5.67) states that the value of τ of a material particle X , which is fixed in a spatial point \mathbf{x} , remains unaffected by the motion of grid point χ .

The remapping from the fictitious to the ALE mesh can be performed by means of a simple interpolation procedure similar to those typically employed in adaptive remeshing. For every Gauss point in the ALE mesh, the updated value ${}^{n+1}\tau$ is computed in two steps: 1) detect the element in the fictitious mesh containing the ALE Gauss point and 2) interpolate τ from the Lagrangian element to the ALE Gauss point with the aid of shape functions, (Rodríguez-Ferran & Huerta, 1995).

This is a very general algorithm, because it does not require the old and new meshes to have the same topology. This generality makes the algorithm relatively expensive, especially the element search of step 1). However, the location of every new Gauss point in the old mesh can be simplified by taking profit from the fact that the ALE remeshing is topology-conserving. The key idea is that each finite element has the same neighbour elements throughout the analysis.

The basic steps of the interpolation procedure are shown in Box 5.5.

- 1.— Produce a list of all the neighbour elements (i.e., sharing at least one node) of every element (Important: only once, for the convection phase of the first time-increment)
- 2.— Compute convective velocity \mathbf{c} according to Eq. (5.50) (transient processes) or Eq. (5.51) (quasistatic processes)
- 3.— Obtain the (fictitious) Lagrangian mesh with the aid of the material velocity
- 4.— Update the coordinates of the (true) ALE Gauss points from t_n to t_{n+1} taking into account the mesh velocity

FOR EVERY ELEMENT OF THE ALE MESH:

LOOP ON GAUSS POINTS OF THE ELEMENT

- 5.— Locate the ALE Gauss point in the Lagrangian mesh. Restrict the search to the element under consideration and its neighbours
- 6.— Compute the natural coordinates (ξ, η) of the ALE Gauss point by solving the nonlinear set of order 2

$$x_{gp} = \sum_{i=1}^{nnod} x_i N_i(\xi, \eta) \quad ; \quad y_{gp} = \sum_{i=1}^{nnod} y_i N_i(\xi, \eta)$$

where x_{gp} and y_{gp} are the (known) Cartesian coordinates of the Gauss point, (x_i, y_i) the nodal coordinates of the Lagrangian element, N_i the shape functions and $nnod$ the number of nodes.

LOOP ON STRESS-RELATED COMPONENTS

- 7.— Compute ${}^{n+1}\tau$ by interpolating, at element level, the Gauss-point values in the Lagrangian element to the ALE Gauss point

END OF LOOP ON STRESS-RELATED COMPONENTS

END OF LOOP ON GAUSS POINTS

5.4.5 Limitations on the time-step

The numerical time-integration of the momentum balance places a limit on the maximum time-step Δt . For transient analysis, this limitation is associated to the stability of the explicit algorithm employed to integrate Eq. (5.21b). For quasistatic analysis, on the other hand, Δt is restricted by the convergence properties of the nonlinear solver employed for the implicit integration of Eq. (5.28).

According to the numerical experimentation of this work, the limitation on time-step associated to the numerical time-integration of the constitutive equation is **not** more restrictive than for the momentum balance, in either transient or quasistatic analysis. In other words, the same Δt that is used for the momentum balance can then be employed for the stress update, with any of the algorithms of the previous Subsections. The basic idea is that, with the typical values of time-step Δt and convective velocity \mathbf{c} , the relative motion between the material and the mesh that occurs during the convection phase carries material particles to a neighbour element (at most), but no further. This situation can be symbolically expressed as

$$|\mathbf{c}|\Delta t \leq h, \quad (5.68)$$

where h is the element size. In fact, Eq. (5.68) can be regarded as the well-known Courant condition. Since the three algorithms previously presented take into account the stress transport between contiguous elements, they can be used for the stress update in both transient and quasistatic processes.

It must be remarked, then, that the **explicit** algorithms for the convection phase of Subsections 5.4.2, 5.4.3 and 5.4.4 are applicable in combination with either an explicit algorithm for the momentum balance (transient analysis) or an implicit algorithm (quasistatic analysis). In the latter case, there is no need to develop different, implicit stress update procedures for the convection phase.

If the time-integration of the momentum balance is explicit (transient analysis), no iterations are made. Thus, every time-step involves performing each of the following tasks once: **1)** material phase of the stress update; **2)** selection of mesh velocity (i.e., remeshing); **3)** convection phase of the stress update.

On the contrary, if the momentum balance is handled by an implicit algorithm (quasistatic analysis), iterations up to equilibrium are required. This means that tasks 1), 2) and 3) must be performed once per iteration, until equilibrium in the ALE mesh is achieved.

An alternative, simplified approach, however, is keeping only task 1) (material phase of the stress update) inside the iteration loop. By doing so, iterations are carried out until equilibrium is verified in the fictitious Lagrangian mesh. After that, tasks 2) and 3) are performed just once, at the end of the time increment, see Box 5.6. This approach has the advantage that the only overhead cost of ALE quasistatic analysis with respect to a Lagrangian analysis is due to the remeshing and the **explicit** convection phase required at the end of every time-step. This overhead cost is acceptable when compared to the cost of the **implicit** time-integration of the equilibrium equation. It must be noted, on the other hand, that with this approach equilibrium is only achieved in the fictitious Lagrangian mesh, not in the true ALE mesh. Indeed, the convection phase somewhat disrupts equilibrium. Numerical experimentation, however, shows that the spurious residual forces associated to the convection phase are small and do not affect the quality of the solution, see next Section and (Baaijens, 1993).

- 1.— **Iteration loop:** neglect the convective velocity and solve the equilibrium equation (5.34) complemented with the Lagrangian constitutive equation (5.46) iteratively, as in a Lagrangian analysis
- 2.— **Remeshing:** compute the convective velocity \mathbf{c} and update the mesh from its fictitious Lagrangian configuration to its true ALE configuration
- 3.— **Convection:** perform the convection phase of the stress update, Eq. (5.47), with any of the algorithms of the previous Subsections

Box 5.6 Simplified approach for ALE quasistatic analysis

5.5 Numerical examples

5.5.1 Test HILL

The explicit algorithms for the convection phase are compared with the aid of a simple test, see Figure 5.1. The mesh is formed of 10×10 unit eight-noded squares, with 2×2 Gauss points. The initial stress field is a cosinoidal hill of radius 3 centered in point P . To assess the 2-D behaviour of the algorithms, a uniform mesh velocity is prescribed in the direction of diagonal AC , Figure 5.1. Material velocity is null, so only convective effects are present (i.e., Eq. (5.48) must be solved). The time-step is set to $\Delta t = 0.25$.

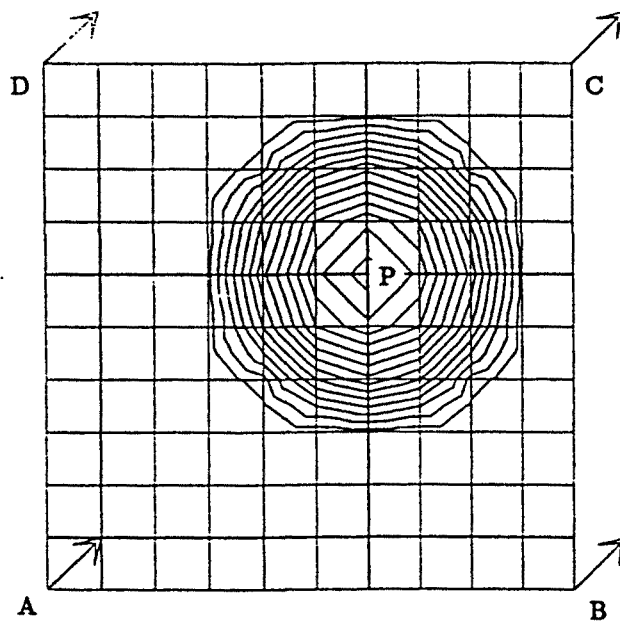


Figure 5.1 Test HILL. Mesh, initial stress field and mesh velocity

Figure 5.2 shows the stress field at $t = 2$. For the interpolation procedure, the initial circular shape of the hill is well maintained, Figure 5.2a. For the Godunov-type update, on the contrary, the diameter normal to velocity direction is quite larger than in

the input field, Figure 5.2b. This is due to the lack of “corner convection” in the Godunov update: since convection is accounted for as fluxes through the element edges, Eq. (5.63), there is no stress transport between two elements that only share a corner node. This leads to the crosswind diffusion that alters the hill shape, and makes this algorithm more diffusive than the interpolation procedure. Figure 5.3 shows the stress profile along diagonal AC at $t = 2$ corresponding to the exact solution (identical to the initial profile, since there are no material effects), and to the two algorithms. The Godunov-type update shows a 13.5% reduction of the peak stress value, compared to only 3.6% reduction for the interpolation procedure.

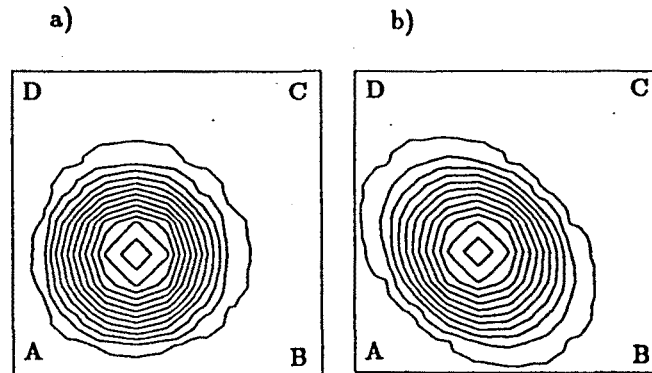


Figure 5.2 Test HILL. Stress field at $t = 2$. a) Interpolation procedure. b) Godunov update.

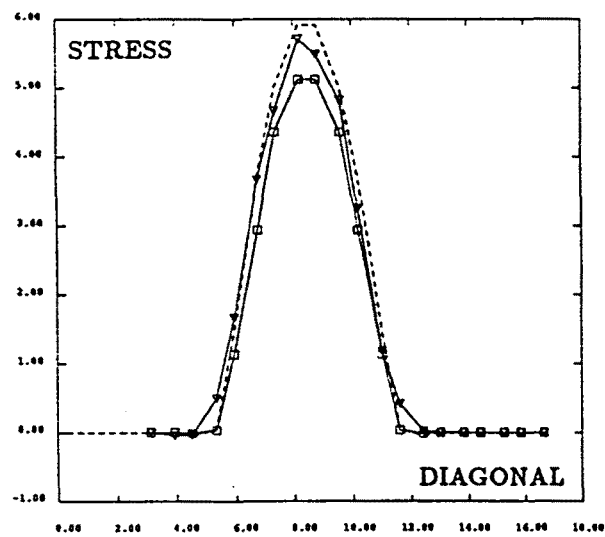


Figure 5.3 Test HILL. Stress profiles along diagonal AC: exact solution (dashed), interpolation procedure (triangles), Godunov update (squares).

5.5.2 Test NECKING

The necking test of Subsection 4.5.7 (Simo, 1988) is employed here to test the full ALE stress update algorithms (i.e. material and convection phases). Two meshes of eight-noded quadrilaterals with 2×2 Gauss points are employed for the analysis, a coarse mesh of 50 elements and a fine mesh of 320 elements. Both Lagrangian and ALE simulations have been carried out, for comparison purposes. For the ALE cases, the simplified approach of Box 5.6 has been employed.

Figure 5.4 shows the whole deformed piece after an axial elongation of 14 mm (7 mm for half the piece, or 26% of initial length). If the Lagrangian formulation is employed in combination with the coarse mesh, Figure 5.4a, the elements in the neck zone become very distorted, following the large material deformation. As a consequence, a poor definition of the deformed shape of the piece is obtained.

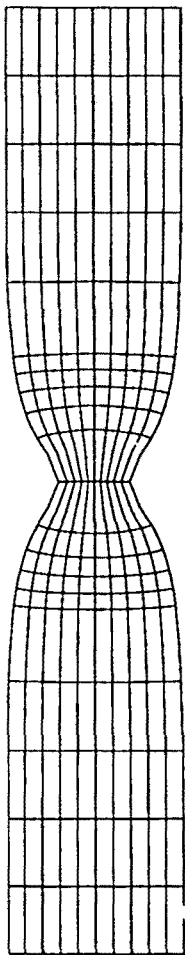


Fig. 5.4a

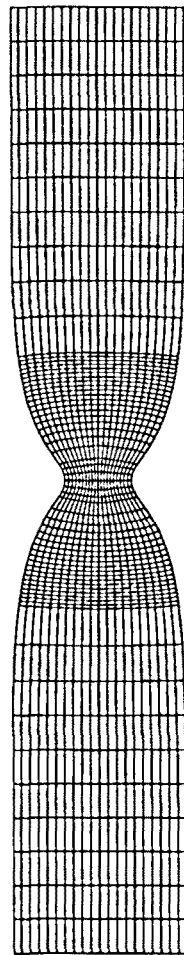


Fig. 5.4b

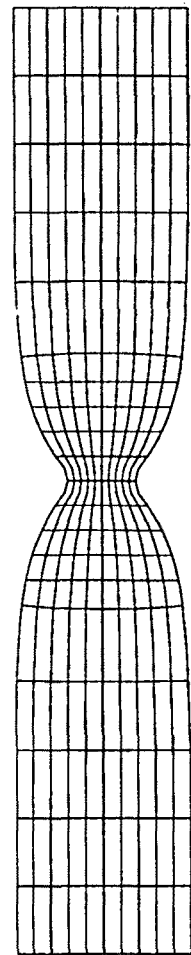


Fig. 5.4c

Figure 5.4 Test NECKING. Deformed mesh after 7mm elongation. a) Lagrangian formulation with coarse mesh. b) Lagrangian formulation with fine mesh. c) ALE formulation

The quality of the solution may be improved by performing the Lagrangian simulation on the fine mesh, Figure 5.4b. An alternative approach, however, is to keep the coarse mesh and use employ the ALE formulation. A very simple ALE remeshing strategy is suggested by the results of the Lagrangian analyses: 1) the upper part of the piece, where strains are not large, remains Lagrangian (that is, convective velocity is set to zero, so there is no need to perform the convection step of the stress update) and 2) equal height of elements is prescribed in the central part, thus avoiding the excessive distortion of elements near the neck of Figure 5.4a. The three update algorithms result in very similar deformed shapes; the output with the interpolation procedure is shown in Figure 5.4c.

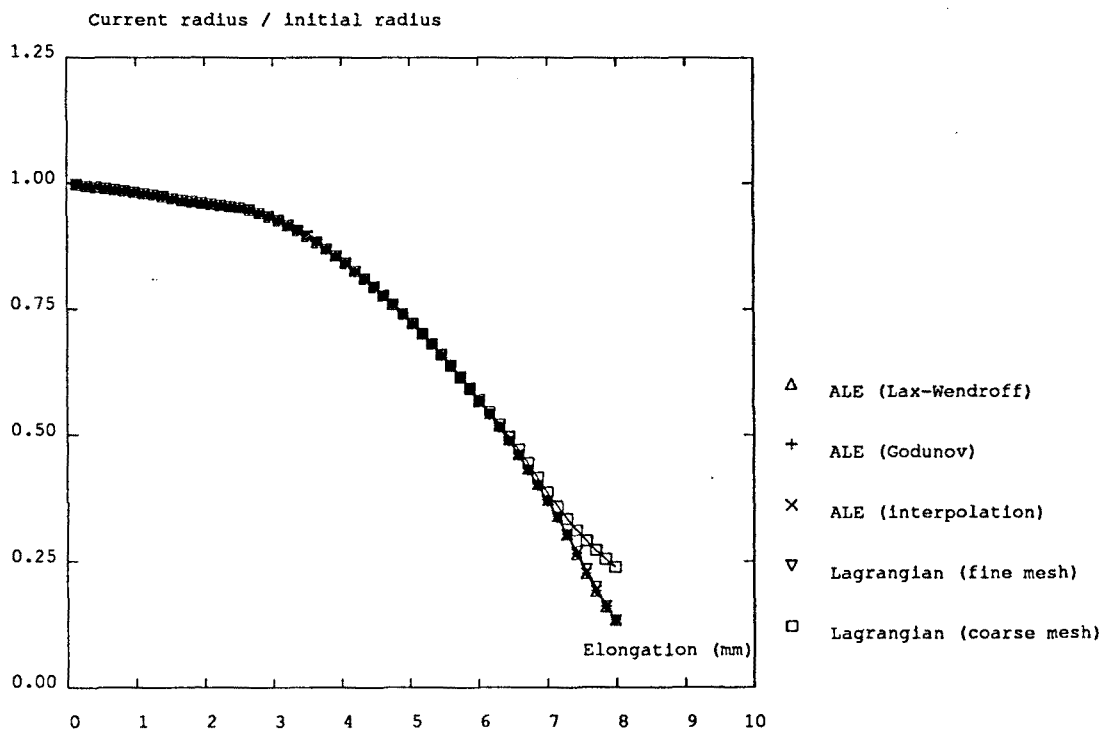


Figure 5.5 Test NECKING. Radius reduction vs. bar elongation for various simulations

A more quantitative comparison of the various simulations is offered in Figure 5.5, which shows the evolution of radius reduction (ratio of current radius to initial radius)

to elongation (in mm, for half the specimen). It can be seen that, up to 7 mm elongation, all the curves are very similar and in good agreement with those in Simo (1988). If pulling proceeds, however, discrepancies arise between the Lagrangian solution with the coarse mesh, on one side, and the Lagrangian solution with the fine mesh and the ALE solutions, on the other. With only one row of (very distorted) elements in the necking zone, the Lagrangian simulation on the coarse mesh does not fully capture the plastification process, and this results in less necking.

A closer look to this behaviour is presented in Figure 5.6. Taking the Lagrangian solution with the fine mesh as a reference, the relative error in radius of the simulations with the coarse mesh is plotted versus elongation. After a pull of 7 mm, the Lagrangian solution offers an acceptable error of 5%, compared to ALE values of slightly under 1%. If pulling continues up to 8 mm, however, the error for the Lagrangian solution grows to 80% while it stays below 5% for the three ALE cases, see Figure 5.6a. A zoom focusing on the ALE analyses, Figure 5.6b, shows that the three update procedures yield almost identical results.

It is apparent from Figures 5.5 and 5.6 that the evolution of necking is properly described by the three ALE stress update algorithms, and that tougher criteria must be employed to assess their relative performance. A possibility is to compare the distributions of a certain stress-related component. Figure 5.7 shows the profiles of the equivalent plastic strain in the neck zone after an elongation of 7 mm. Again, the Lagrangian analysis with the fine mesh is taken as a reference, Figure 5.7a. The Lagrangian simulation with the coarse mesh correctly describes the general aspect of the field, but fails to capture the large plastification in the neck and underestimates the peak value of the plastic strain by 23%, see Table 5.1. As for the ALE cases, the best performance is shown by the interpolation procedure, Figure 5.7c, with a very similar profile and an underestimation of the peak value of only 1.4%. Second comes the Lax-Wendroff update, Figure 5.7d, with an error in the peak value of 4%, and finally the Godunov-like update, Figure 5.7e, with an error of 6%.

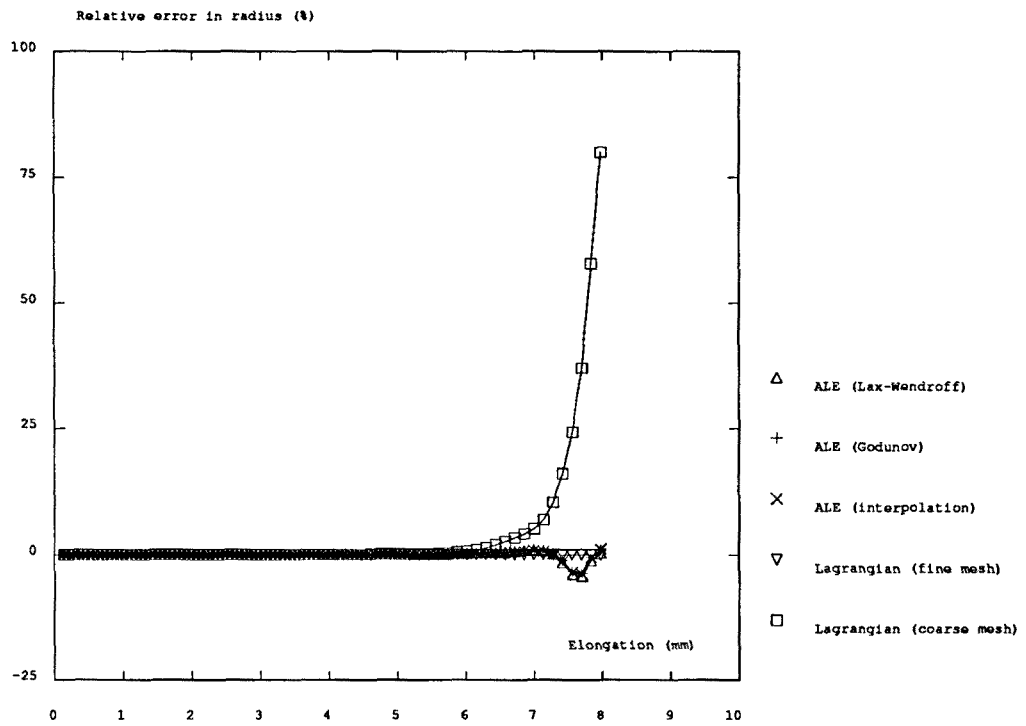


Fig. 5.6a

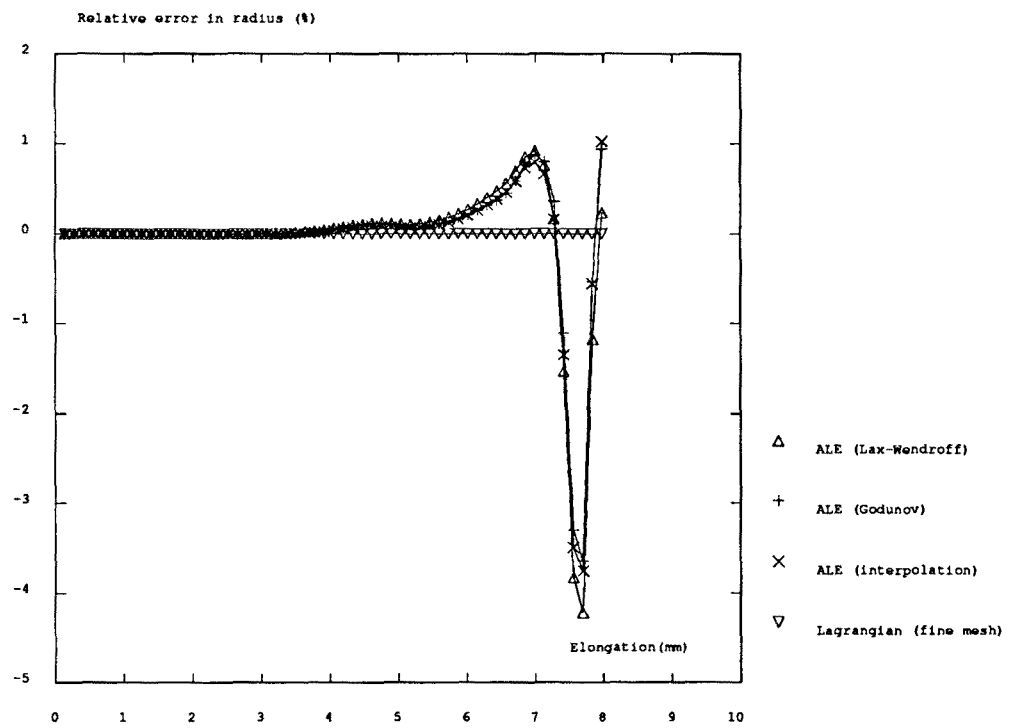


Fig. 5.6b

Figure 5.6 Test NECKING. Relative error in radius vs. bar elongation for various simulations.
 a) Full view. b) Zoom

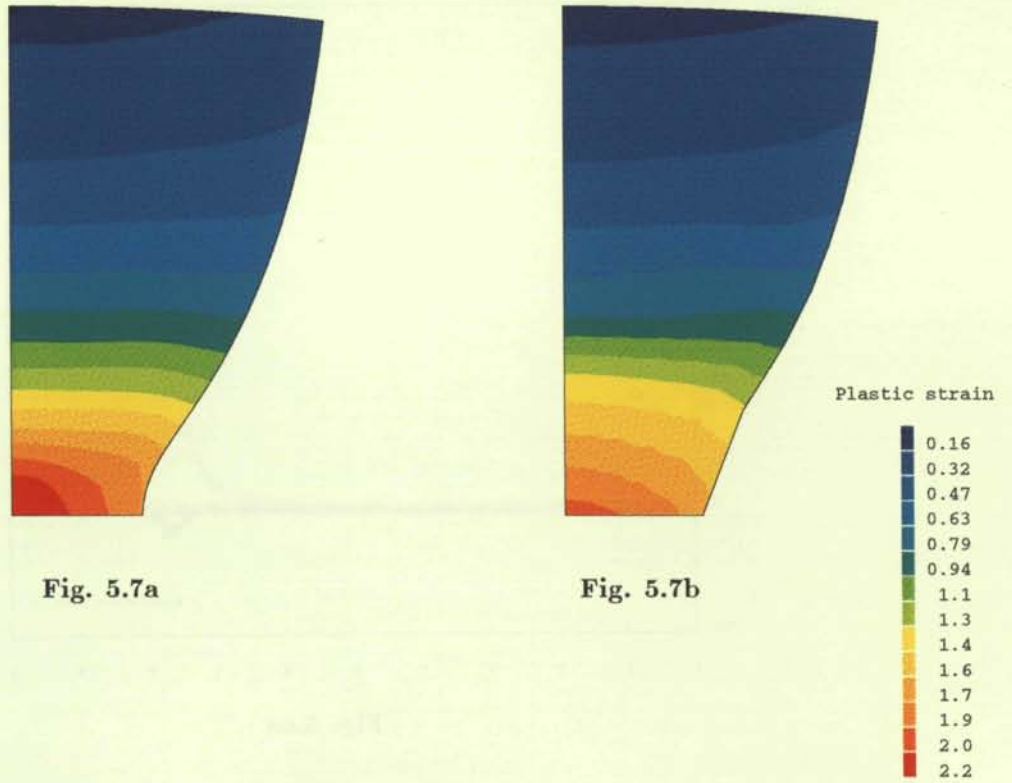


Fig. 5.7a

Fig. 5.7b

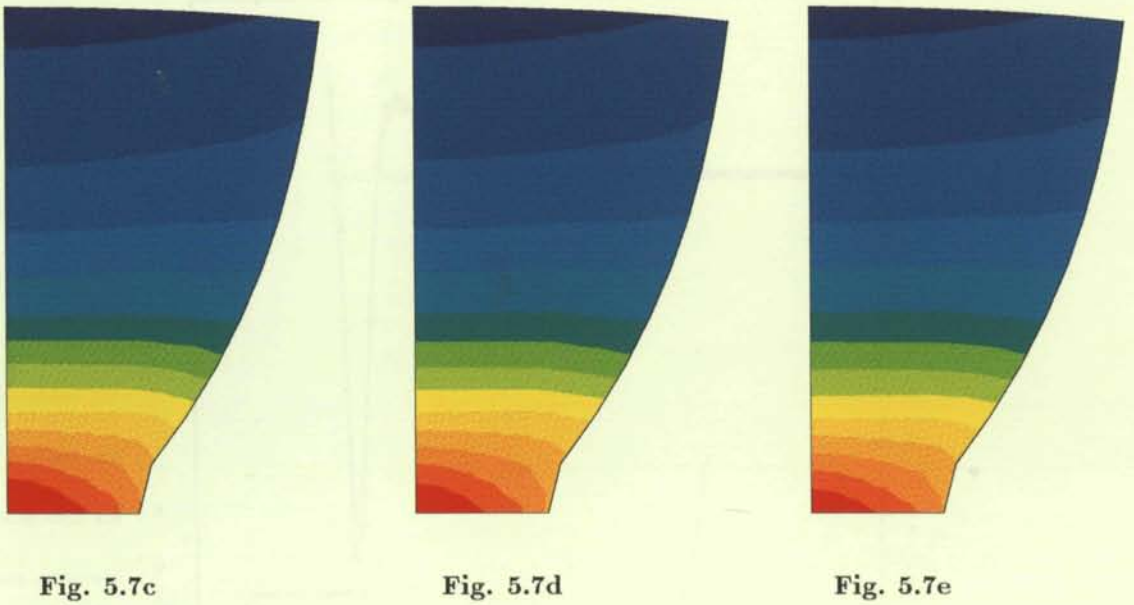


Fig. 5.7c

Fig. 5.7d

Fig. 5.7e

Figure 5.7 Equivalent plastic strain in the neck zone. Lagrangian formulation: a) fine mesh ; b) coarse mesh. ALE formulation (coarse mesh): c) interpolation update ; d) Lax-Wendroff update ; e) Godunov update

	Max. plastic strain	Relative error
Lagrangian analysis		
Fine mesh	2.28	Reference value
Coarse mesh	1.76	-23%
ALE analysis, coarse mesh		
Lax-Wendroff update	2.18	-4%
Godunov update	2.14	-6%
Interpolation update	2.25	-1.4%

Table 5.1 Test NECKING. Maximum plastic strain for various simulations

5.5.3 Test COINING

In this test, a coining process is simulated (Casadei *et al.*, 1995). A metallic disk, with a radius of 30mm and a height of 10mm, is deformed by a punch 12mm in radius, see Figure 5.8. The disk is made of an elastoplastic material with elastic modulus $E = 200\text{GPa}$, Poisson's coefficient $\nu = 0.3$, yield stress $\sigma_y = 250\text{MPa}$ and plastic modulus $E_p = 1\text{GPa}$. Both the punch and the die are rigid. Perfect friction (stick) conditions are assumed in the punch-disk and disk-die interfaces.

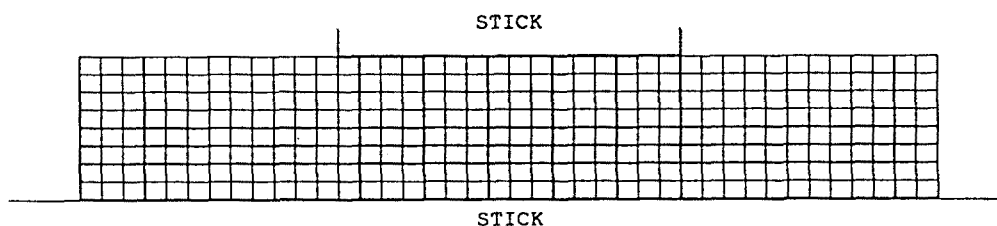


Figure 5.8 Test COINING. Problem statement

An axisymmetric analysis is performed to model a 60% height reduction. The finite element mesh is made of 8×20 eight-noded elements, with 2×2 Gauss points.

Figure 5.9 shows the output of a Lagrangian simulation. It can be seen that, at 36% height reduction, the finite elements under the punch corner and in contact with the die become very distorted, and the analysis cannot proceed. The pattern of material flow is also clear from this sequence of pictures: Since there is perfect friction in the two interfaces, the material tends to flow outward from the central part of the disk. The outer part of the disk is relatively unaffected by deformation and moves in a rather rigid manner.

With an ALE analysis, on the contrary, the analysis can be completed. Figure 5.10 shows the evolution of the ALE mesh and the yield stress. The Godunov-like algorithm is employed for the convection phase of the ALE stress update. A similar result is obtained with the Lax-Wendroff algorithm, see Figure 5.11.

In Casadei *et al.* (1995), a transient analysis of this coining test is performed. To assess the influence of dynamic effects, various punch velocities, ranging from 300m/s to 0.15m/s, are simulated. It is interesting to note that the results shown here for a quasistatic analysis (Figures 5.10 and 5.11) are in good agreement with those of Casadei *et al.* (1995), in the sense that they are the limit case with null inertia effects in the sequence of punch velocities (both in terms of deformed shape and yield stress profile).

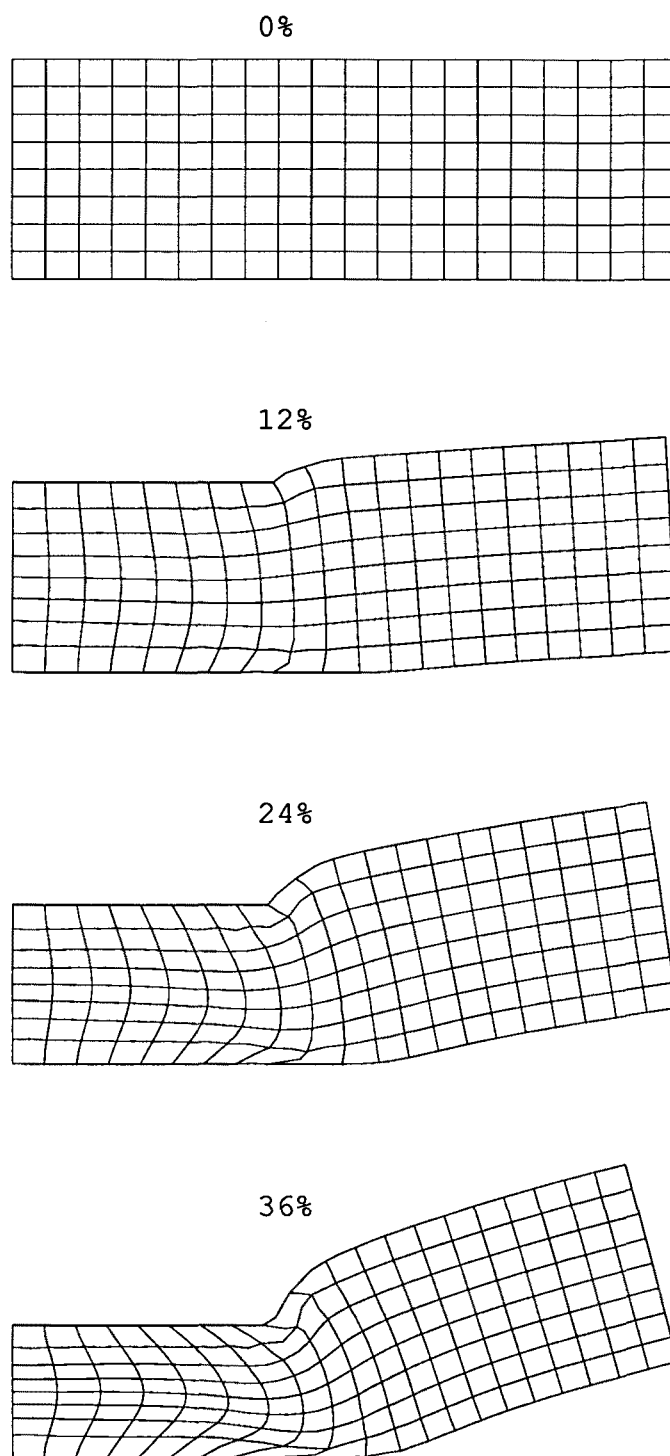


Figure 5.9 Test COINING. Distorted mesh in a Lagrangian analysis

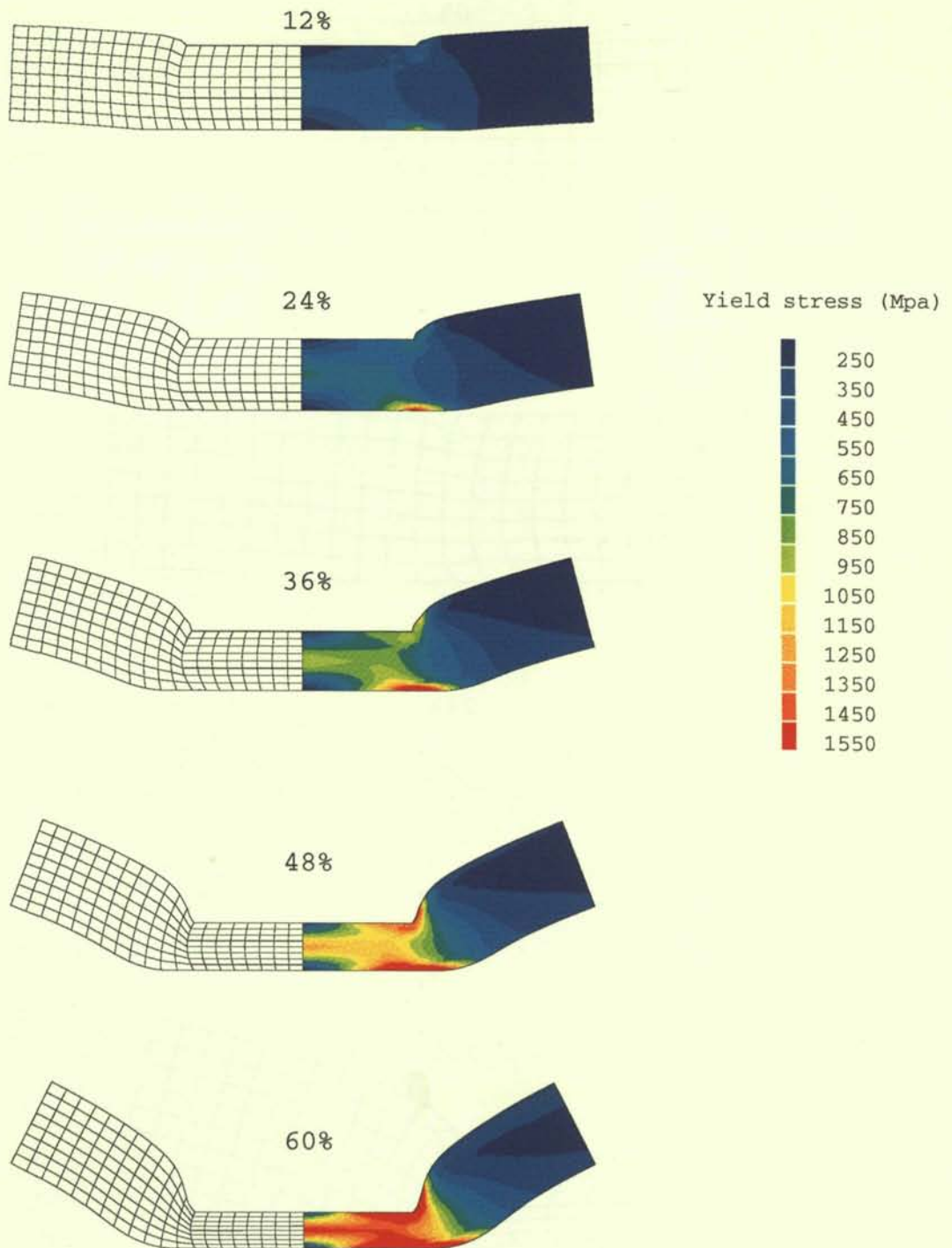


Figure 5.10 Test COINING. Evolution of ALE mesh and yield stress

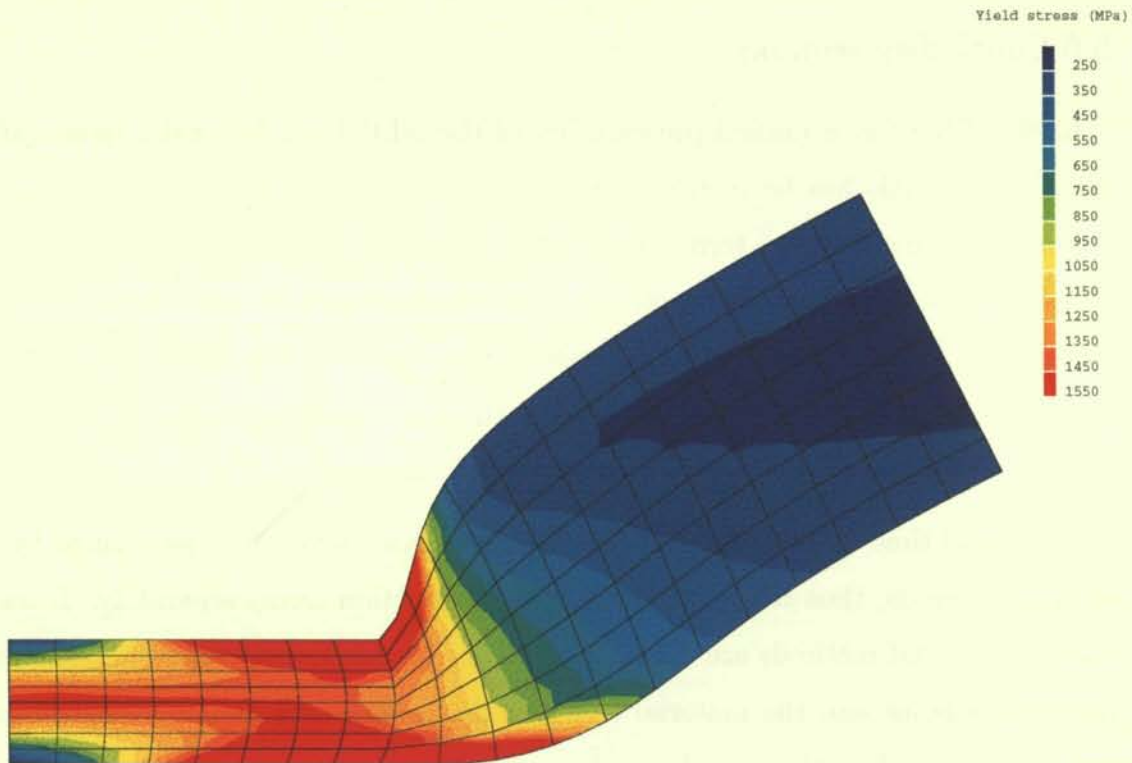


Fig. 5.11a

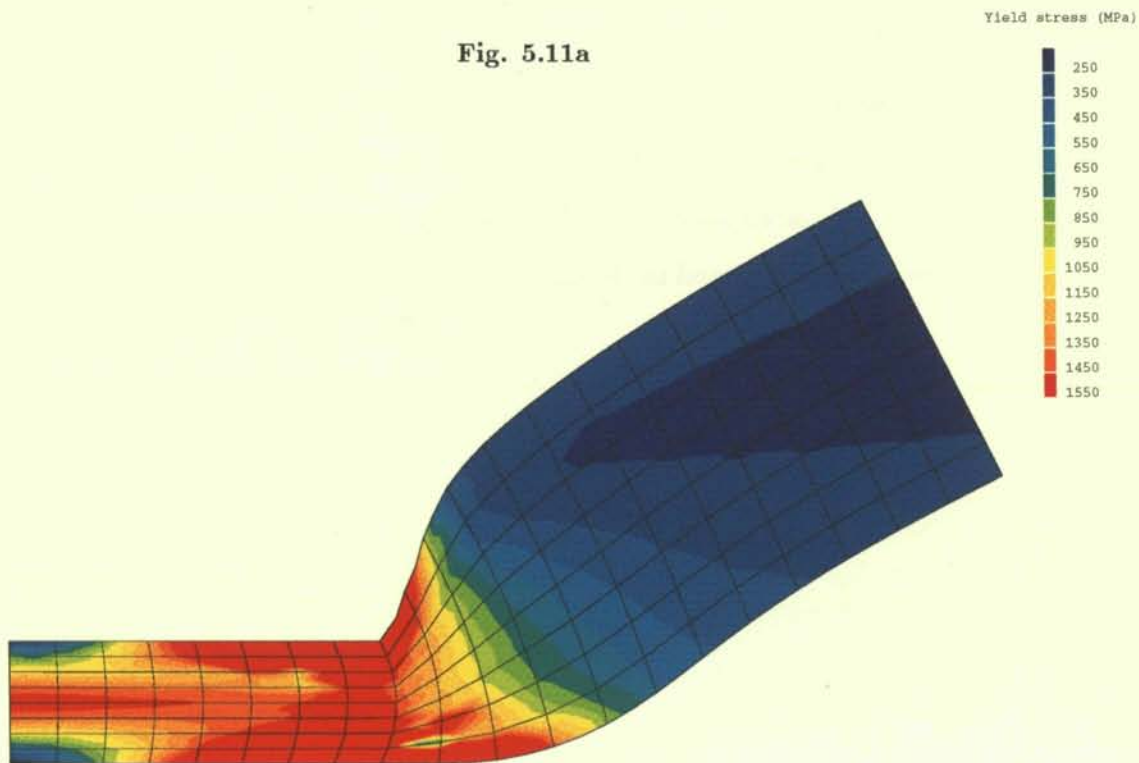


Fig. 5.11b

Figure 5.11 Test COINING. Final ALE mesh and yield stress. a) Godunov update. b) Lax-Wendroff update

5.6 Concluding remarks

In this Chapter, a unified presentation of the ALE formulation for quasistatic and transient analysis has been made. In the general (i.e. transient) case, the governing equations have convective terms that reflect the relative motion between mesh and material. In the quasistatic case, on the contrary, there are no convective terms in the momentum balance (because inertia effects associated to the convective acceleration are neglected), but they are still present in the constitutive equation (representing the arbitrary mesh motion which is inherent to ALE kinematics).

Numerical time-integration of the governing equations has been performed by means of split methods, thus treating material and convection terms separately. It has been shown that split methods are the natural choice for ALE quasistatic analysis: the momentum balance and the material terms in the constitutive equation can be handled with the same algorithms used in a Lagrangian analysis. After that, during the convection phase it is only necessary to worry about the convection terms in the ALE constitutive equation.

As illustrated with some numerical examples, this convection phase can be performed by means of explicit algorithms, in spite of the implicit algorithm employed for the momentum balance. Three explicit algorithms (Lax-Wendroff, Godunov and interpolation procedure) have been discussed in detail.

CONCLUSIONS

A methodology for the ALE analysis of quasistatic problems in nonlinear solid mechanics is presented in this work. The basic features of the proposed strategy are:

General framework. A unified treatment of ALE quasistatic and transient analysis has been made. Emphasis is put on stressing the similarities and differences between the two types of problems. Split methods are employed for the numerical time-integration of the ALE constitutive equation. With this choice, material terms and convection terms are accounted for in two separate phases. It has been shown that the convection term in the ALE constitutive equation can be handled by means of explicit algorithms, in spite of the implicit time-integration of the momentum balance. In this manner, the same algorithms, with no modification, can be used for the ALE stress update in both quasistatic and transient analyses.

Easy implementation. The presented strategy can be used to add ALE capabilities into a small-strain finite element code in a simple way, with few additional features. Since a split approach is chosen for the ALE stress update, the Lagrangian phase can be performed with the algorithms of Chapter 4, just like in a Lagrangian analysis, and the convection phase can be handled with the explicit algorithms of Chapter 5.

Low extra cost (with respect to a Lagrangian analysis). All the additional features (i.e. the determination of mesh motion and the treatment of the convective term in

the ALE constitutive equation) are performed by means of explicit algorithms. Compared with the implicit algorithms typically employed in quasistatic analysis for the momentum balance, this results in a low extra computational cost.

The ALE formulation of quasistatic nonlinear problems has motivated research in some more fundamental topics of nonlinear computational mechanics, which are relevant issues in ALE quasistatic analysis and many other fields: the solution of nonlinear systems of equations, the time-integration of constitutive equations in large strain solid mechanics and the choice of a programming environment to implement and test the new algorithms. These three topics have been addressed in separate Chapters.

Object-oriented codes

Object-oriented codes are a robust, valid tool for both real applications and research. Since the objects directly represent the entities required in a finite element computation, there is no need to translate them into more rudimentary variables.

CASTEM 2000, the object-oriented code employed for this work, provides an interactive object-oriented language which is well suited for research tasks. The new developments can be implemented and tested in a simple way, by writing new metaoperators in this interactive language. Besides, they can be shared among various workteams in an efficient way.

Nonlinear equation solvers

If an object-oriented code is employed, the Lagrange-multiplier technique is a natural way to impose linear constraints (associated to boundary conditions) into the nonlinear equilibrium equation. In this work, various nonlinear equation solvers have been adapted to the Lagrange-multiplier, object-oriented environment.

The adaptation is straightforward for the methods of the Newton-Raphson family (full Newton-Raphson, modified Newton-Raphson and initial stress methods). The stiffness matrix is enlarged into the Jacobian matrix with the constraint matrix. For the Quasi-Newton and Secant-Newton methods, on the contrary, this adaptation process is rather more involved. In fact, two different approaches to defining secant approximations to the Jacobian matrix have been presented: the natural (or standard) approach, and

an alternative (or modified) approach. The alternative approach takes profit from the partial linearity of the nonlinear system of equations.

It has been shown here that these two approaches are identical for rank-two Quasi-Newton methods (DFP and BFGS). They are different, on the contrary, for the rank-one Broyden method. Two different Broyden methods have been found: the total Broyden (natural approach) and the partial Broyden (alternative approach) methods. All these Quasi-Newton methods have superlinear convergence under standard regularity conditions.

A similar situation is found with the Secant-Newton methods (i.e. a partial Secant-Broyden method and a total Secant-Broyden method, but only one Secant BFGS method, are found).

The numerical experiments demonstrate the effectiveness of the adapted algorithms and the superior performance of the partial versions of the Broyden method (QN and SN) over the total versions.

Stress update algorithms for large strains

Two stress update algorithms for large strains, originally developed in a fixed Cartesian frame, have been presented here with a convected frame formalism. The use of convected frames has enabled the development of a general strategy for the accuracy analysis of stress update algorithms, based on standard techniques in numerical analysis. The basic idea is that, if the constitutive equation is stated in convected components, then any standard finite difference scheme can be employed for its numerical time-integration.

For the two algorithms discussed here, this accuracy analysis has shown that one algorithm (Bathe *et al.*, 1975) is first-order accurate, while the other one (Pinsky *et al.*, 1983) has second-order accuracy. This a-priori accuracy analysis is corroborated by a set of numerical experiments.

Regarding the implementation aspects, it has been shown that any of the two algorithms can be employed to add large strain capabilities to a small-strain code in a simple way.

This global approach to the problem has been taken at the expense of some issues which are relevant for more realistic simulations: a general ALE remeshing algorithm, friction and lubrication in the piece-tool interface, These issues are not addressed here and are prospective topics for future research.

Another issue which needs to be studied is the coupling of the ALE formulation and an error estimator. The basic idea would be to define the arbitrary mesh motion to concentrate finite elements where they are most needed according to the error estimator. This strategy could complement the geometrical criteria (i.e. reduction of the element distortion) used here. In a second stage, it is intended to combine the ALE formulation with an adaptive remeshing technique, to profit from their respective advantages: adaptive remeshing allows to increase/decrease the number of elements in the mesh, but the information transfer is simpler in the ALE formulation.

Appendix A

A review of nonlinear equation solvers

In Chapter 3, the adaptation of nonlinear solvers to the context of Lagrange multipliers is discussed. In this Appendix, some of the various solvers are reviewed in their classical form.

A.1 A classical approach to nonlinear problems: Newton–Raphson methods

A.1.1 An incremental solution. The tangent stiffness matrix

We recall first Eq. (3.2),

$$\mathbf{r}(\mathbf{u}) = \mathbf{f}_{\text{int}}(\mathbf{u}) - \mathbf{f}_{\text{ext}} = \mathbf{0},$$

which describes the system of nonlinear equations that needs to be solved. It will be assumed also that the total external load has been fractioned into N increments of load or load steps.

Denoting each increment by ${}^n\Delta\mathbf{f}$ and the intermediate forces by ${}^n\mathbf{f} = {}^{n-1}\mathbf{f} + {}^n\Delta\mathbf{f}$ we have that $\mathbf{f}_{\text{ext}} = {}^1\Delta\mathbf{f} + {}^2\Delta\mathbf{f} + \dots + {}^{N-1}\Delta\mathbf{f} + {}^N\Delta\mathbf{f} = {}^N\mathbf{f}$. The original problem (3.2) is then transformed into the following collection of N nonlinear problems:

1) Find ${}^1\mathbf{u}$ such that

$$\mathbf{r}({}^1\mathbf{u}) = \mathbf{f}_{\text{int}}({}^1\mathbf{u}) - {}^1\mathbf{f} = \mathbf{0}.$$

2) Find ${}^2\Delta\mathbf{u}$ such that, if ${}^2\mathbf{u}$ is built as ${}^2\mathbf{u} = {}^1\mathbf{u} + {}^2\Delta\mathbf{u}$, then

$$\mathbf{r}({}^2\mathbf{u}) = \mathbf{f}_{\text{int}}({}^2\mathbf{u}) - {}^2\mathbf{f} = \mathbf{0},$$

etcetera.

This procedure (incremental approach) is summarized in Box A.1.

For $n = 0, \dots, N - 1$

and assuming we know a vector ${}^n\mathbf{u}$ of nodal displacements for which

$$\mathbf{r}({}^n\mathbf{u}) = \mathbf{f}_{\text{int}}({}^n\mathbf{u}) - {}^n\mathbf{f} = \mathbf{0},$$

we want to obtain a vector ${}^{n+1}\Delta\mathbf{u}$ of incremental displacements such that

$$\text{for point } {}^{n+1}\mathbf{u}, \text{ calculated as } {}^{n+1}\mathbf{u} = {}^n\mathbf{u} + {}^{n+1}\Delta\mathbf{u},$$

we have that

$$\mathbf{r}({}^{n+1}\mathbf{u}) = \mathbf{f}_{\text{int}}({}^{n+1}\mathbf{u}) - {}^{n+1}\mathbf{f} = \mathbf{0}.$$

Box A.1 Incremental approach

All efforts must now be concentrated on the obtention of vector ${}^{n+1}\Delta\mathbf{u}$.

What we are going to do is **linearize** each system of nonlinear equations that is listed in Box A.1; next, we need to check if the solution of the equivalent linear system of equations is similar to that of the original, nonlinear system.

Let us begin by considering the function

$$\mathbf{r}(\mathbf{u}) = \mathbf{f}_{\text{int}}(\mathbf{u}) - {}^{n+1}\mathbf{f}$$

and making a Taylor expansion of $\mathbf{r}({}^{n+1}\mathbf{u})$ around point ${}^n\mathbf{u}$. This expansion consists of an infinite number of terms of increasing order. However, since a linearization of the problem is performed, only terms of first order are taken into account; this means that the following approximation is made

$$\mathbf{r}({}^{n+1}\mathbf{u}) \approx \mathbf{r}({}^n\mathbf{u}) + \left. \frac{\partial \mathbf{r}}{\partial \mathbf{u}} \right|_{{}^n\mathbf{u}} {}^{n+1}\Delta\mathbf{u},$$

where vector $\mathbf{r}^{(n+1)\mathbf{u}}$ is forced to be null. Reorganizing terms in the resulting equation,

$$\mathbf{r}^{(n\mathbf{u})} + \left. \frac{\partial \mathbf{r}}{\partial \mathbf{u}} \right|_{n\mathbf{u}} {}^{n+1}\Delta \mathbf{u} = \mathbf{0}, \quad (\text{A.1})$$

we obtain the desired system of linear equations

$${}^n K {}^{n+1}\Delta \mathbf{u} = -{}^n \mathbf{r}, \quad (\text{A.2})$$

where ${}^n \mathbf{r}$ stands for $\mathbf{r}^{(n\mathbf{u})}$ and

$${}^n K = \left. \frac{\partial \mathbf{r}}{\partial \mathbf{u}} \right|_{n\mathbf{u}}$$

is the expression of the the tangent stiffness matrix (Crisfield, 1991).

A.1.2 The need for incremental/iterative solutions

In the previous Subsection, the N nonlinear systems of equations that make up the initial nonlinear problem (3.2) have been transformed into N linear systems of the type

$${}^n K {}^{n+1}\Delta \mathbf{u} = -{}^n \mathbf{r}.$$

However, for each load step the solution of the linearized system of equations is just an **approximation** to the solution of the nonlinear system defined by Eq. (3.2). Thus, the vector of displacements

$${}^{n+1}\mathbf{u} = {}^n \mathbf{u} + {}^{n+1}\Delta \mathbf{u}$$

does not give null residual forces $\mathbf{r}^{(n+1)\mathbf{u}}$.

This last point is illustrated in Figure A.1. The solution to the nonlinear problem is denoted by ${}^{n+1}\hat{\mathbf{u}}$.

The obtention of vector ${}^{n+1}\mathbf{u}$ begins by performing a Taylor first-order expansion of the function

$$\mathbf{r}(\mathbf{u}) = \mathbf{f}_{\text{int}}(\mathbf{u}) - {}^{n+1}\mathbf{f}$$

around point ${}^n\mathbf{u}$. As only conservative forces are considered, ${}^{n+1}\mathbf{f}$ is constant. Therefore, the nonlinear function $\mathbf{f} = \mathbf{f}_{\text{int}}(\mathbf{u})$ is approximated by the linear function

$$\mathbf{f} = {}^n\mathbf{f} + {}^n\mathbf{K} (\mathbf{u} - {}^n\mathbf{u}),$$

which is tangent to the curve $\mathbf{f} = \mathbf{f}_{\text{int}}(\mathbf{u})$ at point $({}^n\mathbf{u}, {}^n\mathbf{f})$.

The intersection of this linear function with the constant function $\mathbf{f} = {}^{n+1}\mathbf{f}$ gives point ${}^{n+1}\mathbf{u}$, for which

$${}^n\mathbf{f} + {}^n\mathbf{K} ({}^{n+1}\mathbf{u} - {}^n\mathbf{u}) = {}^{n+1}\mathbf{f}.$$

(or also ${}^n\mathbf{K} {}^{n+1}\Delta\mathbf{u} = {}^n\mathbf{K} ({}^{n+1}\mathbf{u} - {}^n\mathbf{u}) = {}^{n+1}\mathbf{f} - {}^n\mathbf{f} = -{}^n\mathbf{r}$).

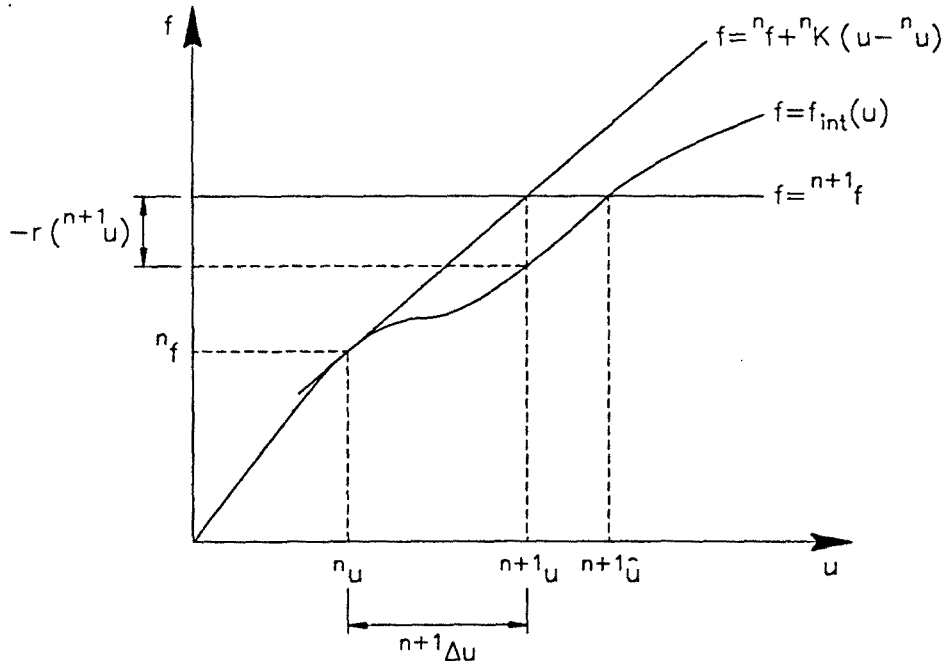


Figure A.1 Solutions to the nonlinear and linearized problems

Visibly, the use of an incremental strategy is not sufficient by itself. However, what can be done is to take profit of the previous ideas to obtain successive approximations (**iterations**) to the solution within each load step. The final scheme (**incremental/iterative solution**) is described as follows.

- For a certain load step ${}^{n+1}\Delta\mathbf{f}$, the linear system of equations

$${}^n\mathbf{K} {}^{n+1}\Delta\mathbf{u} = -{}^n\mathbf{r}, \quad (\text{A.3})$$

is solved.

Writing the out-of-balance forces associated to the displacement vector \mathbf{u} as

$$\mathbf{r}(\mathbf{u}) = \mathbf{f}_{\text{int}}(\mathbf{u}) - {}^{n+1}\mathbf{f} \quad (\text{A.4})$$

we have that

$${}^n\mathbf{r} = \mathbf{r}({}^n\mathbf{u}) = \mathbf{f}_{\text{int}}({}^n\mathbf{u}) - {}^{n+1}\mathbf{f} = -{}^{n+1}\Delta\mathbf{f},$$

that is, the residual forces ${}^n\mathbf{r}$ equal minus the total increment of load associated to the current load step. The solution to Eq. (A.3) will be denoted by ${}^{n+1}\Delta\mathbf{u}^1$ to indicate that it yields a **prediction** ${}^{n+1}\mathbf{u}^1 = {}^n\mathbf{u} + {}^{n+1}\Delta\mathbf{u}^1$ to the solution of the nonlinear problem.

- The prediction vector ${}^{n+1}\mathbf{u}^1$ might however be a good solution for Eq. (A.4). To check this possibility, we compute the value of the residual forces

$$\mathbf{r}({}^{n+1}\mathbf{u}^1) = \mathbf{f}_{\text{int}}({}^{n+1}\mathbf{u}^1) - {}^{n+1}\mathbf{f}.$$

If this value is close enough to zero, no iterations are needed.

- If, on the contrary, $\mathbf{r}({}^{n+1}\mathbf{u}^1)$ is too different from zero, the iterative part of the process begins, where point ${}^{n+1}\mathbf{u}^1$ will be assumed to play the same role as ${}^n\mathbf{u}$ did in the prediction phase.

The out-of-balance forces are computed as

$$\mathbf{r}^{(n+1)\mathbf{u}^1} = \mathbf{f}_{\text{int}}^{(n+1)\mathbf{u}^1} - {}^{n+1}\mathbf{f}$$

and we consequently need to solve the linear system

$${}^{n+1}\mathbf{K}^1 {}^{n+1}\delta\mathbf{u}^2 = -{}^{n+1}\mathbf{r}^1, \quad (\text{A.5})$$

where ${}^{n+1}\mathbf{r}^1 = \mathbf{r}^{(n+1)\mathbf{u}^1}$ and ${}^{n+1}\mathbf{K}^1$ is the tangent matrix calculated at point $({}^{n+1}\mathbf{u}^1, \mathbf{f}_{\text{int}}^{(n+1)\mathbf{u}^1})$, namely

$${}^{n+1}\mathbf{K}^1 = \left. \frac{\partial \mathbf{r}}{\partial \mathbf{u}} \right|_{n+1\mathbf{u}^1}.$$

The solution ${}^{n+1}\delta\mathbf{u}^2$ to Eq. (A.5) is a correction of the solution to Eq. (A.3); this correction is used to build a better approximation ${}^{n+1}\mathbf{u}^2$ to the solution of the nonlinear system after a simple update such as the following:

$${}^{n+1}\Delta\mathbf{u}^2 = {}^{n+1}\Delta\mathbf{u}^1 + {}^{n+1}\delta\mathbf{u}^2$$

$${}^{n+1}\mathbf{u}^2 = {}^n\mathbf{u} + {}^{n+1}\Delta\mathbf{u}^2.$$

- At this stage, we need to check if point ${}^{n+1}\Delta\mathbf{u}^2$ supplies a good enough solution to the problem by evaluating the residual vector

$$\mathbf{r}^{(n+1)\mathbf{u}^2} = \mathbf{f}_{\text{int}}^{(n+1)\mathbf{u}^2} - {}^{n+1}\mathbf{f}.$$

- Otherwise, we calculate ${}^{n+1}\delta\mathbf{u}^3$ such that

$${}^{n+1}\mathbf{K}^2 {}^{n+1}\delta\mathbf{u}^3 = -{}^{n+1}\mathbf{r}^2$$

and proceed to obtain a new -corrected- vector of displacements.

After a certain amount of iterations, this process is expected to come to an end. To decide whether the solution is good enough to stop iterating, we evaluate how close the approximation ${}^{n+1}\mathbf{u}^{k+1}$ is to the real solution (for example, by computing how different the residual vector $\mathbf{r}({}^{n+1}\mathbf{u}^{k+1}) = \mathbf{f}_{\text{int}}({}^{n+1}\mathbf{u}^{k+1}) - {}^{n+1}\mathbf{f}$ is from zero).

In Box A.2 the whole process is described for a generic iteration $k + 1$. For $k = 0$, we assume that $\delta\mathbf{u}^1 = \Delta\mathbf{u}^1$.

$$\text{Solve } {}^{n+1}\mathbf{K}^k {}^{n+1}\delta\mathbf{u}^{k+1} = -{}^{n+1}\mathbf{r}^k$$

$$\text{Update } {}^{n+1}\Delta\mathbf{u}^{k+1} = {}^{n+1}\Delta\mathbf{u}^k + {}^{n+1}\delta\mathbf{u}^{k+1}$$

$$\text{Update } {}^{n+1}\mathbf{u}^{k+1} = {}^n\mathbf{u} + {}^{n+1}\Delta\mathbf{u}^{k+1}$$

Convergence control : if the approximation ${}^{n+1}\mathbf{u}^{k+1}$ is good enough, exit.

Box A.2 Iterative scheme within load step $n + 1$

To end this Subsection, a word needs to be said about the **convergence control** step. A **force-based** criterion has been used up until now; however, many other strategies can be followed to check convergence. For instance, we could use the following **displacement-based** variation measure of the solution vector

$$e = \frac{\|{}^{n+1}\delta\mathbf{u}^{k+1}\|}{\|{}^{n+1}\mathbf{u}^{k+1}\|}, \quad (\text{A.6})$$

where $\|\cdot\|$ stands for some vectorial norm (typically, the Euclidean or maximum norm). A very small value of the scalar e means that the total displacements are experiencing a very small change and that we can consequently pass on to the next load step.

Force-based and displacement-based convergence criteria are the most common. Other criteria, such as those involving **energies**, need to be used with more precaution, (Crisfield, 1990).

A.1.3 Full and modified Newton–Raphson methods

In Subsections A.1.1 and A.1.2 the classical **incremental/iterative** approach to nonlinear problems has been presented. As a matter of fact, a specific **method** (the **full Newton–Raphson** or **fNR** method) has already been used to illustrate this approach.

In this Subsection, we review some variations on the fNR method that also follow the **incremental/iterative** scheme.

Let us begin by rewriting the full Newton–Raphson algorithm. Suppose we are located in a certain load step $n + 1$, where $({}^n\mathbf{u}, {}^n\mathbf{f})$ is a known equilibrium point and we want to obtain ${}^{n+1}\mathbf{u}$. Considering ${}^n\mathbf{u}$ as a starting point for this increment, we rename it to ${}^{n+1}\mathbf{u}^0$, where the right superscript 0 indicates that the construction of point ${}^{n+1}\mathbf{u}^0$ is previous to iterations 1, 2, ... Consequently, the stiffness matrix ${}^n\mathbf{K}$ and the out-of-balance force vector ${}^n\mathbf{r}$ can also be rewritten respectively as ${}^{n+1}\mathbf{K}^0$ and ${}^{n+1}\mathbf{r}^0$.

After these substitutions, all terms in the iterative scheme presented in Subsection A.1.2 turn up to have a $n + 1$ left superscript. To lighten the overall notation, we drop left superscripts. Thus, in the coming equations, ${}^{n+1}\mathbf{K}^0$ is written as \mathbf{K}^0 , ${}^{n+1}\Delta\mathbf{u}^1$ is denoted by $\Delta\mathbf{u}^1$, and so on.

Finally, the original list of instructions can be rearranged to produce the compact algorithm that is detailed in Box A.3. Figure A.2 illustrates how this method works for a one-dimensional problem.

prediction

- 1.– Compute K^0
- 2.– Solve the linear system $K^0 \Delta u^1 = -r^0$
- 3.– Update $u^1 = u^0 + \Delta u^1$
- 4.– Evaluate $r^1 = r(u^1)$
- 5.– Convergence control: if point u^1 is good enough, exit.

corrections

$k \geq 1$

- 6.– Compute K^k
- 7.– Solve the linear system $K^k \delta u^{k+1} = -r^k$
- 8.– Update $u^{k+1} = u^k + \delta u^{k+1}$
- 9.– Evaluate $r^{k+1} = r(u^{k+1})$
- 10.– Convergence control: if point u^{k+1} is good enough, exit.
- 11.– Assign $k = k + 1$ and go back to 6.–

Box A.3 Algorithm for the full Newton–Raphson method

Remark A.1: Under certain conditions, (Crisfield, 1991; Dennis & Schnabel, 1983), the full Newton–Raphson method shows a quadratic rate of convergence, i.e., a scalar γ can be found for which

$$\|u^{k+1} - u^*\| \leq \gamma \|u^k - u^*\|^2$$

where u^* denotes the solution corresponding to the current increment. Thus, fNR results in a powerful technique.

Remark A.2: At every iteration, we must compute and factorize one tangent stiffness

matrix, solve one linear system of equations and evaluate one vector of residual forces. Consequently, the fNR is an expensive method.

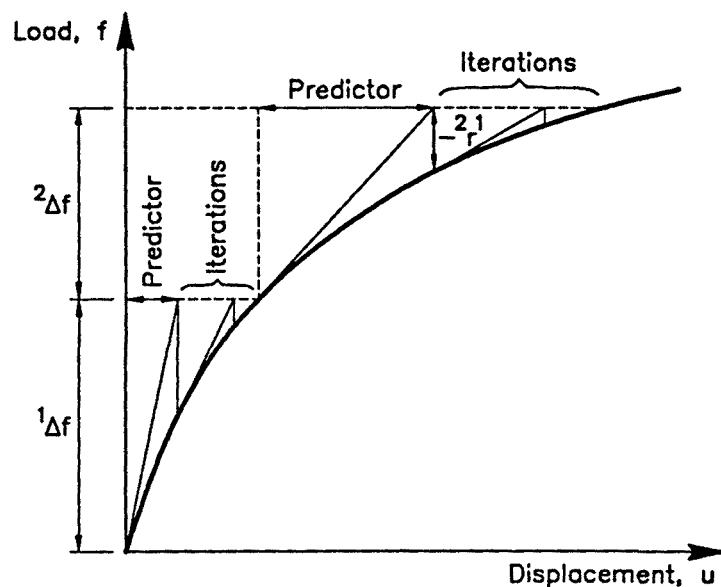


Figure A.2 The full Newton-Raphson method

Once the basic full Newton-Raphson scheme is defined, a whole collection of 'modified Newton-Raphson methods' can be derived from it. The idea of these methods is to keep the original pattern, but using not always up-to-date stiffness matrices.

The reason for these modifications is that calculating a stiffness matrix K and factorizing it to solve the corresponding linear system of equations at every iteration is a very expensive task. Therefore, using a previously calculated and factorized stiffness matrix will result in a much lower computational cost per iteration.

For instance, while in the fNR method a new tangent stiffness matrix is computed at every iteration, in the standard modified Newton-Raphson (or mNR) method

the stiffness matrix is updated just at the beginning of each load step, see Figure A.3. This means that less work is required because only one tangent matrix is computed and factorized per increment. But it also means that a higher amount of iterations should be expected to attain convergence. Thus, the relative global cost between a mNR and a fNR method for the resolution of a given nonlinear problem will in general be unknown beforehand. The algorithm for the mNR method is listed in Box A.4.

Remark A.3: This method has a linear rate of convergence, i.e., a scalar γ can be found for which

$$\|u^{k+1} - u^*\| \leq \gamma \|u^k - u^*\|,$$

where u^* denotes the solution corresponding to the current increment, (Crisfield, 1991; Dennis & Moré, 1977).

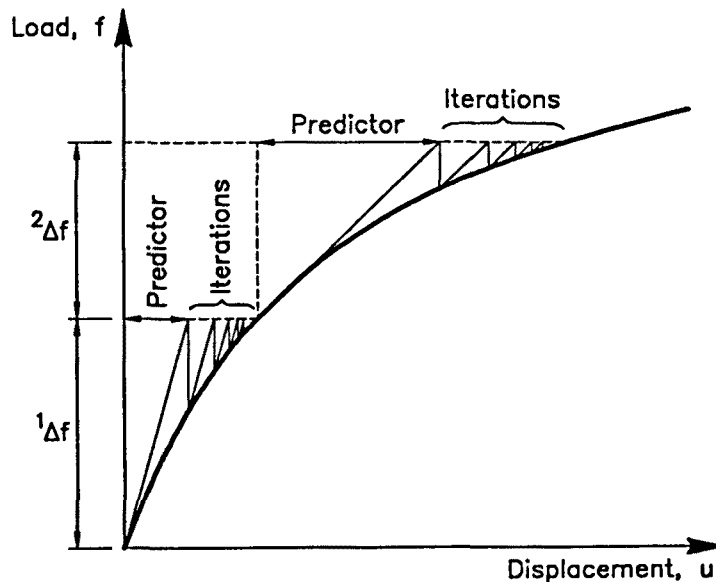


Figure A.3 The modified Newton-Raphson method

prediction

- 1.- Compute K^0
- 2.- Solve the linear system $K^0 \Delta u^1 = -r^0$
- 3.- Update $u^1 = u^0 + \Delta u^1$
- 4.- Evaluate $r^1 = r(u^1)$
- 5.- Convergence control: if point u^1 is good enough, exit.

corrections

$k \geq 1$

- 6.- Solve the linear system $K^0 \delta u^{k+1} = -r^k$
- 7.- Update $u^{k+1} = u^k + \delta u^{k+1}$
- 8.- Evaluate $r^{k+1} = r(u^{k+1})$
- 9.- Convergence control: if point u^{k+1} is good enough, exit.
- 10.- Assign $k = k + 1$ and go back to 6.-

Box A.4 Algorithm for the modified Newton-Raphson method

Obviously, an infinite number of Newton-Raphson techniques can be produced by updating the tangent stiffness matrix at different times during the resolution of the problem.

One more of those techniques, the so-called **initial stress method**, will be discussed here. In this case, one tangent stiffness matrix is computed at the beginning of the first load step, namely 0K , and it is used to solve all the linear systems of equations that are encountered during the resolution of the problem. Therefore, just one tangent matrix is calculated and factorized in all. This means, as it has already been indicated, that more iterations will probably be needed within each load step before convergence is reached – that is, if the process converges, which becomes more improbable since the

tangent stiffness matrix differs more radically from the employed matrix. This technique is illustrated in Figure A.4.

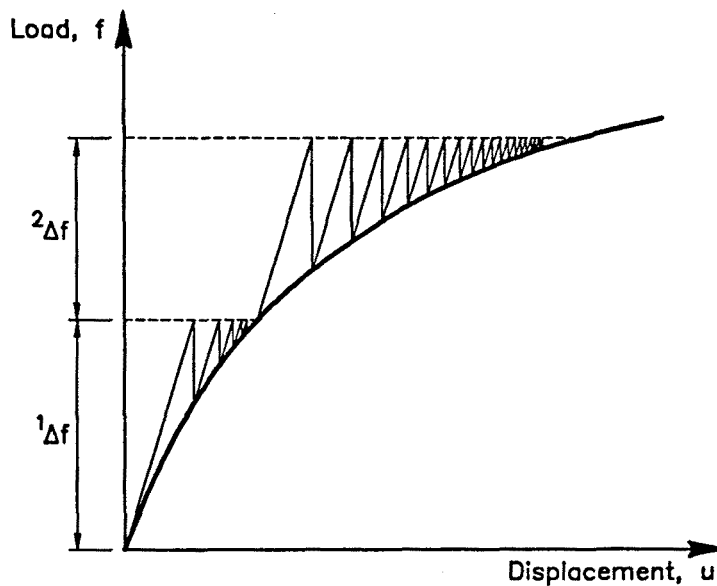


Figure A.4 The Initial Stress method

A.2 Quasi-Newton methods

A.2.1 Motivation and theory

In Section A.1, the classical Newton-Raphson methods have been presented. First, the **full Newton-Raphson method**, which provides second order convergence (i.e., reduced number of iterations) in most of the cases where tangent stiffness matrices are available, but requires in return the computation and factorization of a tangent matrix at the beginning of every iteration.

On the other hand, the **modified Newton–Raphson** and **initial stress** methods require a lower computational cost per iteration as far as stiffness matrices are concerned. However, their convergence rates are also lower, which means that more iterations will probably be needed, with the associated increase in the total cost; moreover, the process may fail to converge.

Naturally, it would be interesting to obtain a method with the convergence behaviour of fNR at a computational cost similar to that of the mNR technique. Quasi-Newton (QN) methods were designed for this purpose. These methods, which have their origin in the field of unconstrained optimization, are generalizations to n -dimensional problems of the well-known “secant method” for finding roots of one nonlinear equation. In fact, the choice of a **secant** iteration matrix instead of a tangent stiffness matrix ensures the two following properties:

- the secant stiffness matrix resembles the tangent stiffness matrix
- but a strict recomputation of this secant stiffness matrix is not required at all when performing the matrix update.

All Quasi-Newton methods use secant stiffness matrices; however, for n -dimensional problems with $n > 1$, while the tangent stiffness matrix is unique, the secant matrix is not (for $n = 1$, both matrices are unique). Consequently, a whole number of different Quasi-Newton methods can be produced by employing **different** secant stiffness matrices.

Before describing some of those methods, let us formalize the unifying feature of all Quasi-Newton techniques: they all use secant stiffness matrices. Assume that iteration k belonging to load step n has just been completed, Figure A.5.

Points \mathbf{u}^{k-1} , \mathbf{u}^k in Figure A.5 are known from the previous iterations. Consequently, the vectors of residual forces \mathbf{r}^{k-1} and \mathbf{r}^k are also known.

Let us turn back for a moment to the one-dimensional problem. In this case, we are able to compute the slope B^k of the secant line that passes through points $(u^{k-1}, f_{\text{int}}^{k-1})$

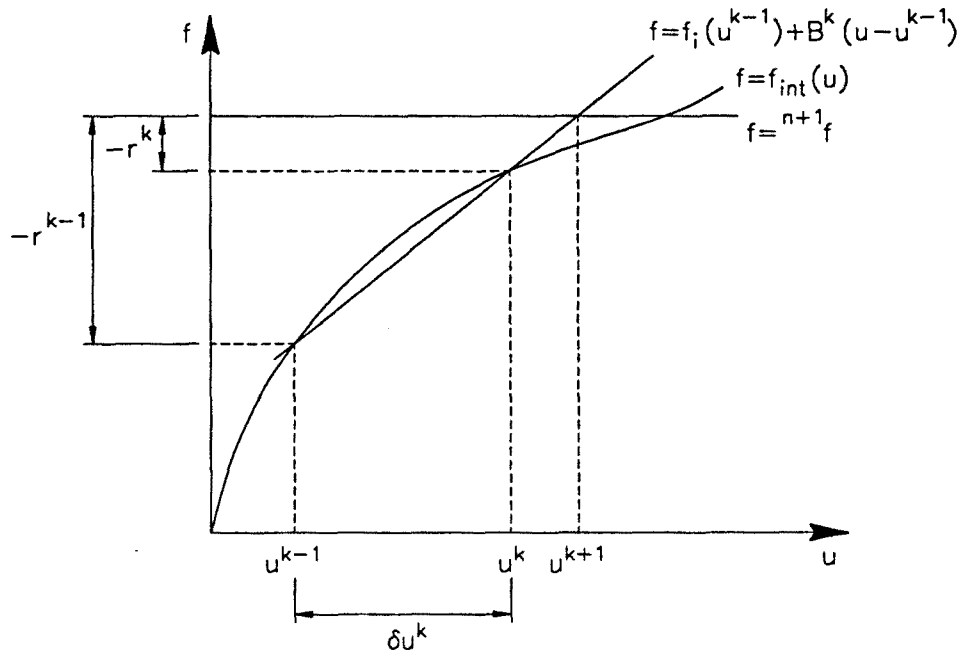


Figure A.5 Performance of a generic Quasi-Newton method

and (u^k, f_{int}^k) , see Figure A.5, directly as

$$B^k = \frac{r^k - r^{k-1}}{\delta u^k}. \quad (A.7)$$

In an n -dimensional problem ($n > 1$), and for a generic value of n , Eq. (A.7) is rewritten into

$$B^k \delta u^k = r^k - r^{k-1}. \quad (A.8)$$

The previous expression (A.8) is known as the **Quasi-Newton equation**, which characterises the **secant stiffness matrix** B^k for a general nonlinear problem. It is important to notice that in this system of equations, the unknown is the matrix, not the vector. That is, if a total of n equations are listed in Eq. (A.8), n^2 unknowns will be obtained. Therefore, as it was briefly indicated in some previous paragraph, the secant matrix that passes through two specified points is **not unique**, and there is a need to

impose some additional equations on B^k ($n^2 - n$ equations, to be precise) in order to be able to determine it.

Defining

$$\mathbf{y}^{k-1} = \mathbf{r}^k - \mathbf{r}^{k-1} \quad (\text{A.9})$$

$$\mathbf{s}^{k-1} = \delta \mathbf{u}^k = \mathbf{u}^k - \mathbf{u}^{k-1}, \quad (\text{A.10})$$

Eq. (A.8) can be put in its most common form

$$B^k \mathbf{s}^{k-1} = \mathbf{y}^{k-1}. \quad (\text{A.11})$$

From this point on, the notation $[\mathbf{s}, \mathbf{y}]$, which is widely accepted in the Quasi-Newton context, will be preferred to the original notation, $[\mathbf{u}, \mathbf{r}]$.

Once a secant matrix B^k is computed, vector \mathbf{u}^{k+1} is obtained as shown in Figure A.5, i.e., by solving the linear system

$$B^k \mathbf{s}^k = -\mathbf{r}^k \quad (\text{A.12})$$

and updating subsequently

$$\mathbf{u}^{k+1} = \mathbf{u}^k + \mathbf{s}^k.$$

A new secant matrix B^{k+1} that passes through points $(\mathbf{u}^k, \mathbf{f}_{\text{int}}^k)$ and $(\mathbf{u}^{k+1}, \mathbf{f}_{\text{int}}^{k+1})$ can then be computed, yielding a new approximation \mathbf{u}^{k+2} , and so on.

This “direct” extension of the one-dimensional secant method into n -dimensional nonlinear problems induces the computation of the “generalized” slope, i.e. matrix B , which is an approximation of the tangent matrix. This strategy defines the **Direct** Quasi-Newton family, in opposition to the **Inverse** Quasi-Newton techniques which are concerned with the approximation of the inverse of the tangent matrix, namely $H = B^{-1}$. In fact, the Quasi-Newton equation (A.11) can also be written as

$$H^k \mathbf{y}^{k-1} = \mathbf{s}^{k-1}, \quad (\text{A.13})$$

which, together with some extra conditions, enables the determination of matrix H^k . Inverse Quasi-Newton are usually preferred, because Eq. (A.12) can be transformed into

$$s^k = -H^k r^k, \quad (\text{A.14})$$

so that there is no need for solving linear systems of equations.

Some Quasi-Newton methods –direct and inverse– are discussed in the following Subsection.

A.2.2 Discussion of various Quasi-Newton methods. Algorithms for Inverse Broyden and BFGS

The Broyden method

Let us consider the direct form of the Quasi-Newton equation that was introduced in Eq. (A.11),

$$B^k s^{k-1} = y^{k-1}.$$

As indicated in Subsection A.2.1, if n is the number of equations in Eq. (A.11), then $n^2 - n$ additional equations on B^k are needed to determine its n^2 components.

Notice that condition (A.11) defines completely the behaviour of B^k along the direction given by s^{k-1} . Moreover, it is the only new information about B^k with respect to B^{k-1} . For this reason, Broyden suggested that matrices B^k and B^{k-1} should have the same behaviour in any direction except s^{k-1} . To accomplish this purpose, the Broyden method requires for the secant matrix B^k to behave in the same way as B^{k-1} on the orthogonal complement of s^{k-1}

$$B^k z = B^{k-1} z \quad \text{for all } z \text{ such that } z^T s^{k-1} = 0. \quad (\text{A.15})$$

It can be proved, (Dennis & Moré, 1977; Fletcher, 1987b), that Eq. (A.11) together with Eq. (A.15) determines one and only one matrix B^k in terms of B^{k-1} , following the expression

$$B^k = B^{k-1} + \frac{(y^{k-1} - B^{k-1} s^{k-1})(s^{k-1})^T}{(s^{k-1})^T s^{k-1}}. \quad (\text{A.16})$$

It is interesting to remark that Eq. (A.16) describes a simple update of B^k after B^{k-1} . Defining vectors v^{k-1} and w^{k-1} as

$$v^{k-1} = \frac{y^{k-1} - B^{k-1} s^{k-1}}{(s^{k-1})^T s^{k-1}}$$

$$w^{k-1} = s^{k-1},$$

Eq. (A.16) is rewritten as

$$B^k = B^{k-1} + v^{k-1} (w^{k-1})^T, \quad (\text{A.17})$$

which is the common expression of the **rank-one update**, so termed because the modification $v^{k-1} (w^{k-1})^T$ is a rank-one matrix. Equation (A.17) must be initialized with a certain B^0 ; some options are discussed later on.

The inverse version of the Broyden method

Once B^k is computed from Eq. (A.16), we need to solve the linear system (A.12),

$$B^k s^k = -r^k,$$

to obtain vector s^k . This operation can be avoided if an expression for H^k rather than for B^k is obtained. By doing so, we can compute s^k directly after Eq. (A.14),

$$H^k r^k = -s^k. \quad (\text{A.18})$$

In order to obtain an expression for H^k from H^{k-1} , the Sherman & Morrison lemma is employed, see Sherman & Morrison (1949) or Dennis & Schnabel (1983).

From the lemma and since $(B^{k-1})^{-1} = H^{k-1}$, we can write $(B^k)^{-1} = H^k$ as

$$H^k = H^{k-1} + \frac{(s^{k-1} - H^{k-1} y^{k-1}) (s^{k-1})^T H^{k-1}}{(s^{k-1})^T (H^{k-1} y^{k-1})}, \quad (\text{A.19})$$

which can also be expressed as

$$H^k = \left[I + \frac{(s^{k-1} - H^{k-1} y^{k-1})}{(s^{k-1})^T (H^{k-1} y^{k-1})} (s^{k-1})^T \right] H^{k-1}, \quad (\text{A.20})$$

where I denotes the n -dimensional identity matrix. Equation (A.20), which must be initialized with some matrix H^0 , is the fundamental relation in the Inverse Broyden method. Notice that, for this inverse QN method, the update formula for H^k is obtained by inverting explicitly the update formula for B^k rather than by using the inverse Quasi-Newton equation and imposing additional conditions on H^k .

From an algorithmic viewpoint, instead of employing Eq. (A.20), it is more convenient, (Engelman *et al.*, 1981; Soria, 1990), to relate H^k to H^0 through

$$\begin{aligned} H^k &= \left[I + \omega^k (s^{k-1})^T \right] \left[I + \omega^{k-1} (s^{k-2})^T \right] \dots \left[I + \omega^1 (s^0)^T \right] H^0 = \\ &= \prod_{i=k}^1 \left[I + \omega^i (s^{i-1})^T \right] H^0, \end{aligned} \quad (\text{A.21})$$

where the auxiliary vectors ω^{i+1} are defined as

$$\omega^{i+1} = \frac{(s^i - H^i y^i)}{(s^i)^T (H^i y^i)}, \quad i = 0, \dots, k-1. \quad (\text{A.22})$$

Before discussing the advantages of Eq. (A.21) over Eq. (A.20), it is convenient to deal with the initialization of the Quasi-Newton updates. As it was previously commented, it is necessary to initialize all Quasi-Newton methods with some matrix B^0 (direct QN) or H^0 (inverse QN). The simplest choice is $B^0 = I$ or $H^0 = I$, where I is the n -dimensional identity matrix.

Nevertheless, if the computation of tangent stiffness matrices is available, it seems that a better choice could be

$$B^0 = K^0, \quad (\text{A.23})$$

that is, initializing the algorithm with a true tangent matrix. This last choice of the initial secant matrix is expected to yield better secant matrices B^k , in the sense that

these matrices B^k are expected to resemble tangent stiffness matrices more closely than the ones we would obtain after $B^0 = I$.

Assume then that B^0 is the true tangent stiffness matrix at the beginning of the increment. If we want to use the inverse version of the Broyden method, Eq. (A.21), matrix K^0 should be inverted in order to define H^0 . A more efficient strategy is to rewrite the computation of the prediction s^0

$$s^0 = \Delta u^1 = -H^0 r^0$$

into the linear system

$$K^0 s^0 = K^0 \Delta u^1 = -r^0.$$

At this point, the algorithmic convenience of Eq. (A.21) becomes clear: since it relates H^k to H^0 , the computation of a generic correction vector s^k , Eq. (A.14), involves a matrix-vector product of the form

$$x = -H^0 r^k,$$

which can be transformed into the linear system

$$K^0 x = -r^k.$$

The advantage of Eq. (A.21) over Eq. (A.20) is then justified by the fact that the same matrix, K^0 , is employed at every iteration.

The final algorithm for this version of the inverse Broyden method is presented in Box A.5, where the original formulation $[u, r]$ has been recovered through Eqs. (A.9) and (A.10). It is assumed, as usual, that $\delta u^1 = \Delta u^1$. Also, two auxiliary vectors t and v are defined.

- 1.- Compute K^0
- 2.- Solve the linear system $K^0 \Delta u^1 = -r^0$
- 3.- Update $u^1 = u^0 + \Delta u^1$
- 4.- Evaluate $r^1 = r(u^1)$
- 5.- Convergence control: if point u^1 is good enough, exit.
- $k \geq 1$
- 6.- Solve the linear system $K^0 v = -r^k$
- 7.- $k = 1$ assign $t = v$
 $k > 1$ compute $t = \prod_{i=k-1}^1 (I + \omega^i (\delta u^i)^T) v$
- 8.- Compute and store $\omega^k = \frac{1}{(\delta u^k)^T (\delta u^k - t)}$
- 9.- Compute and store $\delta u^{k+1} = t + ((\delta u^k)^T t) \omega^k$
- 10.- Update $u^{k+1} = u^k + \delta u^{k+1}$
- 11.- Evaluate $r^{k+1} = r(u^{k+1})$
- 12.- Convergence control: if point u^{k+1} is good enough, exit.
- 13.- Assign $k = k + 1$ and go back to 6.-

Box A.5 Algorithm for the Inverse Broyden method

Remark A.4: All linear systems are solved with matrix K^0 , see step 6.-. Thus, only one tangent stiffness matrix needs to be computed and factorized per increment.

Remark A.5: Step 7.- of the algorithm does not require the computation of matrix $\prod_{i=k-1}^1 (I + \omega^i (\delta u^i)^T)$, since the matrix-vector products of the form $(I + a b^T)c$ can be computed as $c + (b^T c)a$ (only scalar products and scalar-vector products). As a matter of fact, apart from the linear systems of steps 2.- and 6.-, all the operations involved in Box A.5 are vectorial operations.

Remark A.6: For each iteration k , the computational cost consists approximately, (Engelman *et al.*, 1981; Soria, 1990), in

- evaluating one vector of out-of-balance forces and solving one linear system (the same as in mNR)
- recovering $2k$ vectors from storage ($\omega^i, \delta u^i$ for $i = 1, \dots, k$).
- computing $2k$ scalar products.

Therefore, the total cost of iteration k increases with k , which means that this method will only be competitive if convergence is attained within a small number of iterations per increment.

Remark A.7: Under certain conditions, (Dennis & Schnabel, 1983), this method shows a superlinear rate of convergence, i.e., a succession $\{\gamma_k\}$ such that $\lim_{k \rightarrow \infty} \gamma_k = 0$ can be found for which

$$\|u^{k+1} - u^*\| \leq \gamma_k \|u^k - u^*\|,$$

where u^* denotes the solution corresponding to the current increment.

Symmetrical rank-one update

As it has already been indicated, it is advisable in general to iterate with a secant matrix as similar to the tangent stiffness matrix as possible. In many applications, tangent matrices are symmetrical (see Section 3.5). It seems thus interesting to begin with a symmetrical B^0 and transfer this property at every iteration. Therefore, to impose hereditary symmetry as

$$B^{k-1} \text{ symmetrical} \implies B^k \text{ symmetrical} \quad (\text{A.24})$$

seems a reasonable choice.

It can be proved, (Dennis & Moré, 1977), that the direct Quasi-Newton equation (A.11) together with (A.24) leads to the following update formula for B^k :

$$B^k = B^{k-1} + \frac{(y^{k-1} - B^{k-1} s^{k-1})(y^{k-1} - B^{k-1} s^{k-1})^T}{(y^{k-1} - B^{k-1} s^{k-1})^T s^{k-1}}. \quad (\text{A.25})$$

Equation (A.25) is known as the symmetrical rank-one formula. According to Dennis & Moré (1977), this method shows a poor performance and is not frequently employed.

Rank-two updates

Direct rank-two updates

In many applications, the tangent stiffness matrices are not only symmetrical but also positive definite. Thus, it is a reasonable strategy to initialize a Quasi-Newton method with a symmetrical positive definite (SPD) B^0 and transfer these properties at every iteration. That is, we want to satisfy the Quasi-Newton equation (A.11), the symmetry condition (A.24) and

$$B^{k-1} \text{ positive definite} \implies B^k \text{ positive definite.} \quad (\text{A.26})$$

The only rank-one update with hereditary symmetry is the one shown in Eq. (A.25), so there is a need for more complicated updates if condition (A.26) must be verified as well.

Rank-two updates, where B^k is obtained by adding to B^{k-1} two rank-one matrices instead of one, fulfill this need. In particular, a family of rank-two updates is defined by

$$B^k = B^{k-1} + \frac{(y^{k-1} - B^{k-1} s^{k-1})(c^{k-1})^T + c^{k-1}(y^{k-1} - B^{k-1} s^{k-1})^T}{(s^{k-1})^T c^{k-1}} - \frac{(y^{k-1} - B^{k-1} s^{k-1})^T s^{k-1}}{((c^{k-1})^T s^{k-1})^2} c^{k-1} (c^{k-1})^T, \quad (\text{A.27})$$

where c^{k-1} represents any vector belonging to \mathbb{R}^n . Equation (A.27) satisfies both the Quasi-Newton equation (A.11) and the hereditary symmetry condition (A.24), (Dennis & Moré, 1977). It has the interesting property that vector c^{k-1} is arbitrary (for instance, for $c^{k-1} = y^{k-1} - B^{k-1} s^{k-1}$ the rank-one symmetrical update (A.25) is recovered).

If it is required for matrix B^k in (A.27) to be positive definite if B^{k-1} is positive definite, that is, if conditions (A.11), (A.24) and (A.26) must be satisfied simultaneously, a natural choice for c^{k-1} is y^{k-1} , (Dennis & Moré, 1977).

Substituting

$$\mathbf{c}^{k-1} = \mathbf{y}^{k-1} \quad (\text{A.28})$$

into Eq. (A.27) and rearranging the resulting expression of B^k , the Davidon-Fletcher-Powell compact update formula is obtained, (Fletcher, 1987b),

$$B_{DFP}^k = \left[I - \frac{\mathbf{y}^{k-1} (\mathbf{s}^{k-1})^T}{(\mathbf{s}^{k-1})^T \mathbf{y}^{k-1}} \right] B^{k-1} \left[I - \frac{\mathbf{s}^{k-1} (\mathbf{y}^{k-1})^T}{(\mathbf{s}^{k-1})^T \mathbf{y}^{k-1}} \right] + \frac{\mathbf{y}^{k-1} (\mathbf{y}^{k-1})^T}{(\mathbf{s}^{k-1})^T \mathbf{y}^{k-1}}, \quad (\text{A.29})$$

which describes the **DFP method**.

Inverse rank-two updates

In an inverse Quasi-Newton context, it is possible to impose hereditary symmetry and positive definiteness on the inverse of matrix B^k ,

$$H^{k-1} \text{ symmetrical} \implies H^k \text{ symmetrical} \quad (\text{A.30})$$

$$H^{k-1} \text{ positive definite} \implies H^k \text{ positive definite} \quad (\text{A.31})$$

After some algebra, (Brodie *et al.*, 1972), the Broyden-Fletcher-Goldfarb-Shanno update formula for H can be obtained

$$H_{BFGS}^k = \left[I - \frac{\mathbf{s}^{k-1} (\mathbf{y}^{k-1})^T}{(\mathbf{y}^{k-1})^T \mathbf{s}^{k-1}} \right] H^{k-1} \left[I - \frac{\mathbf{y}^{k-1} (\mathbf{s}^{k-1})^T}{(\mathbf{y}^{k-1})^T \mathbf{s}^{k-1}} \right] + \frac{\mathbf{s}^{k-1} (\mathbf{s}^{k-1})^T}{(\mathbf{y}^{k-1})^T \mathbf{s}^{k-1}}, \quad (\text{A.32})$$

which describes the **BFGS method**.

It is interesting to remark, by comparing Eqs. (A.29) and (A.32), that one can be obtained from the other simply by interchanging

$$\mathbf{s} \longleftrightarrow \mathbf{y}, \quad B \longleftrightarrow H.$$

In this sense, Eqs. (A.29) and (A.32) are said to be **dual** or also **complementary**.

The detailed algorithm for this method (Brodie *et al.*, 1972; Matthies & Strang, 1979; Soria, 1990), is listed in Box A.6.

- 1.- Compute K^0
 - 2.- Solve the linear system $K^0 \Delta u^1 = -r^0$
 - 3.- Update $u^1 = u^0 + \Delta u^1$
 - 4.- Evaluate $r^1 = r(u^1)$
 - 5.- Convergence control: if point u^1 is good enough, exit.
- $k \geq 1$
- 6.- Compute and store $\omega^k = \frac{\delta u^k}{(\delta u^k)^T (r^{k-1} - r^k)}$
 - 7.- Compute and store $v^k = r^k - \left[1 + \sqrt{\frac{(\delta u^k)^T (r^{k-1} - r^k)}{(\delta u^k)^T r^{k-1}}} \right] r^{k-1}$
 - 8.- Compute $p = \prod_{i=1}^k (I + v^i (\omega^i)^T) r^k$
 - 9.- Solve the linear system $K^0 q = -p$
 - 10.- Compute and store $\delta u^{k+1} = \prod_{i=k}^1 (I + \omega^i (v^i)^T) q$
 - 11.- Update $u^{k+1} = u^k + \delta u^{k+1}$
 - 12.- Evaluate $r^{k+1} = r(u^{k+1})$
 - 13.- Convergence control: if point u^{k+1} is good enough, exit.
 - 14.- Assign $k = k + 1$ and go back to 6.-

Box A.6 Algorithm for the BFGS method

Remark A.8: As explained in Remark A.5 for the Inverse Broyden method, step 10.- does **not** require the computation of matrix $I + \omega^i (\delta v^i)^T$.

Remark A.9: For each iteration k , we need to solve one linear system and evaluate one vector of out-of-balance forces, recover $2k$ vectors from storage and compute $4k$ scalar products, (Brodlić *et al.*, 1972; Soria, 1990).

The cost of a BFGS iteration is therefore higher than that of a Broyden iteration. However, we should expect this difference in cost to be compensated by a difference in the quality of the matrices: assuming that tangent stiffness matrices are SPD, BFGS iteration matrices, which are also SPD, are supposed to be better –that is, more similar to the true tangent matrices– than Broyden matrices (nor symmetrical neither positive definite).

Remark A.10: The BFGS method shows the same kind of superlinear convergence as the Broyden method, see Dennis & Moré (1977).

A.3 Secant–Newton methods

A.3.1 Secant-related acceleration techniques

As commented in Section A.2, a generic iteration k of a Quasi–Newton method involves the evaluation of one vector of residual forces and the resolution of one linear system of equations (as in mNR), plus the recovery of $2k$ vectors from storage and the computation of $2k$ (Broyden)/ $4k$ (BFGS) scalar products, (Brodlić *et al.*, 1972; Engelman *et al.*, 1981; Soria, 1990).

The increasing cost of the iterations is indeed a handicap of the Quasi–Newton methods in the sense that, if convergence is not reached within a small number of iterations, the total cost of the method may turn up to be prohibitive. However, the use of secant matrices has been shown to supply a very good alternative to tangent matrices.

Thus the so-called **Secant-related acceleration techniques** or simply **Secant–Newton methods** (SN), see Crisfield (1991), appeared as a modification of the Quasi–Newton scheme with a **constant computational cost per iteration**. The keyword ‘acceleration’ indicates that SN iterations are computed faster (indeed, Quasi–Newton iterations have an increasing cost, whereas for Secant–Newtons the cost is kept constant and equal to that of the first iteration).

In all direct/inverse Quasi–Newton methods, matrix B^k/H^k can be obtained as a function of B^{k-1}/H^{k-1} and other parameters, Eqs. (A.16), (A.19), (A.25), (A.29),

(A.32). In the corresponding Secant–Newton version, matrix B^k/H^k is built as if the previous secant matrix had never been updated. In other words, B^{k-1}/H^{k-1} is replaced with B^0/H^0 in the QN update formula that gives the value for matrix B^k/H^k . This is equivalent to a Quasi–Newton without memory where only the last correction is recalled.

In the following Subsection, the Inverse Broyden and BFGS methods in their Secant–Newton form will also be presented.

A.3.2 Algorithms for Secant Inverse Broyden and Secant BFGS methods

Secant Inverse Broyden method

Recalling the original update formula for the Quasi–Newton Inverse Broyden method, Eq. (A.19), the Secant–Newton version of this method is obtained after exchanging H^{k-1} for H^0 , which yields

$$H^k = H^0 + \frac{(s^{k-1} - H^0 y^{k-1})(s^{k-1})^T H^0}{(s^{k-1})^T (H^0 y^{k-1})}. \quad (\text{A.33})$$

Thus, vector s^k is obtained by postmultiplying Eq. (A.33) by $(-r^k)$,

$$\begin{aligned} s^k &= H^k (-r^k) = \left[H^0 + \frac{(s^{k-1} - H^0 y^{k-1})(s^{k-1})^T H^0}{(s^{k-1})^T (H^0 y^{k-1})} \right] (-r^k) = \\ &= H^0 (-r^k) + \frac{(s^{k-1} - H^0 y^{k-1})(s^{k-1})^T H^0}{(s^{k-1})^T (H^0 y^{k-1})} (s^{k-1})^T H^0 (-r^k). \end{aligned} \quad (\text{A.34})$$

We can now introduce

$$\hat{s}^k = H^0 (-r^k), \quad (\text{A.35})$$

which is in fact the mNR correction vector.

Using Eq. (A.35) and recalling the definition given in Eq. (A.8) for y^{k-1} we have that

$$H^0 y^{k-1} = H^0 (r^k - r^{k-1}) = H^0 r^k - H^0 r^{k-1} = \hat{s}^{k-1} - \hat{s}^k. \quad (\text{A.36})$$

Replacing now Eqs. (A.35) and (A.36) into Eq. (A.34), we obtain the following expression for s^k , (Soria, 1990):

$$s^k = \tilde{s}^k + \frac{s^{k-1} - (\tilde{s}^{k-1} - \tilde{s}^k)}{(s^{k-1})^T (\tilde{s}^{k-1} - \tilde{s}^k)} (s^{k-1})^T \tilde{s}^k, \quad (\text{A.37})$$

which can be rearranged to yield

$$s^k = (1 + \rho) \tilde{s}^k + \rho s^{k-1} - \rho \tilde{s}^{k-1}, \quad (\text{A.38})$$

where

$$\rho = \frac{(s^{k-1})^T \tilde{s}^k}{\tau} \quad (\text{A.39})$$

and

$$\tau = (s^{k-1})^T (\tilde{s}^{k-1} - \tilde{s}^k). \quad (\text{A.40})$$

From Eq. (A.38) it is clear that the correction vector s^k will be obtained by performing a linear combination of vectors \tilde{s}^k , s^{k-1} , \tilde{s}^{k-1} , which correspond respectively, Eq. (A.35), to the previous Broyden correction vector, the previous mNR correction vector and the current mNR correction vector. Notice that only two vectors (s^{k-1} , \tilde{s}^{k-1}) will need to be transferred from one iteration to the next and that just two scalar products are performed to obtain τ and ρ .

The complete algorithm for this method is listed in Box A.7. As in the QN version, if the first secant matrix B^0 is chosen as the tangent stiffness matrix K^0 , the computation of the inverse matrix H^0 should be avoided. This is accomplished by solving linear systems with matrix K^0 rather than performing direct products with matrix $H^0 = (K^0)^{-1}$.

Apart from this, the original context $[u, r]$ is recovered in Box A.7. For that purpose, the notation

$$\delta \tilde{u}^k = \tilde{s}^{k-1} \quad (\text{A.41})$$

is introduced.

- 1.- Compute K^0
- 2.- Solve the linear system $K^0 \Delta u^1 = -r^0$
- 3.- Assign $\delta \tilde{u}^1 = \Delta u^1$
- 4.- Update $u^1 = u^0 + \Delta u^1$
- 5.- Evaluate $r^1 = r(u^1)$
- 6.- Convergence control: if point u^1 is good enough, exit.
- $k \geq 1$
- 7.- Solve the linear system $K^0 \delta \tilde{u}^{k+1} = -r^k$
- 8.- Compute $\tau = (\delta u^k)^T (\delta \tilde{u}^k - \delta \tilde{u}^{k+1})$
- 9.- Compute $\rho = \frac{(\delta u^k)^T \delta \tilde{u}^{k+1}}{\tau}$
- 10.- Compute $\delta u^{k+1} = (1 + \rho) \delta \tilde{u}^{k+1} + \rho \delta u^k - \rho \delta \tilde{u}^k$
- 11.- Update $u^{k+1} = u^k + \delta u^{k+1}$
- 12.- Evaluate $r^{k+1} = r(u^{k+1})$
- 13.- Convergence control: if point u^{k+1} is good enough, exit.
- 14.- Assign $k = k + 1$ and go back to 7.-

Box A.7 Algorithm for the Secant Inverse Broyden method

Remark A.11: For each iteration k , we need to evaluate one vector of out-of-balance forces and solve one linear system of equations (the same as in mNR), recover two vectors from storage and compute two scalar products. Therefore, the cost of a Secant Inverse Broyden iteration is constant, as it was intended from the beginning.

Remark A.12: No rate-of-convergence properties are known for the Secant-Newton methods.

Remark A.13: If ρ is equal to zero in Eq. (A.38), the following value for the correction vector is obtained

$$\mathbf{s}^k = \tilde{\mathbf{s}}^k = \mathbf{H}^0 (-\mathbf{r}^k) = (\mathbf{K}^0)^{-1} (-\mathbf{r}^k),$$

which corresponds to the modified Newton–Raphson method. In this sense, the SN Inverse Broyden method can also be regarded as a refinement of a mNR in which a weighted linear combination of the mNR correction vector $\tilde{\mathbf{s}}^k$ together with vectors \mathbf{s}^{k-1} and $\tilde{\mathbf{s}}^{k-1}$ is performed.

Secant BFGS method

The Secant BFGS (or BFGSS) method is obtained by exchanging matrix \mathbf{H}^{k-1} for \mathbf{H}^0 in the QN BFGS update formula, (Crisfield, 1991),

$$\mathbf{H}_{BFGSS}^k = \left[\mathbf{I} - \frac{\mathbf{s}^{k-1} (\mathbf{y}^{k-1})^T}{(\mathbf{y}^{k-1})^T \mathbf{s}^{k-1}} \right] \mathbf{H}^0 \left[\mathbf{I} - \frac{\mathbf{y}^{k-1} (\mathbf{s}^{k-1})^T}{(\mathbf{y}^{k-1})^T \mathbf{s}^{k-1}} \right] + \frac{\mathbf{s}^{k-1} (\mathbf{s}^{k-1})^T}{(\mathbf{y}^{k-1})^T \mathbf{s}^{k-1}}. \quad (\text{A.42})$$

Using Eq. (A.42), the correction vector \mathbf{s}^k can be computed as

$$\begin{aligned} \mathbf{s}^k = \mathbf{H}^k (-\mathbf{r}^k) &= \left[\mathbf{I} - \frac{\mathbf{s}^{k-1} (\mathbf{y}^{k-1})^T}{(\mathbf{y}^{k-1})^T \mathbf{s}^{k-1}} \right] \mathbf{H}^0 \left[\mathbf{I} - \frac{\mathbf{y}^{k-1} (\mathbf{s}^{k-1})^T}{(\mathbf{y}^{k-1})^T \mathbf{s}^{k-1}} \right] (-\mathbf{r}^k) + \\ &+ \frac{\mathbf{s}^{k-1} (\mathbf{s}^{k-1})^T}{(\mathbf{y}^{k-1})^T \mathbf{s}^{k-1}} (-\mathbf{r}^k). \end{aligned} \quad (\text{A.43})$$

Equation (A.43) can be manipulated in a similar way as in the Secant Broyden method. The final result of this manipulation is, using the previous definition of $\tilde{\mathbf{s}}^k$, Eq. (A.35),

$$\mathbf{s}^k = (1 + C) \tilde{\mathbf{s}}^k + (-C) \tilde{\mathbf{s}}^{k-1} + (C - (1 + C)B + CA) \mathbf{s}^{k-1}, \quad (\text{A.44})$$

with A , B , C being the three following scalar values:

$$A = \frac{(\mathbf{y}^{k-1})^T \tilde{\mathbf{s}}^{k-1}}{(\mathbf{s}^{k-1})^T \mathbf{y}^{k-1}} \quad (\text{A.45})$$

$$B = \frac{(\mathbf{y}^{k-1})^T \tilde{\mathbf{s}}^k}{(\mathbf{s}^{k-1})^T \mathbf{y}^{k-1}} \quad (\text{A.46})$$

$$C = \frac{(\mathbf{y}^{k-1})^T \tilde{\mathbf{r}}^k}{(\mathbf{s}^{k-1})^T \mathbf{y}^{k-1}}. \quad (\text{A.47})$$

From Eqs. (A.43) to (A.47) it can be seen that a Secant BFGS update involves the storage of two vectors and the computation of four scalar products.

Box A.8 contains the step-by-step algorithm of this method.

Remark A.14: For each iteration k , we need to evaluate one vector of out-of-balance forces (the same as in mNR), recover two vectors from storage and compute four scalar products ($\tau_1, \tau_2, \tau_3, \tau_4$).

Remark A.15: The Secant BFGS method can also be interpreted as a refinement of the modified Newton–Raphson method. The mNR correction can be recovered by setting the values of A, B, C in Eq. (A.44) to zero.

Remark A.16: As commented previously, no rate-of-convergence properties are known for the Secant–Newton methods.

Equation (A.44) may be further simplified into the linear combination of only two or one correction vectors. The resulting methods, developed by Crisfield, are known respectively as SN2, SN1, see Crisfield (1991).

- 1.- Compute K^0
- 2.- Solve the linear system $K^0 \Delta u^1 = -r^0$
- 3.- Assign $\delta \tilde{u}^1 = \Delta u^1$
- 4.- Update $u^1 = u^0 + \Delta u^1$
- 5.- Evaluate $r^1 = r(u^1)$
- 6.- Convergence control: if point u^1 is good enough, exit.

$$k \geq 1$$

- 7.- Solve the linear system $K^0 \delta \tilde{u}^{k+1} = -r^k$
- 8.- Compute

$$\tau_1 = (r^{k-1} - r^k)^T \delta \tilde{u}^k$$

$$\tau_2 = (r^{k-1} - r^k)^T \delta \tilde{u}^{k+1}$$

$$\tau_3 = (\delta u^k)^T r^k$$

$$\tau_4 = (\delta u^k)^T (r^{k-1} - r^k)$$

- 9.- Compute

$$A = \frac{\tau_1}{\tau_4}, \quad B = \frac{\tau_2}{\tau_4}, \quad C = \frac{\tau_3}{\tau_4}$$

- 10.- Compute

$$\delta u^{k+1} = (1 + C) \delta \tilde{u}^{k+1} + (-C) \delta \tilde{u}^k + (C - (1 + C)B + CA) \delta u^k$$

- 11.- Update $u^{k+1} = u^k + \delta u^{k+1}$
- 12.- Evaluate $r^{k+1} = r(u^{k+1})$
- 13.- Convergence control: if point u^{k+1} is good enough, exit.
- 14.- Assign $k = k + 1$ and go back to 7.-

Box A.8 Algorithm for the Secant BFGS method

A.4 Line searches

A.4.1 Theory and detailed flowchart

So far, several methods have been discussed that supply different solution techniques for nonlinear problems. In this Section, we will be concerned about performing simple modifications of those solutions to yield better results.

Assume that $\delta\mathbf{u}^{k+1}$ is the correction vector supplied by some NR, QN or SN method at iteration k . Up until this point, displacement updates have been performed following the expression

$$\mathbf{u}^{k+1} = \mathbf{u}^k + \delta\mathbf{u}^{k+1}. \quad (\text{A.48})$$

What we do now is keep the advance direction determined by $\delta\mathbf{u}^{k+1}$, but modify its modulus so that a different update

$$\mathbf{u}^{k+1} = \mathbf{u}^k + \eta_{k+1} \delta\mathbf{u}^{k+1} \quad (\text{A.49})$$

is obtained. The scale factor η_{k+1} in Eq. (A.49) is the advance length, (Crisfield, 1991; Matthies & Strang, 1979; Soria, 1990), which has been implicitly set to 1 in the previous Sections. Obviously, we want to compute η_{k+1} so that vector \mathbf{u}^{k+1} in Eq. (A.49) be better than the one in Eq. (A.48). In other words, we want to search for a better solution \mathbf{u}^{k+1} along the line determined by point \mathbf{u}^k and the direction $\delta\mathbf{u}^{k+1}$.

As for all nonlinear strategies, the main target is to find a displacement vector \mathbf{u}^{k+1} for which $\mathbf{r}(\mathbf{u}^{k+1}) = \mathbf{r}(\mathbf{u}^k + \delta\mathbf{u}^{k+1}) = \mathbf{0}$. In the line-search context, since \mathbf{u}^k and $\delta\mathbf{u}^{k+1}$ are fixed, all functions of \mathbf{u}^{k+1} can be rewritten, after Eq. (A.49), as functions of η_{k+1} . Thus, the problem is reduced to finding the value of the scalar η_{k+1} for which

$$\mathbf{r}(\eta_{k+1}) = \mathbf{r}(\mathbf{u}^k + \eta_{k+1} \delta\mathbf{u}^{k+1}) = \mathbf{0}. \quad (\text{A.50})$$

However, in general the real solution \mathbf{u}^{k+1} to $\mathbf{r}(\mathbf{u}^{k+1}) = \mathbf{0}$ does not lie on the search line. This means that we can expect to minimize the vectorial function of residual forces given in Eq. (A.50), but not to cancel it.

A typical alternative requirement consists of finding the value of η_{k+1} that gives the minimum absolute value of the function ψ defined by

$$\psi(\eta_{k+1}) = (\delta \mathbf{u}^{k+1})^T \mathbf{r}(\mathbf{u}^k + \eta_{k+1} \delta \mathbf{u}^{k+1}). \quad (\text{A.51})$$

The expression of ψ given in Eq. (A.51) is a linearized version of the **total potential energy** associated to point $\mathbf{u}^k + \eta_{k+1} \delta \mathbf{u}^{k+1}$, (Crisfield, 1991). This means that the vectorial problem of minimizing the out-of-balance forces in Eq. (A.50) is transformed into the scalar problem of minimizing the energy function (A.51).

Almost any minimization technique can be used to obtain η_{k+1} for which $|\psi(\eta_{k+1})|$ is the least possible, (Fletcher, 1987b). Nevertheless, we want to use a **simple** strategy, since the evaluation of function ψ involves the evaluation of a vector of residual forces, and this is an expensive operation from a computational point of view; furthermore, more accurate values of η_{k+1} have been shown to give smaller but not zero norms of the residual forces. For this reason, most of the line-search procedures, (Crisfield, 1991; Zienkiewicz & Taylor, 1991), just demand for the modulus of $\psi(\eta_{k+1})$ to be **small** in comparison to the modulus of $\psi(0)$, i.e.,

$$|\psi(\eta_{k+1})| < \varepsilon |\psi(0)|, \quad (\text{A.52})$$

with ε being the **line-search tolerance**. This tolerance is set to large values that range from 0.5, see Matthies & Strang (1979), to 0.8, see Crisfield (1991), thus not making Eq. (A.52) a restrictive condition.

To find an η_{k+1} that meets Eq. (A.52), it seems interesting to take profit of the fact that the values of $\psi(0)$ and $\psi(1)$ are almost known:

- For $\eta_{k+1} = 0$, the residual forces equal $\mathbf{r}(\eta_{k+1}) = \mathbf{r}(0) = \mathbf{r}(\mathbf{u}^k) = \mathbf{r}^k$, where vector \mathbf{r}^k can be obtained from the previous iteration.
- For $\eta_{k+1} = 1$, we have $\mathbf{r}(\eta_{k+1}) = \mathbf{r}(1) = \mathbf{r}(\mathbf{u}^{k+1}) = \mathbf{r}^{k+1}$, which needs to be calculated anyhow at the end of iteration $k+1$.

Therefore, to obtain the values of

$$\psi(0) = (\delta \mathbf{u}^{k+1})^T \mathbf{r}^k \quad (\text{A.53})$$

$$\psi(1) = (\delta \mathbf{u}^{k+1})^T \mathbf{r}^{k+1}, \quad (\text{A.54})$$

only two scalar products need to be computed.

For a start, it is advisable to check if condition (A.52) is satisfied for $\eta_{k+1} = 1$. If this is not the case, a simple interpolation in the plane (η, ψ) is performed between points $\eta_{k+1} = 0$ and $\eta_{k+1} = 1$, Figure A.6. This results in

$$\eta_{k+1,1} = \frac{\psi(0)}{\psi(0) - \psi(1)} \quad (\text{A.55})$$

where the additional subindex 1 in $\eta_{k+1,1}$ indicates that Eq. (A.55) is a prediction of η_{k+1} .

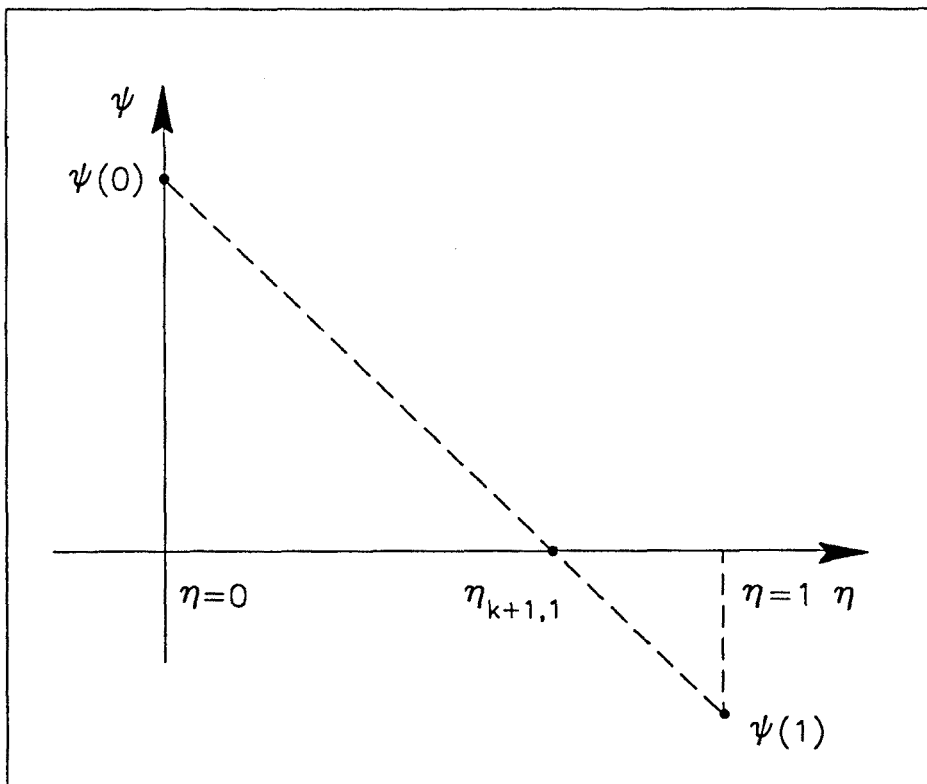


Figure A.6 Line-search interpolation procedure

A common approach is to compute $\eta_{k+1,1}$ from Eq. (A.55), update displacements as $\mathbf{u}^{k+1} = \mathbf{u}^k + \eta_{k+1,1} \delta \mathbf{u}^{k+1}$ and pass on to the next iteration $k+2$. However, if condition (A.52)

$$|\psi(\eta_{k+1,1})| < \varepsilon |\psi(0)|$$

is not met, more 'line-search iterations' can be performed by interpolating between points $(0, \psi(0))$ and $(\eta_{k+1,1}, \psi(\eta_{k+1,1}))$, then between points $(0, \psi(0))$ and $(\eta_{k+1,2}, \psi(\eta_{k+1,2}))$, and so on. In this case, the value for the step-length $\eta_{k+1,i}$ associated to line-search iteration i is computed as

$$\eta_{k+1,i} = \eta_{k+1,i-1} \frac{\psi(0)}{\psi(0) - \psi(\eta_{k+1,i-1})}. \quad (\text{A.56})$$

If the ratio $\psi(\eta_{k+1,i-1})/\psi(0)$ is positive, that is, if $\psi(0)$ and $\psi(\eta_{k+1,i-1})$ have the same sign, an **extrapolation** instead of an interpolation is automatically carried out. This extrapolation can give very large values of $\eta_{k+1,i}$, which in general will not lead to good results. To avoid this situation, an upper bound for η_{k+1} must be specified.

In a similar way, for large negative values of $\psi(\eta_{k+1,i-1})/\psi(0)$, Eq. (A.56) will produce an almost null value of $\eta_{k+1,i+1}$. This is also a bad situation, since point \mathbf{u}^{k+1} will be placed very near to \mathbf{u}^k and thus the algorithm will not progress. Consequently, we also specify a lower bound for η_{k+1} .

If a fixed number of line-search iterations is not sufficient to verify the line-search convergence condition (A.52), the following update is performed

$$\mathbf{u}^{k+1} = \mathbf{u}^k + \eta_{k+1,\text{opt}} \delta \mathbf{u}^{k+1} \quad (\text{A.57})$$

and the procedure passes on to the next iteration $k+2$. The term $\eta_{k+1,\text{opt}}$ in Eq. (A.57) represents the **optimum** value of η_{k+1} , that is, the one that yields the least ratio $|\psi(\eta_{k+1})| / |\psi(0)|$.

Figure A.7 gives the flowchart of a complete line search.

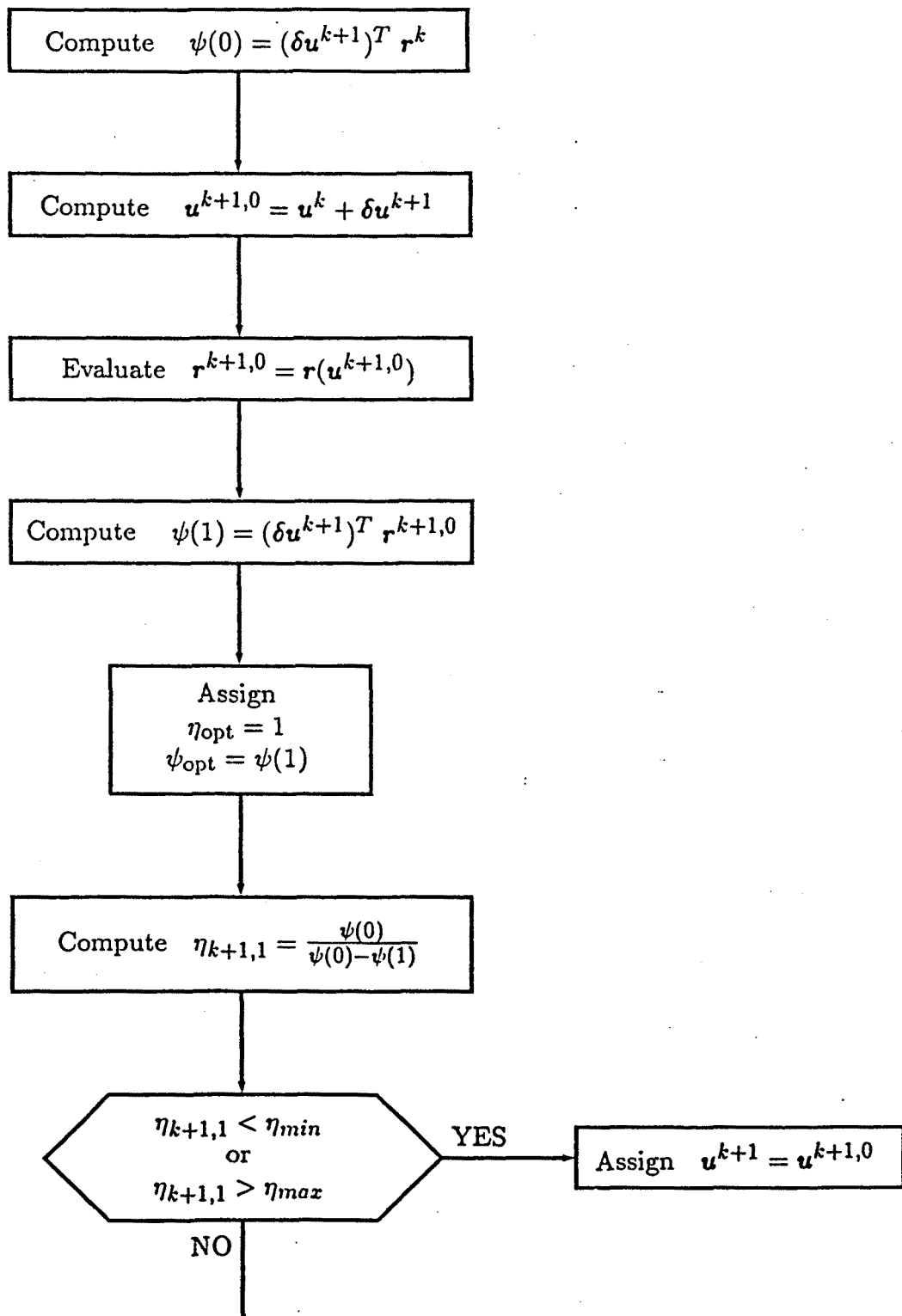


Figure A.7 Flowchart for the line-search procedure

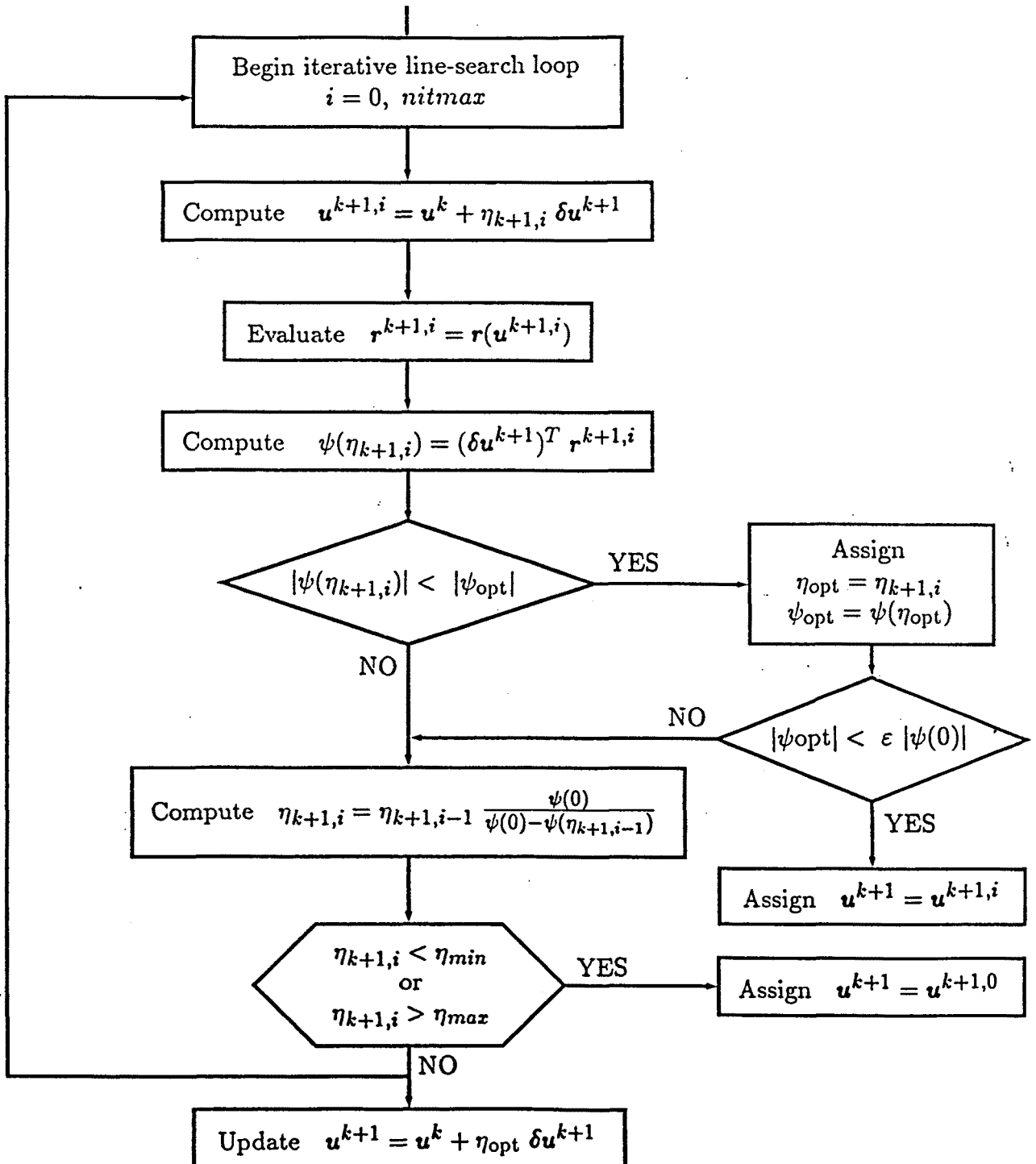


Figure A.7 Flowchart for the line-search procedure (continued)

A.5 Arc-length methods

A.5.1 Some words about load control and the need for continuation techniques

In Section A.1, the incremental/iterative solution for nonlinear problems was introduced. This procedure consists of splitting up the total load \mathbf{f}_{ext} into N increments of load ${}^n\Delta\mathbf{f}$, so that

$$\mathbf{f}_{\text{ext}} = {}^1\Delta\mathbf{f} + {}^2\Delta\mathbf{f} + \dots + {}^{N-1}\Delta\mathbf{f} + {}^N\Delta\mathbf{f} = {}^N\mathbf{f}. \quad (\text{A.58})$$

Once this fragmentation is performed, we only need to focus on each load step. Indeed, all the discussed methods deal with the calculation of ${}^{n+1}\mathbf{u}$ for which $\mathbf{f}_{\text{int}}({}^{n+1}\mathbf{u}) - {}^{n+1}\mathbf{f} = \mathbf{0}$, given that ${}^n\mathbf{u}$ is an output from the previous increment. This is also known as a **load control** procedure.

This Section starts with an exposition of some cases in which the decomposition of the load described in Eq. (A.58) may not provide a good solution to the problem. Later on, some alternative techniques will be presented.

Figure A.8 shows a typical “brittle collapse” load-displacement curve: both load and displacement increase until a **limit point** A is achieved; beyond this point, displacement goes on growing but load starts to decrease. When using a load control procedure, solutions for all loads below the limit load will be obtained, but the algorithm will fail to find a solution past point A.

If a pre-collapse analysis of the solid is made, the previous indicators may precipitate the conclusion of a limit point being passed by. However, this is a dangerous interpretation to make, since a failure of the algorithm may be also due to a bad choice of the load increment, etc.

On the contrary, we may want to follow the equilibrium path of the structure; for instance, because it is integrated in some bigger structure, or because we believe that point A is just a local maximum and not the ultimate load, or even because we want to make sure that we have gone past a limit point. In this case, a load control procedure will be of no use.

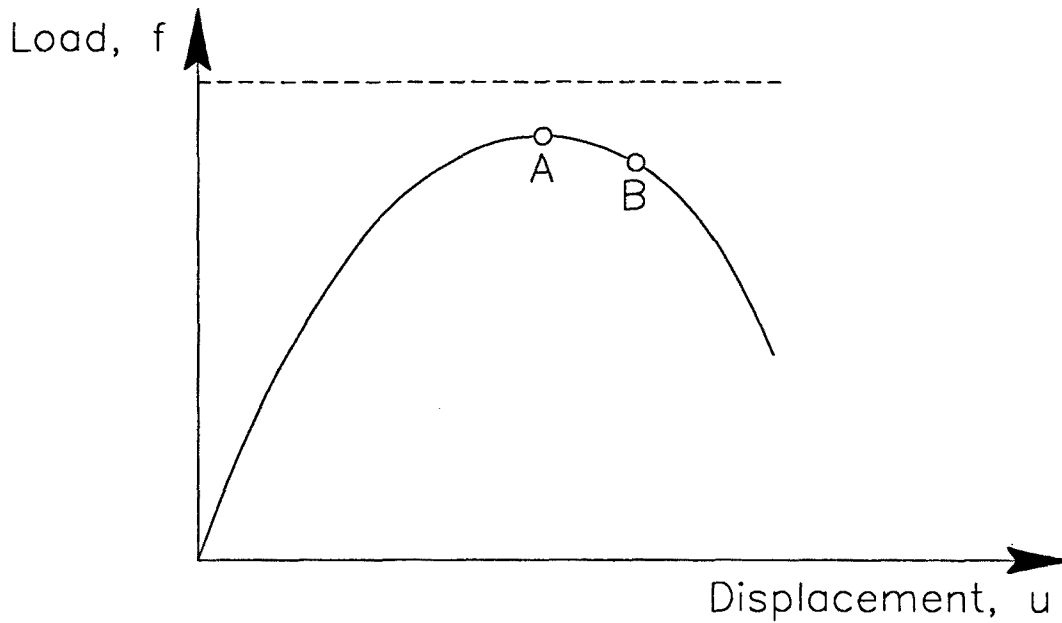
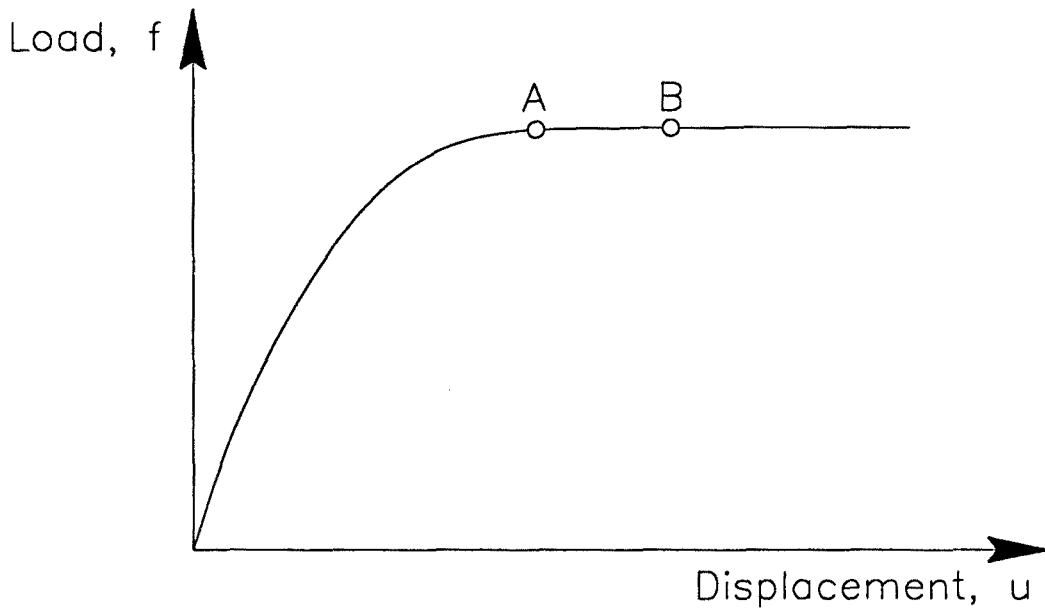
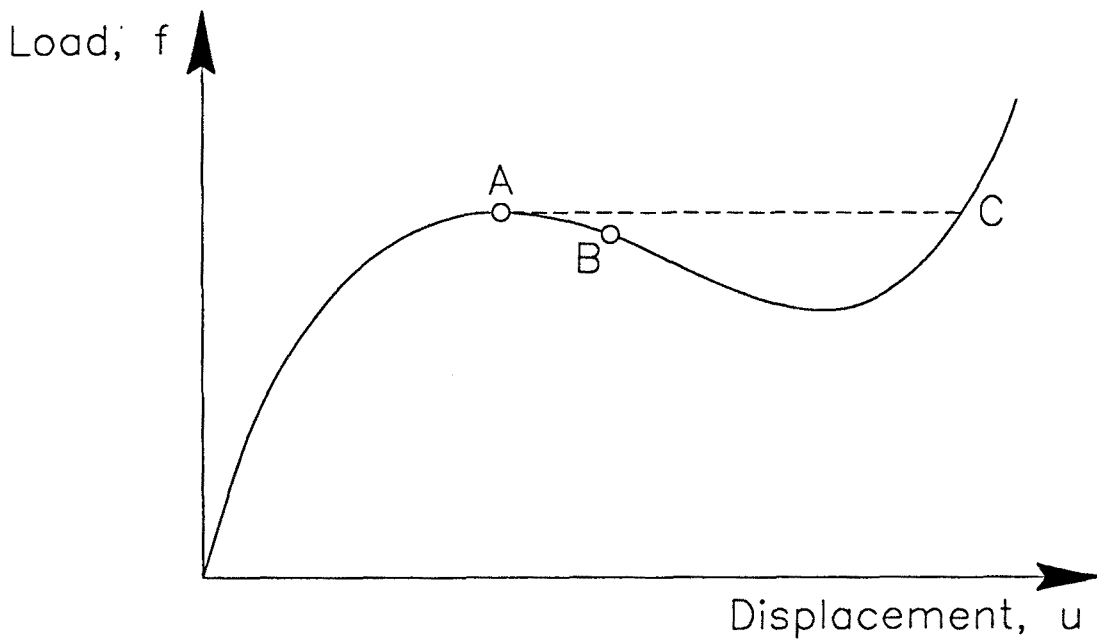


Figure A.8 Brittle collapse

The curve in Figure A.9 describes a **ductile collapse**. This behaviour is identical to the one represented in Figure A.8, except for the fact that the load stays constant instead of lessening beyond point A. As in Figure A.8, if increments of load are applied, the load-displacement curve can be followed up until point A, but no further.

Let us consider now the curve in Figure A.10. In this case, A is a local maximum. If a load control is performed by imposing some small increments of load, we will be able to get close enough to point A. The next increment, if converged, is going to yield point C, passing over everything that happens under the straight line AC. Figure A.10 describes what is known as a **snap-through** response.

To obtain good solutions to the problems illustrated in Figures A.8 to A.10, an alternative technique could be used that consists of applying increments of displacement rather than increments of load. This **displacement control** procedure would follow

**Figure A.9** Ductile collapse**Figure A.10** Snap-through

successfully path AB and beyond, since in this case increasing values of the displacement axis would be used.

In the case of Figure A.11, however, neither of the two procedures, load or displacement control, is good enough. A load control would give the same problems as in Figure A.10, while a displacement control would force loads to jump from A to C, ignoring the snap-back behaviour around point B.

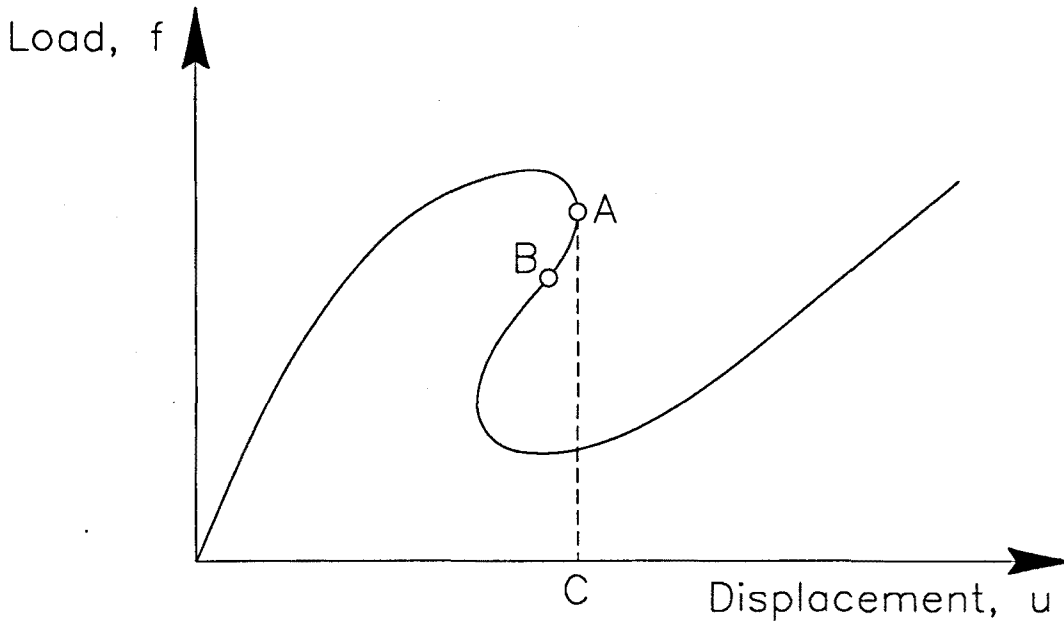


Figure A.11 Snap-back

For cases such as the one in Figure A.11, we need to use the so-called **continuation methods**. These methods allow for the obtention of points on the load-deflection curve by prescribing the **distance** from one point to the next rather than by requiring increasing values of loads or displacements.

A.5.2 The arc-length method. A general formulation including various control strategies

The arc-length method

Arc-length methods were firstly introduced by Riks (1979), and Wempner (1971), and have been thoroughly studied later on by Crisfield (1980, 1983, 1991), and others, see

Cervera (1986), Kouhia & Mikkola (1989), Schweizerhof and Wriggers (1986). As indicated in the previous Subsection, these methods consist of imposing fixed arc lengths between points on the load-displacement curve, so that any structural behaviour can be captured.

We start by defining the total load associated to the current increment as

$${}^{n+1}\mathbf{f} = {}^n\mathbf{f} + {}^{n+1}\Delta\mathbf{f}. \quad (\text{A.59})$$

The increment of load ${}^{n+1}\Delta\mathbf{f}$ is then written as

$${}^{n+1}\Delta\mathbf{f} = {}^{n+1}\Delta\alpha \mathbf{f}_{\text{ext}}, \quad (\text{A.60})$$

where \mathbf{f}_{ext} denotes a fixed referential load and α is the so-called load-level parameter. If ${}^{n+1}\Delta\alpha$ is fixed a priori, then Eq. (A.59) describes a load control procedure. For the arc length method, however, the value of ${}^{n+1}\Delta\alpha$ is unknown at the beginning of the increment.

Let ${}^{n+1}\Delta\mathbf{u}$ be, as usual, the solution associated to the increment of load ${}^{n+1}\Delta\mathbf{f}$. Then, the expression

$$\sqrt{({}^{n+1}\Delta\mathbf{u})^T {}^{n+1}\Delta\mathbf{u} + ({}^{n+1}\Delta\mathbf{f})^T {}^{n+1}\Delta\mathbf{f}} \quad (\text{A.61})$$

gives the distance between points $({}^n\mathbf{u}, {}^n\mathbf{f})$ and $({}^{n+1}\mathbf{u}, {}^{n+1}\mathbf{f})$ in the load-displacement space. Recalling from Eq. (A.60) that ${}^{n+1}\Delta\mathbf{f} = {}^{n+1}\Delta\alpha \mathbf{f}_{\text{ext}}$ and introducing a scale parameter c to weigh the contribution of load and displacement terms, we go on to require for this distance to equal a prescribed arc length $\Delta\ell$

$$\sqrt{({}^{n+1}\Delta\mathbf{u})^T {}^{n+1}\Delta\mathbf{u} + c ({}^{n+1}\Delta\alpha)^2 (\mathbf{f}_{\text{ext}})^T \mathbf{f}_{\text{ext}}} = \Delta\ell. \quad (\text{A.62})$$

The value $c = 0$ leads to the so-called cylindrical arc-length methods; $c = 1$ yields the spherical methods.

The arc-length condition (A.62) involves $n + 1$ unknowns (n for ${}^{n+1}\Delta\mathbf{u}$ and one more for the scalar ${}^{n+1}\Delta\alpha$), which need to be computed from the n equilibrium equations

$$\mathbf{f}_{\text{int}}({}^{n+1}\mathbf{u}) - {}^{n+1}\mathbf{f} = \mathbf{f}_{\text{int}}({}^n\mathbf{u} + {}^{n+1}\Delta\mathbf{u}) - ({}^n\mathbf{f} + {}^{n+1}\Delta\alpha \mathbf{f}_{\text{ext}}) = \mathbf{0} \quad (\text{A.63})$$

together with the additional condition (A.62).

Riks (1971) and Wempner (1971) suggest to obtain those $n + 1$ variables by using an extended Taylor expansion. First, a prediction ${}^{n+1}\Delta\alpha^1$ to ${}^{n+1}\Delta\alpha$ and also a prediction ${}^{n+1}\Delta\mathbf{u}^1$ to ${}^{n+1}\Delta\mathbf{u}$ are computed. For this purpose, the following system of n linear equations and $n + 1$ unknowns must be solved

$${}^{n+1}K^0 {}^{n+1}\Delta\mathbf{u}^1 = {}^{n+1}\Delta\mathbf{f}^1 = {}^{n+1}\Delta\alpha^1 \mathbf{f}_{\text{ext}}. \quad (\text{A.64})$$

Defining a new vector ${}^{n+1}\Delta\mathbf{u}_T$ through the expression

$${}^{n+1}\Delta\mathbf{u}^1 = {}^{n+1}\Delta\alpha^1 {}^{n+1}\Delta\mathbf{u}_T, \quad (\text{A.65})$$

Eq. (A.64) can be rewritten in terms of ${}^{n+1}\Delta\mathbf{u}_T$ as

$${}^{n+1}K^0 {}^{n+1}\Delta\mathbf{u}_T = \mathbf{f}_{\text{ext}}, \quad (\text{A.66})$$

where the T subscript stands for **total** load. The arc-length condition (A.62) can now be applied to the prediction ${}^{n+1}\Delta\mathbf{u}^1$ yielding

$$\sqrt{({}^{n+1}\Delta\mathbf{u}^1)^T {}^{n+1}\Delta\mathbf{u}^1 + c ({}^{n+1}\Delta\alpha^1)^2 (\mathbf{f}_{\text{ext}})^T \mathbf{f}_{\text{ext}}} = \Delta\ell,$$

or also, using ${}^{n+1}\Delta\mathbf{u}_T$ rather than ${}^{n+1}\Delta\mathbf{u}^1$

$$({}^{n+1}\Delta\alpha^1)^2 \left[({}^{n+1}\Delta\mathbf{u}_T)^T {}^{n+1}\Delta\mathbf{u}_T + c (\mathbf{f}_{\text{ext}})^T \mathbf{f}_{\text{ext}} \right] = (\Delta\ell)^2. \quad (\text{A.67})$$

The sequence of computations consists of:

1. Compute ${}^{n+1}\Delta\mathbf{u}_T$ by solving Eq. (A.66).
2. Obtain the value of ${}^{n+1}\Delta\alpha^1$ from Eq. (A.67) as

$${}^{n+1}\Delta\alpha^1 = \pm \frac{\Delta\ell}{\sqrt{({}^{n+1}\Delta\mathbf{u}_T)^T {}^{n+1}\Delta\mathbf{u}_T + c (\mathbf{f}_{\text{ext}})^T \mathbf{f}_{\text{ext}}}}. \quad (\text{A.68})$$

The choice of the sign in Eq. (A.68) will be discussed later on.

After ${}^{n+1}\Delta\mathbf{u}_T$ and ${}^{n+1}\Delta\alpha^1$, a prediction ${}^{n+1}\Delta\mathbf{u}^1$ to the displacements can be obtained from Eq. (A.65).

We can now update displacements

$${}^{n+1}\mathbf{u}^1 = {}^n\mathbf{u} + {}^{n+1}\Delta\mathbf{u}^1 \quad (\text{A.69})$$

and also external forces, computing first the new value of the load level,

$${}^{n+1}\alpha^1 = {}^n\alpha + {}^{n+1}\Delta\alpha^1 \quad (\text{A.70})$$

where ${}^n\alpha$ is such that ${}^n\mathbf{f} = {}^n\alpha \mathbf{f}_{\text{ext}}$. After ${}^{n+1}\alpha^1$ is obtained from Eq. (A.70), the updated load ${}^{n+1}\mathbf{f}^1$ is calculated as

$${}^{n+1}\mathbf{f}^1 = {}^{n+1}\alpha^1 \mathbf{f}_{\text{ext}}. \quad (\text{A.71})$$

In general, $({}^{n+1}\mathbf{u}^1, {}^{n+1}\mathbf{f}^1)$ will not be an equilibrium point. Therefore, we need to iterate on the variables \mathbf{u} and also on the load level α since ${}^{n+1}\Delta\alpha$ is not prescribed a priori, until convergence is achieved.

In a general iteration, we look for a displacement correction vector ${}^{n+1}\delta\mathbf{u}^{k+1}$ such that

$$\mathbf{r}({}^{n+1}\mathbf{u}^{k+1}) = \mathbf{r}({}^{n+1}\mathbf{u}^k + {}^{n+1}\delta\mathbf{u}^{k+1}) = \mathbf{0}. \quad (\text{A.72})$$

Under the arc-length formulation, and dropping $n+1$ left superscripts, Eq. (A.72) yields

$$\mathbf{K}(\mathbf{u}^k) \delta\mathbf{u}^{k+1} = -\mathbf{r}(\mathbf{u}^k) + \delta\alpha^{k+1} \mathbf{f}_{\text{ext}}, \quad (\text{A.73})$$

where K^k can be any iteration matrix depending on the method that is used and the term $\delta\alpha^{k+1} \mathbf{f}_{\text{ext}}$ accounts for the variation of external load due to the iterative correction of the load level α .

Equation (A.73) can be inverted symbolically to give the following expression for $\delta\mathbf{u}^{k+1}$

$$\delta\mathbf{u}^{k+1} = (K(\mathbf{u}^k))^{-1} (-\mathbf{r}(\mathbf{u}^k)) + \delta\alpha^{k+1} (K(\mathbf{u}^k))^{-1} \mathbf{f}_{\text{ext}}. \quad (\text{A.74})$$

If two vectors $\delta\tilde{\mathbf{u}}^{k+1}$ and $\delta\mathbf{u}_T^{k+1}$ are now defined as

$$\delta\tilde{\mathbf{u}}^{k+1} = (K(\mathbf{u}^k))^{-1} (-\mathbf{r}(\mathbf{u}^k)) \quad (\text{A.75})$$

$$\delta\mathbf{u}_T^{k+1} = (K(\mathbf{u}^k))^{-1} \mathbf{f}_{\text{ext}}, \quad (\text{A.76})$$

then Eq. (A.74) can be rewritten into

$$\delta\mathbf{u}^{k+1} = \delta\tilde{\mathbf{u}}^{k+1} + \delta\alpha^{k+1} \delta\mathbf{u}_T^{k+1}. \quad (\text{A.77})$$

The arc-length condition must then be imposed. Points $(\mathbf{u}^0, \mathbf{f}^0)$, $(\mathbf{u}^{k+1}, \mathbf{f}^{k+1})$ are forced to maintain the same distance $\Delta\ell$ all through the iterations. Consequently, the arc-length restriction will be expressed as

$$(\Delta\mathbf{u}^{k+1})^T \Delta\mathbf{u}^{k+1} + c (\Delta\alpha^{k+1})^2 (\mathbf{f}_{\text{ext}})^T \mathbf{f}_{\text{ext}} = (\Delta\ell)^2. \quad (\text{A.78})$$

Recalling that $\Delta\mathbf{u}^{k+1} = \Delta\mathbf{u}^k + \delta\mathbf{u}^{k+1}$, where the arc-length condition is verified for $\Delta\mathbf{u}^k$, and using Eq. (A.77), Eq. (A.78) can be manipulated to yield the following quadratic equation on $\delta\alpha^{k+1}$, (Crisfield, 1991),

$$a_1 (\delta\alpha^{k+1})^2 + a_2 \delta\alpha^{k+1} + a_3 = 0, \quad (\text{A.79})$$

with a_1, a_2, a_3 being the three scalar values given by

$$a_1 = (\delta\mathbf{u}_T^{k+1})^T \delta\mathbf{u}_T^{k+1} + c (\mathbf{f}_{\text{ext}})^T \mathbf{f}_{\text{ext}} \quad (\text{A.80})$$

$$a_2 = 2 (\Delta \mathbf{u}^k + \delta \tilde{\mathbf{u}}^{k+1})^T \delta \mathbf{u}_T^{k+1} + \Delta \alpha^k c (\mathbf{f}_{\text{ext}})^T \mathbf{f}_{\text{ext}} \quad (\text{A.81})$$

$$a_3 = 2 (\Delta \mathbf{u}^k)^T \delta \tilde{\mathbf{u}}^{k+1} + (\delta \tilde{\mathbf{u}}^{k+1})^T \delta \tilde{\mathbf{u}}^{k+1} \quad (\text{A.82})$$

The sequence of computations can be arranged as follows:

1. Compute $\delta \tilde{\mathbf{u}}^{k+1}$ by solving

$$\mathbf{K}(\mathbf{u}^k) \delta \tilde{\mathbf{u}}^{k+1} = -\mathbf{r}(\mathbf{u}^k) \quad (\text{A.83})$$

(equivalent to Eq. (A.75)).

2. Compute $\delta \mathbf{u}_T^{k+1}$ by solving

$$\mathbf{K}(\mathbf{u}^k) \delta \mathbf{u}_T^{k+1} = \mathbf{f}_{\text{ext}} \quad (\text{A.84})$$

(equivalent to Eq. (A.76)).

3. Compute the values of a_1 , a_2 , a_3 from Eqs. (A.80), (A.81) and (A.82).
4. Solve Eq. (A.79). Here, two cases are possible:

If no real values are obtained, that is, if Eq. (A.79) has two conjugate complex roots, the process will stop*.

If two real values are obtained, one of them* will be used to compute $\delta \mathbf{u}^{k+1}$ from Eq. (A.77).

Finally, we update the load-level parameter, and also displacements and forces, as

$$\alpha^{k+1} = \alpha^k + \delta \alpha^{k+1}$$

$$\mathbf{u}^{k+1} = \mathbf{u}^k + \delta \mathbf{u}^{k+1}$$

$$\mathbf{f}^{k+1} = \mathbf{f}^k + \delta \alpha^{k+1} \mathbf{f}_{\text{ext}} = \alpha^{k+1} \mathbf{f}_{\text{ext}}.$$

* Further considerations on this situation will be made later on.

Additional considerations on the arc-length procedure

1. Choosing the sign of the prediction ${}^{n+1}\Delta\alpha^1$

As indicated in Fafard & Massicotte (1993), the natural choice for the sign of ${}^{n+1}\Delta\alpha^1$ is

$$\text{sign}({}^{n+1}\Delta\alpha^1) = \text{sign}\left({}^{(n)}\Delta\mathbf{u}^T {}^{n+1}\Delta\mathbf{u}_T\right).$$

Writing this scalar product as

$$({}^{n}\Delta\mathbf{u})^T {}^{n+1}\Delta\mathbf{u}_T = \|{}^n\Delta\mathbf{u}\| \|{}^{n+1}\Delta\mathbf{u}_T\| \cos({}^n\Delta\mathbf{u}, {}^{n+1}\Delta\mathbf{u}_T)$$

we have that

- An acute angle between the previous increment of displacement ${}^n\Delta\mathbf{u}$ and ${}^{n+1}\Delta\mathbf{u}_T$ forces an increase of the load level
- whereas an obtuse angle $({}^n\Delta\mathbf{u}, {}^{n+1}\Delta\mathbf{u}_T)$ yields a negative value of ${}^{n+1}\Delta\alpha^1$ and thus a decrease of the load level.

2. Choosing the appropriate root ${}^{n+1}\delta\alpha^{k+1}$

Let us assume that the quadratic equation

$$a_1 ({}^{n+1}\delta\alpha^{k+1})^2 + a_2 {}^{n+1}\delta\alpha^{k+1} + a_3 = 0$$

has two real roots ${}^{n+1}\delta\alpha_1^{k+1}$ and ${}^{n+1}\delta\alpha_2^{k+1}$. We denote the associated correction vectors respectively as ${}^{n+1}\delta\mathbf{u}_1^{k+1}$ and ${}^{n+1}\delta\mathbf{u}_2^{k+1}$.

To choose between these two roots, and since we want to continue to move in the same direction as in the previous iteration, we demand for the angle defined by vectors ${}^{n+1}\Delta\mathbf{u}^k$ and ${}^{n+1}\Delta\mathbf{u}^{k+1}$ to be acute. This means that we need to build the two vectors

$${}^{n+1}\Delta\mathbf{u}_i^{k+1} = {}^{n+1}\Delta\mathbf{u}^k + {}^{n+1}\delta\mathbf{u}_i^{k+1}, \quad i = 1, 2.$$

We take the correction ${}^{n+1}\delta\alpha_i^{k+1}$ that gives a positive value of the scalar product

$$({}^{n+1}\Delta\mathbf{u}^k)^T {}^{n+1}\Delta\mathbf{u}_i^{k+1}.$$

If both scalar products have the same sign, we will choose, (Cervera, 1986), the load-level correction ${}^{n+1}\delta\alpha_i^{k+1}$ that is most similar to the solution $\delta\alpha_{lin}^{k+1} = -a_3/a_2$ of the linearized version of the quadratic equation, $a_2 {}^{n+1}\delta\alpha^{k+1} + a_3 = 0$.

3. What to do when complex roots are obtained

When complex values of the roots are obtained, the algorithm fails. Some authors, however, detect this situation and force a smaller value of the arc length $\Delta\ell$; in this way, an additional computational effort is required as we need to go back to the previous iteration, but in return the algorithm does not stop.

Nevertheless, this situation (presence of complex roots) is less likely to happen if an automatic update of the arc length is performed.

4. Automatic update of the arc length

Since an appropriate value of the arc length $\Delta\ell$ is unknown a priori, it is not advisable in general to choose a value at the beginning of the analysis and keep it fixed. Besides, the appropriate value may differ from one step to the next.

Crisfield (1991) suggests a recomputation of $\Delta\ell$ at the beginning of each increment following the expression

$$\Delta\ell_{new} = \sqrt{\frac{N_{opt}}{N_{old}}} \Delta\ell_{old}, \quad (A.85)$$

where $\Delta\ell_{old}$ is the arc length that was used in the previous increment, N_{old} is the total number of iterations that were needed before convergence and N_{opt} is the desired number of iterations. In this way, if $N_{old} < N_{opt}$, the arc length will be increased. If, on the contrary, the previous increment required a number of iterations $N_{old} > N_{opt}$, the arc length will be made smaller.

Alternative updates have been suggested. In Bellini & Chulya (1987), the expression

$$\Delta l_{new} = \sqrt[4]{\frac{N_{old}}{N_{opt}}} \Delta l_{old} \quad (\text{A.86})$$

is employed and compared to Crisfield's update, Eq. (A.85).

A general formulation of the arc-length method including load and displacement control

Let us recall the arc-length condition in its original form

$$(\Delta \mathbf{u})^T \Delta \mathbf{u} + c (\Delta \alpha)^2 (\mathbf{f}_{ext})^T \mathbf{f}_{ext} = (\Delta l)^2. \quad (\text{A.87})$$

As indicated in Fafard & Massicotte (1993), Eq. (A.87) can be generalised by introducing an n -dimensional diagonal matrix \mathbf{P} as follows

$$(\Delta \mathbf{u})^T \mathbf{P} \Delta \mathbf{u} + c (\Delta \alpha)^2 (\mathbf{f}_{ext})^T \mathbf{f}_{ext} = (\Delta l)^2. \quad (\text{A.88})$$

In this way, most solution strategies can be obtained by defining an adequate \mathbf{P} matrix in (A.88).

- The **original arc-length method** is recovered by using $\mathbf{P} = \mathbf{I}$; indeed, this choice of matrix \mathbf{P} turns us back to Eq. (A.87).
- To perform a **load control**, we only need to set \mathbf{P} to a zero matrix and assign $c = 1$.
- A **displacement control** on the m -th degree of freedom \mathbf{u}_m will be associated to the following definition of \mathbf{P} :

$$P_{ii} = \delta_{im}, \quad i = 1, \dots, n$$

In this case, we also need to set c to zero.

- Finally, a **general arc-length method** can be produced that takes into account only some specified components of \mathbf{u} ; this is accomplished by setting to zero the rest of the associated diagonal terms of matrix \mathbf{P} .

Appendix B

The two stress update algorithms in Cartesian coordinates

In Chapter 4, two stress update algorithms for large strains are discussed in a convected framework. The use of convected frames allows for a unified presentation of both algorithms and a-priori error analysis. In this Appendix, these two algorithms are presented in a more classical way, by using Cartesian coordinates.

B.1 Preliminaries

B.1.1 Basic equations

The first ingredient of continuum mechanics is the equation of motion, $\mathbf{x} = \mathbf{x}(\mathbf{X}, t)$, which yields the position \mathbf{x} of material particles, denoted by their material coordinates \mathbf{X} , at time t , (Malvern, 1969). If the initial spatial coordinates are employed as material coordinates, the material displacements can be defined as $\mathbf{u}(t) = \mathbf{x}(t) - \mathbf{X}$. Once the displacements are defined, the kinematical description continues with strain representation. The starting point is the deformation gradient \mathbf{F}

$$\mathbf{F} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}}. \quad (\text{B.1})$$

Various strain tensors may be defined by means of \mathbf{F} . The Lagrange strain tensor, for instance, is

$$\mathbf{E} = \frac{1}{2} \left(\mathbf{F}^T \mathbf{F} - \mathbf{I} \right), \quad (\text{B.2})$$

where T means transpose and I is the identity. Another tensor representing strain is the spatial gradient of velocity l . This tensor yields relevant tensors if decomposed into symmetric part (rate-of-deformation tensor, d) and skew-symmetric part (spin tensor, ω),

$$l = \frac{\partial v}{\partial x} = d + \omega. \quad (B.3)$$

A very common simplification in solid mechanics is that of small deformations. If displacements, rotations and strains are small enough, two important points follow: *i*) the relation between displacements and strain is linear and *ii*) the initial configuration of the body, Ω_0 , can be used to solve the governing equations. Because of this, a **geometrically linear** problem results.

In some other problems, on the contrary, displacements are large when compared to the initial dimensions of the body. The relation between displacements and strains is no longer linear and, moreover, the governing equations must be solved over the current configuration Ω_t at time t , not over Ω_0 . Since the motion that transforms Ω_0 into Ω_t is precisely the fundamental unknown, a **geometrically nonlinear** problem is obtained.

The balance laws of continuum mechanics state the conservation of mass, momentum and energy, (Malvern, 1969). For a wide range of problems in solid mechanics, three simplifying assumptions are common: *i*) mechanical and thermal effects are uncoupled, *ii*) the density is constant and *iii*) inertia forces are negligible in comparison to the other forces acting on the body (quasistatic process). The mechanical problem is then governed by the momentum balance alone, which becomes a static equilibrium equation,

$$\frac{\partial \sigma_{ij}}{\partial x_j} + b_i = 0, \quad (B.4)$$

where σ is the Cauchy stress tensor and b is the force per unit volume. Equation (B.4) models many problems of practical interest –including, for instance, various forming processes, see NUMIFORM (1995).

B.1.2 Stress tensors

The most common representation of stress is the Cauchy stress tensor $\boldsymbol{\sigma}$, defined in the current configuration Ω_t and already presented in Eq. (B.4). This tensor has a clear physical meaning, because it involves only forces and surfaces in the current configuration. Experimental stress measures taken in a laboratory correspond to Cauchy stresses, also known as true stresses.

In a large strain context, other representations of stress are possible and indeed useful. The key idea, (Pinsky *et al.*, 1983), is that Ω_0 and Ω_t are different configurations, so tensors defined in each configuration cannot be combined by operations such as subtraction and addition. Let ${}^0\boldsymbol{\sigma}$ and ${}^t\boldsymbol{\sigma}$ be the Cauchy stress tensors at the initial time t_0 and current time t respectively; the increment of stress may not be defined as ${}^t\boldsymbol{\sigma} - {}^0\boldsymbol{\sigma}$, because the two tensors are referred to different configurations. As stress increments will be needed to update stresses, a proper definition is required.

An alternative representation of stress is the second Piola-Kirchhoff tensor \boldsymbol{S} , defined as the pull-back of $\boldsymbol{\sigma}$

$$\boldsymbol{S} = J\boldsymbol{F}^{-1}\boldsymbol{\sigma}\boldsymbol{F}^{-T}, \quad (\text{B.5})$$

where $J = \det(\boldsymbol{F})$ is the Jacobian of the motion, which reflects the variation of unit volume associated to the deformation, and the inverse of the deformation gradient \boldsymbol{F}^{-1} is employed to transform $\boldsymbol{\sigma}$ from Ω_t to Ω_0 , see Figure B.1a. Equation (B.5) is called the pull-back Piola transformation. It must be remarked that \boldsymbol{S} represents the state of stress at time t but referred to configuration Ω_0 , and should not be confused with ${}^0\boldsymbol{\sigma}$, the stress at initial time t_0 .

Equation (B.5) may be reversed, and then $\boldsymbol{\sigma}$ may be seen as the push-forward of \boldsymbol{S}

$$\boldsymbol{\sigma} = \frac{1}{J}\boldsymbol{F}\boldsymbol{S}\boldsymbol{F}^T, \quad (\text{B.6})$$

where \boldsymbol{F} transforms \boldsymbol{S} from Ω_0 to Ω_t . Equation (B.6) is the so-called push-forward Piola transformation, Figure B.1b. With the help of Eqs. (B.5) and (B.6), the stress increment may be represented either as

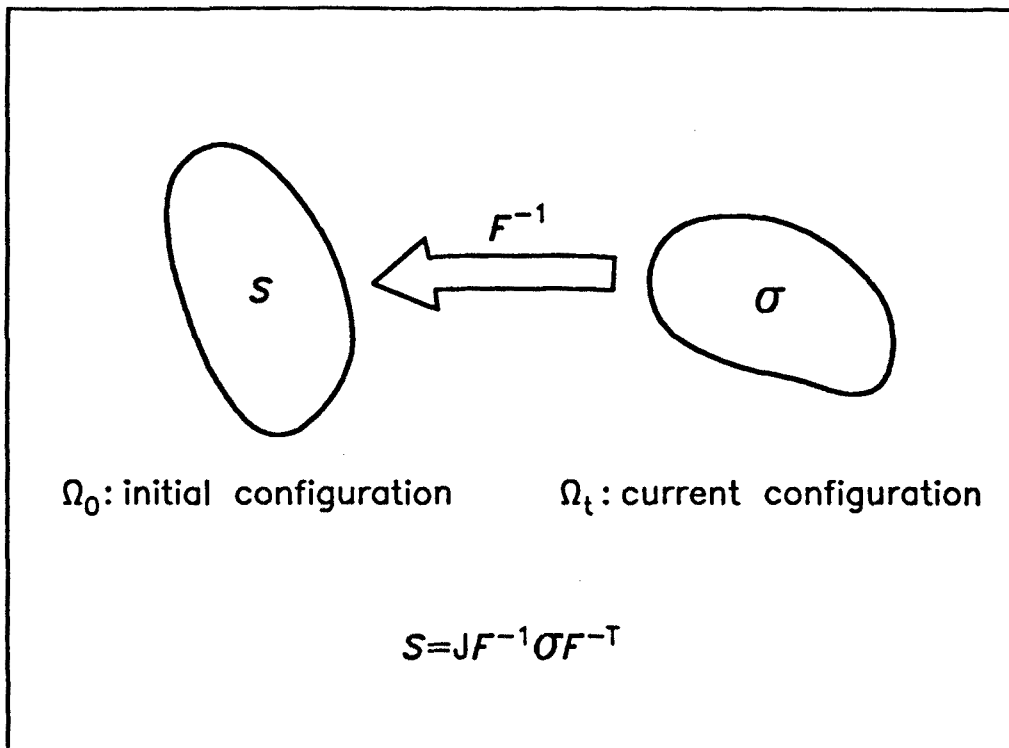


Figure B.1a Pull-back Piola transformation

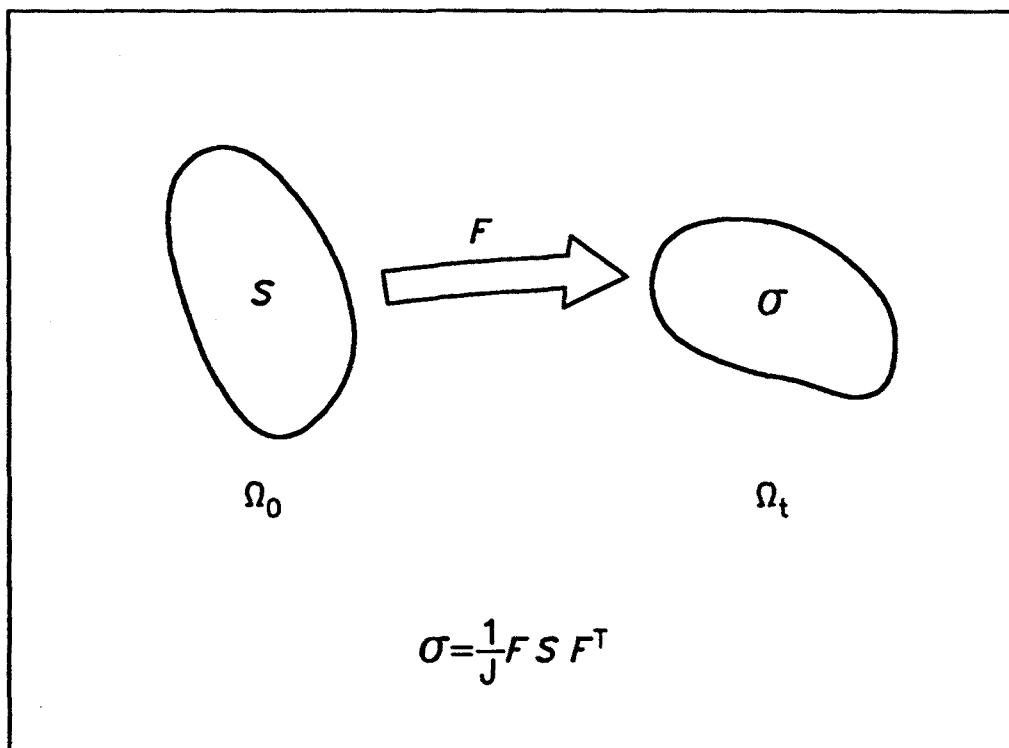


Figure B.1b Push-forward Piola transformation

$${}^t\Delta\sigma = {}^t\sigma - J^{-1}F {}^0\sigma F^T \quad (B.7a)$$

or

$${}^0\Delta\sigma = JF^{-1} {}^t\sigma F^{-T} - {}^0\sigma \quad (B.7b)$$

referred to Ω_t or Ω_0 respectively. The Piola transformations are employed to refer the two tensors to a common configuration, where the subtraction can be properly performed.

B.1.3 Constitutive equations and objectivity

In nonlinear solid mechanics, the material behaviour is often described by a rate-form constitutive equation, relating the stress rate to velocity and/or its derivatives and the stress state (and eventually, some internal variables). The particular case of **hypoelastic materials**, where the stress rate depends linearly on the rate-of-deformation tensor \mathbf{d} , (Malvern, 1969), will be considered here to present the two stress update algorithms. The two algorithms, however, can be extended to elastoplastic problems, by profiting from the decomposition of the rate-of-deformation \mathbf{d} into elastic and plastic parts, see Khan & Huang (1995). The hypoelastic constitutive law is

$$\dot{\sigma} = \mathbf{C} : \mathbf{d}, \quad (B.8)$$

where $\dot{\sigma}$ is the material rate of stress, and \mathbf{C} is the fourth-order modulus tensor. For isotropic materials, \mathbf{C} can be written in terms of just two parameters, the Lamé constants λ and μ , just like in classical elasticity, (Malvern, 1969).

In fact, Eq. (B.8) is only valid for small strains. As shown next, the material rate of stress $\dot{\sigma}$ may not be employed to represent stress variation in a large strain problem, because it is not an **objective** tensor.

The **principle of objectivity** is a fundamental requirement regarding the constitutive equation in large strain solid mechanics, (Hughes, 1984; Marsden & Hughes, 1983):

if the constitutive equations really describe the physical behaviour of the continuum, they must be independent of the observer. In other words, they must remain invariant under any change of reference frame.

This requirement is fulfilled if objective quantities appear in the constitutive equations. A quantity is said to be objective if it transforms in a proper tensorial manner under a superposed rigid-body motion. Let the rigid motion be represented by an orthogonal rotation tensor Q , ($Q^{-1} = Q^T$) and a translation \mathbf{a} . The time-dependent relation between old and new coordinates is then $\mathbf{x}^{\text{new}}(t) = Q(t)\mathbf{x} + \mathbf{a}(t)$. It is postulated that the Cauchy stress tensor σ is objective. As it is a second-order tensor, it transforms according to

$$\sigma^{\text{new}}(t) = Q(t)\sigma(t)Q(t)^T. \quad (B.9)$$

If Eq. (B.9) is derivated with respect to time, it is readily observed that the material derivative of an objective tensor is not objective:

$$\dot{\sigma}^{\text{new}} = \dot{Q}\sigma Q^T + Q\dot{\sigma}Q^T + Q\sigma\dot{Q}^T \neq Q\dot{\sigma}Q^T. \quad (B.10)$$

This invalidates the use of $\dot{\sigma}$ as the stress rate in a rate-form constitutive equation. An alternative, objective stress rate σ^* is therefore needed. In fact, Eq. (B.8), which is valid for small strain analysis, provides unrealistic stress distributions in very simple large strain tests, such as a rigid rotation test (see, for instance, Rodríguez-Ferran & Huerta (1996)).

As for the rate-of-deformation tensor d , it can be shown that it is an objective tensor, so it may be employed to represent strains in a constitutive equation. Indeed, the hypoelastic constitutive equation is rewritten, in a large strain framework, as

$$\sigma^* = C : d. \quad (B.11)$$

The stress rate σ^* is not uniquely determined by the objectivity principle. Some classical options reviewed by Pinsky *et al.* (1983) are the Jaumann rate

$$\sigma_J^* = \dot{\sigma} + \sigma\omega - \omega\sigma, \quad (B.12a)$$

the Green-Naghdi rate

$$\sigma_{GN}^* = \dot{\sigma} + \sigma\Omega - \Omega\sigma, \quad (B.12b)$$

and the Truesdell rate

$$\sigma_T^* = \dot{\sigma} - l\sigma - \sigma l^T + \text{tr}(\mathbf{d})\sigma, \quad (B.12c)$$

where Ω is the so-called rate-of-rotation tensor, see Malvern (1969), and $\text{tr}(\mathbf{d})$ denotes the trace of tensor \mathbf{d} .

It can be easily checked that the terms in the RHS of Eqs. (B.12) additional to the material rate $\dot{\sigma}$ ensure that the defined rates are indeed objective. Either by means of the spin rate ω , the rate-of-rotation Ω or the gradient of velocity \mathbf{l} , the non-objectivity of $\dot{\sigma}$ is compensated, and an objective σ^* is obtained.

Regarding the Truesdell rate, it has been defined in Eq. (B.12c) in terms of the Eulerian tensors σ , \mathbf{l} and \mathbf{d} , referred to the current configuration. An alternative expression, which provides insight into its physical meaning and is useful from an algorithmic viewpoint, see Pinsky *et al.* (1983), is

$$\sigma_T^* = \frac{1}{J} F \dot{S} F^T. \quad (B.13)$$

In this equation, the Truesdell rate can be interpreted as the push-forward Piola transformation of the material derivative of the second Piola-Kirchhoff stress tensor \mathbf{S} . Thus, instead of using the time derivative of the Cauchy stress tensor which yields the non-objective material rate, see Eq. (B.10), the Truesdell rate is preferred because it is by construction an objective rate. In Eq. (B.13) it is easily observed that the Truesdell rate proceeds in three steps: *i*) σ is pulled-back into \mathbf{S} , *ii*) the material derivative of \mathbf{S} is performed and *iii*) the resulting rate is pushed-forward into the current configuration. Where the rationale is that the material derivative of a material tensor (i.e., a tensor referred to the initial configuration) yields an objective tensor.

B.2 Two stress update algorithms for large strain solid mechanics

B.2.1 Introductory remarks

If the Finite Element Method (Bathe, 1982; Zienkiewicz & Taylor, 1991) is employed, the partial differential equation (B.4) is transformed into the nonlinear system of equations

$$\mathbf{r}(\mathbf{u}) = \mathbf{f}_{\text{int}}(\mathbf{u}) - \mathbf{f}_{\text{ext}}(\mathbf{u}) = \mathbf{0}, \quad (\text{B.14})$$

where \mathbf{f}_{int} is the internal force vector, \mathbf{f}_{ext} is the external load vector and \mathbf{r} are the residual forces, which are null if equilibrium is attained. Equation (B.14) is typically solved incrementally with a displacement-based implicit method, (Bathe, 1982). The fundamental unknowns are then the incremental displacements $\Delta \mathbf{u} = {}^{n+1}\mathbf{x} - {}^n\mathbf{x}$ from one (known) equilibrium configuration Ω_n at time t_n to a new (unknown) equilibrium configuration Ω_{n+1} at time $t_{n+1} = t_n + \Delta t$.

Nonlinear systems of equations like Eq. (B.14) may be solved by a number of iterative techniques, (Crisfield, 1991). The two key ideas are that *i*) a linearized form of Eq. (B.14) is used to predict and then iteratively correct $\Delta \mathbf{u}$, and *ii*) the constitutive equation (B.11) must be integrated after each iteration to check equilibrium. Indeed, each iteration i yields a candidate configuration Ω_{n+1}^i . To check whether it is the equilibrium configuration at time t_{n+1} , stresses must be updated from the previous configuration Ω_n by time-integrating the rate-form constitutive equation. By doing so, the internal forces and the residual forces, Eq. (B.14), may be computed.

B.2.2 Incremental objectivity

As remarked by Hughes (1984), stress update is the central problem in nonlinear solid mechanics and affects fundamentally the accuracy of the overall algorithm.

In the context of the incremental build-up of the solution, it is useful to define the incremental versions of the tensors presented in Eqs. (B.1) and (B.2). Let ${}^n\mathbf{F}$ and ${}^{n+1}\mathbf{F}$ be the deformation gradients relating Ω_n and Ω_{n+1} respectively to the reference

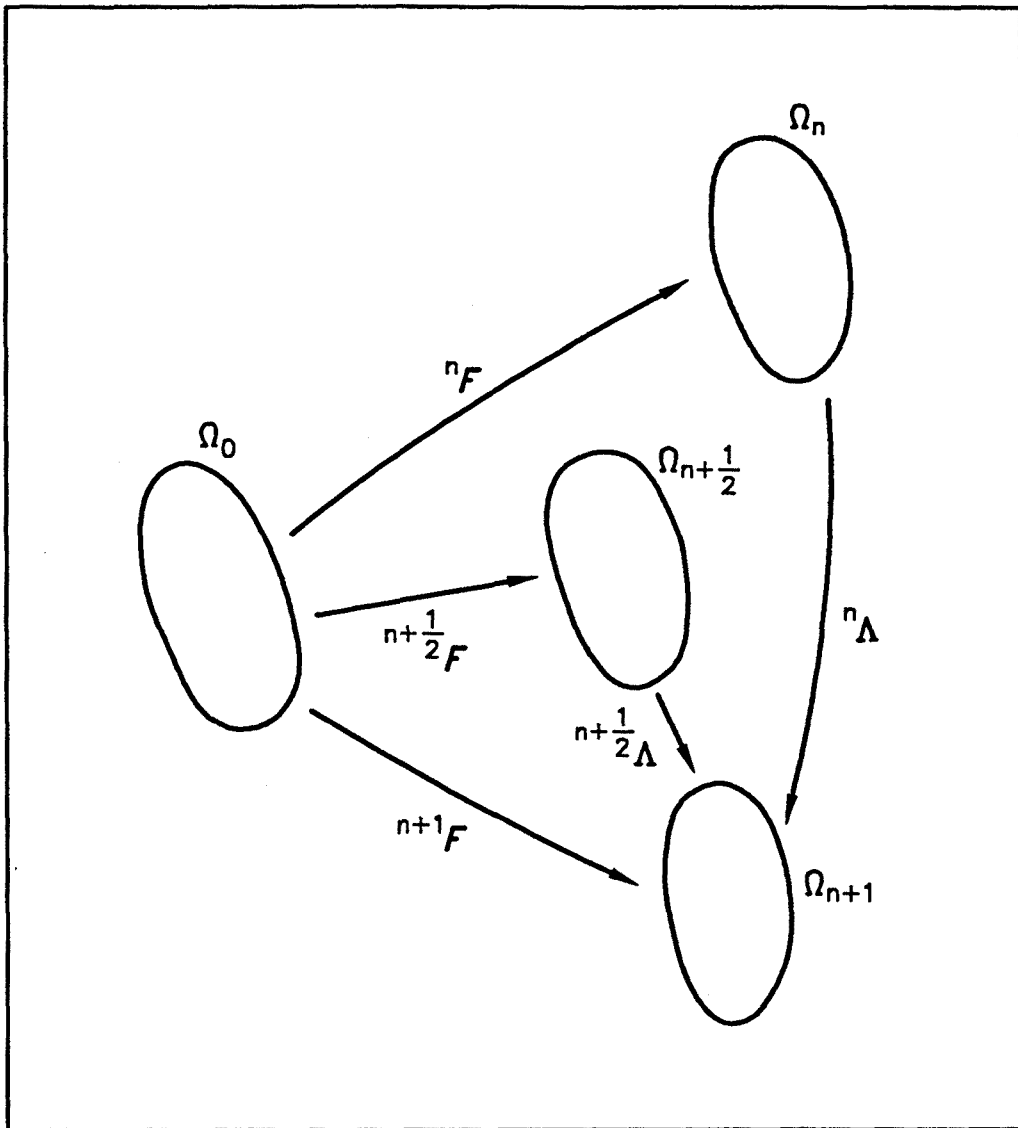


Figure B.2 Deformation gradients in incremental analysis

configuration Ω_0 , see Figure B.2. The incremental deformation gradient ${}^n A$ is

$${}^n A = {}^{n+1} F {}^n F^{-1}, \quad (B.15)$$

which refers configuration Ω_{n+1} to configuration Ω_n . The corresponding incremental Lagrange strain tensor is then

$${}^n \Delta E = \frac{1}{2} ({}^n A^T {}^n A - I). \quad (B.16)$$

The objectivity of the constitutive equation (B.11) is attained through the definition of objective stress rates, Eqs. (B.12). **Incremental objectivity** is a requirement on the algorithm for the numerical time-integration of the constitutive equation, which is often presented as the discrete counterpart of the principle of objectivity (Hughes & Winget, 1980). Let the incremental deformation gradient ${}^n\mathbf{A}$ relating configurations Ω_n and Ω_{n+1} be an orthogonal tensor ${}^n\mathbf{R}$. The numerical algorithm is said to be **incrementally objective** if it predicts a stress state at t_{n+1} that is simply a rotation of the stress state at t_n ,

$${}^{n+1}\boldsymbol{\sigma} = {}^n\mathbf{R} {}^n\boldsymbol{\sigma} {}^n\mathbf{R}^T. \quad (\text{B.17})$$

In other words, an incrementally objective algorithm assumes that the body motion between t_n and t_{n+1} is a rigid rotation and rotates stresses in accordance with that assumption, with no spurious stress variations, Eq. (B.17). It must be remarked, however, that the incremental deformation gradient ${}^n\mathbf{A}$ being an orthogonal tensor ${}^n\mathbf{R}$ does not necessarily imply that the true (unknown) body motion between t_n and t_{n+1} is a rigid rotation. For this reason, incremental objectivity is just a reasonable property of the numerical algorithm (i.e., a rigid rotation is assumed when possible) rather than a physical requirement like the principle of objectivity.

B.2.3 Two stress update algorithms for large strains

First stress update algorithm

It is possible to employ the incremental Lagrange strain tensor defined in Eq. (B.16) as the strain measure in the increment Δt . The stress increment is then

$${}^n\Delta\boldsymbol{\sigma} = \mathbf{C} : {}^n\Delta\mathbf{E}, \quad (\text{B.18})$$

where the superscript n in $\Delta\boldsymbol{\sigma}$ indicates that this tensorial quantity is, like ${}^n\Delta\mathbf{E}$, referred to configuration Ω_n .

In a large-strain context it is no longer valid to compute the new stresses ${}^{n+1}\boldsymbol{\sigma}$ by simply adding the stress increment ${}^n\Delta\boldsymbol{\sigma}$ to the old stresses ${}^n\boldsymbol{\sigma}$, because these latter two

tensors are in the configuration Ω_n and ${}^{n+1}\boldsymbol{\sigma}$ is sought in the configuration Ω_{n+1} . It is necessary to transform the tensors adequately by means of the push-forward Piola transformation, Eq. (B.6). The numerical algorithm for stress update is then

$${}^{n+1}\boldsymbol{\sigma} = {}^n J^{-1} {}^n \boldsymbol{\Lambda} {}^n \boldsymbol{\sigma} {}^n \boldsymbol{\Lambda}^T + {}^n J^{-1} {}^n \boldsymbol{\Lambda} ({}^n \Delta \boldsymbol{\sigma}) {}^n \boldsymbol{\Lambda}^T, \quad (\text{B.19})$$

where the Jacobian ${}^n J$ is defined as $\det({}^n \boldsymbol{\Lambda})$ and the incremental deformation gradient, Eq. (B.15), is employed to push-forward both ${}^n \boldsymbol{\sigma}$ and ${}^n \Delta \boldsymbol{\sigma}$ into the new configuration Ω_{n+1} .

This algorithm is incrementally objective: if ${}^n \boldsymbol{\Lambda}$ is an orthogonal tensor, Eq. (B.16) yields a null strain tensor ${}^n \Delta \boldsymbol{E}$ and Eq. (B.19) reduces to Eq. (B.17), thus predicting a rigid rotation of stresses, with no spurious stress variations. Note that the use of the full incremental Lagrange tensor, including quadratic terms, is essential for the incremental objectivity of the algorithm.

Second stress update algorithm

An alternative, more accurate numerical algorithm will be shown next. Following Pinsky *et al.*, 1983), the hypoelastic constitutive equation is written in terms of the Truesdell objective stress rate, Eq. (B.12c),

$$\boldsymbol{\sigma}_T^* = \boldsymbol{C} : \boldsymbol{d}. \quad (\text{B.20})$$

A basic ingredient of this algorithm is that \boldsymbol{d} is evaluated in the midstep configuration $\Omega_{n+\frac{1}{2}}$, defined through linear interpolation between Ω_n and Ω_{n+1} . The midstep spatial coordinates are

$${}^{n+\frac{1}{2}}\boldsymbol{x} = \frac{1}{2} ({}^n \boldsymbol{x} + {}^{n+1} \boldsymbol{x}) = {}^n \boldsymbol{x} + \frac{1}{2} \Delta \boldsymbol{u}, \quad (\text{B.21})$$

the associated deformation gradient is

$${}^{n+\frac{1}{2}}\boldsymbol{F} = \frac{1}{2} ({}^n \boldsymbol{F} + {}^{n+1} \boldsymbol{F}), \quad (\text{B.22})$$

and, similarly to Eq. (B.15), the incremental deformation gradient relating the midstep and final configurations is

$${}^{n+\frac{1}{2}}\mathbf{A} = {}^{n+1}\mathbf{F} {}^{n+\frac{1}{2}}\mathbf{F}^{-1}. \quad (\text{B.23})$$

The different deformation gradient tensors are summarized in Figure B.2. Using a midpoint rule algorithm to integrate Eq. (B.20), the stress update becomes

$${}^{n+1}\boldsymbol{\sigma} = {}^n J^{-1} {}^n \mathbf{A} {}^n \boldsymbol{\sigma} {}^n \mathbf{A}^T + {}^{n+\frac{1}{2}} J^{-1} {}^{n+\frac{1}{2}} \mathbf{A} (\Delta t \mathbf{C} : \mathbf{d})|_{n+\frac{1}{2}} {}^{n+\frac{1}{2}} \mathbf{A}^T. \quad (\text{B.24})$$

with the Jacobian ${}^{n+\frac{1}{2}}J$ defined as $\det({}^{n+\frac{1}{2}}\mathbf{A})$. As in Eq. (B.19), tensors referred to the initial and midstep configurations are pushed-forward into the final one by means of the appropriate incremental deformation gradients ${}^n \mathbf{A}$ and ${}^{n+\frac{1}{2}} \mathbf{A}$.

Recalling the definition of \mathbf{d} , Eq. (B.3), the approximation to ${}^{n+\frac{1}{2}}\mathbf{d}$ needed in Eq. (B.24) will be

$${}^{n+\frac{1}{2}}\mathbf{d} = \frac{{}^{n+\frac{1}{2}}\Delta\boldsymbol{\varepsilon}}{\Delta t} = \frac{1}{2\Delta t} \left\{ \left[\frac{\partial(\Delta\mathbf{u})}{\partial({}^{n+\frac{1}{2}}\mathbf{x})} \right] + \left[\frac{\partial(\Delta\mathbf{u})}{\partial({}^{n+\frac{1}{2}}\mathbf{x})} \right]^T \right\}. \quad (\text{B.25})$$

The stress increment is then

$${}^{n+\frac{1}{2}}\Delta\boldsymbol{\sigma} = (\Delta t \mathbf{C} : \mathbf{d})|_{n+\frac{1}{2}} = \mathbf{C} : \frac{1}{2} \left\{ \left[\frac{\partial(\Delta\mathbf{u})}{\partial({}^{n+\frac{1}{2}}\mathbf{x})} \right] + \left[\frac{\partial(\Delta\mathbf{u})}{\partial({}^{n+\frac{1}{2}}\mathbf{x})} \right]^T \right\}. \quad (\text{B.26})$$

It must be noted that in Eq. (B.25) the strain increment ${}^{n+\frac{1}{2}}\Delta\boldsymbol{\varepsilon}$ is represented by the symmetrized gradient of the incremental displacements, like in a small strain analysis. No additional quadratic terms are needed in Eq. (B.26), because large strains are properly modelled by employing the midstep configuration to compute the gradient of displacements. As discussed in Pinsky *et al.* (1983), this algorithm is also incrementally objective. Both the accuracy analysis and the numerical experiments of Chapter 4 show that the second algorithm (which is second-order accurate) is superior to the first (which only has first-order accuracy).

The two stress update algorithms can also be employed in elastoplasticity. The basic idea is to model the elastic part of the deformation with an hypoelastic law, and use any of the two algorithms to compute the elastic trial stress, (Hughes, 1984; Pinsky *et al.*, 1983). After that, a plastic corrector –a radial return algorithm, for instance–, is required to account for material nonlinearity, see Hughes (1984).

B.2.4 Implementation aspects

It is shown in this Subsection that any of the two stress update algorithms can be employed to add large-strain capabilities to a small-strain FE code in a simple way.

The basic idea is that the incremental deformation gradients required in Eqs. (B.19) and (B.24) can be computed in a straightforward manner by using quantities that are available in a small strain code. Consider, for instance, the incremental deformation gradient ${}^n\mathbf{A}$ relating Ω_n to Ω_{n+1} , Eq. (B.15). Recalling the definition of \mathbf{F} in Eq. (B.1) and the expression of incremental displacements, it can be easily checked that ${}^n\mathbf{A}$ can be put as

$${}^n\mathbf{A} = \frac{\partial({}^{n+1}\mathbf{x})}{\partial({}^n\mathbf{x})} = \mathbf{I} + \frac{\partial(\Delta\mathbf{u})}{\partial({}^n\mathbf{x})}. \tag{B.27}$$

If an Updated Lagrangian formulation is used, (Bathe, 1982), the configuration Ω_n is taken as a reference to compute the incremental displacements. In such a context, ${}^n\mathbf{A}$ can be computed from Eq. (B.27) with the aid of standard nodal shape functions, by expressing $\Delta\mathbf{u}$ in terms of the nodal values of incremental displacements. Since the derivatives of shape functions are available in a standard FE code, (Hughes, 1987; Oñate, 1991; Zienkiewicz & Taylor, 1991), no new quantities must be computed to obtain ${}^n\mathbf{A}$.

Recalling the expression of the incremental Lagrange strain tensor ${}^n\Delta\mathbf{E}$ in terms of ${}^n\mathbf{A}$, Eq. (B.16), it can be written as

$${}^n\Delta\mathbf{E} = \frac{1}{2} \left\{ \left[\frac{\partial(\Delta\mathbf{u})}{\partial({}^n\mathbf{x})} \right] + \left[\frac{\partial(\Delta\mathbf{u})}{\partial({}^n\mathbf{x})} \right]^T + \left[\frac{\partial(\Delta\mathbf{u})}{\partial({}^n\mathbf{x})} \right]^T \left[\frac{\partial(\Delta\mathbf{u})}{\partial({}^n\mathbf{x})} \right] \right\}. \tag{B.28}$$

As for ${}^{n+\frac{1}{2}}\mathbf{A}$, combining Eqs. (B.21), (B.22) and (B.23) renders

$${}^{n+\frac{1}{2}}\mathbf{A} = \frac{\partial({}^{n+1}\mathbf{x})}{\partial({}^{n+\frac{1}{2}}\mathbf{x})} = \mathbf{I} + \frac{1}{2} \frac{\partial(\Delta\mathbf{u})}{\partial({}^{n+\frac{1}{2}}\mathbf{x})}, \quad (\text{B.29})$$

so ${}^{n+\frac{1}{2}}\mathbf{A}$ can also be directly computed with the aid of the shape functions, once the configuration of the mesh has been updated from Ω_n to $\Omega_{n+\frac{1}{2}}$.

As a result, the only two additional features that are required to handle large strains are **1.** the updating of mesh configuration and **2.** the computation of incremental deformation gradients, Eqs. (B.27) and (B.29). This can be seen by comparing the schematic algorithm for a small-strain analysis with nonlinear material behaviour, shown in Box B.1 with the large-strain versions, depicted in Box B.2 (first stress update algorithm) and Box B.3 (second stress update algorithm). In Boxes B.2 and B.3 (large strains), the modifications with respect to Box B.1 (small strains) are highlighted with boldface, and the symbol \star is employed to designate additional steps.

FOR EVERY TIME-INCREMENT $[t_n, t_{n+1}]$;

FOR EVERY ITERATION k WITHIN THE TIME-INCREMENT;

1.- Compute the incremental displacements $\Delta \mathbf{u}^k$ by solving a linearized form of Eq. (B.14)

2.- Compute the incremental strains $\Delta \boldsymbol{\epsilon}^k$ as the symmetrized gradient of displacements:

$$\Delta \boldsymbol{\epsilon}^k = \frac{1}{2} \left\{ \left[\frac{\partial(\Delta \mathbf{u}^k)}{\partial \mathbf{X}} \right] + \left[\frac{\partial(\Delta \mathbf{u}^k)}{\partial \mathbf{X}} \right]^T \right\}$$

3.- Compute the elastic trial incremental stresses $\Delta \boldsymbol{\sigma}_{\text{trial}}^k$ via the elastic modulus tensor:

$$\Delta \boldsymbol{\sigma}_{\text{trial}}^k = \mathbf{C} : \Delta \boldsymbol{\epsilon}^k$$

4.- Compute the elastic trial stresses at t_{n+1} :

$$\boldsymbol{\sigma}_{\text{trial}}^k = {}^n \boldsymbol{\sigma} + \Delta \boldsymbol{\sigma}_{\text{trial}}^k$$

5.- Compute the final stresses $\boldsymbol{\sigma}^k$ at t_{n+1} by performing the plastic correction

6.- Compute the internal forces $\mathbf{f}_{\text{int}}^k$ by integrating the stresses $\boldsymbol{\sigma}^k$

7.- Check convergence. If it is not attained, go back to step 1.

Box B.1 Small-strain analysis with nonlinear material behaviour

FOR EVERY TIME-INCREMENT $[t_n, t_{n+1}]$;

FOR EVERY ITERATION k WITHIN THE TIME-INCREMENT;

- 1.- Compute the incremental displacements $\Delta \mathbf{u}^k$ by solving a linearized form of Eq. (B.14)
- 2.- Compute the incremental strains $\Delta \boldsymbol{\varepsilon}^k$ accounting for quadratic terms, Eq. (B.28):

$$\Delta \boldsymbol{\varepsilon}^k = \frac{1}{2} \left\{ \left[\frac{\partial(\Delta \mathbf{u}^k)}{\partial({}^n \mathbf{x})} \right] + \left[\frac{\partial(\Delta \mathbf{u}^k)}{\partial({}^n \mathbf{x})} \right]^T + \left[\frac{\partial(\Delta \mathbf{u}^k)}{\partial({}^n \mathbf{x})} \right]^T \left[\frac{\partial(\Delta \mathbf{u}^k)}{\partial({}^n \mathbf{x})} \right] \right\}$$

- 3.- Compute the elastic trial incremental stresses $\Delta \boldsymbol{\sigma}_{\text{trial}}^k$ via the elastic modulus tensor:

$$\Delta \boldsymbol{\sigma}_{\text{trial}}^k = \mathbf{C} : \Delta \boldsymbol{\varepsilon}^k$$

- 4.- Compute the elastic trial stresses at t_{n+1} , pushing forward both ${}^n \boldsymbol{\sigma}$ and $\Delta \boldsymbol{\sigma}_{\text{trial}}^k$ from configuration Ω_n to Ω_{n+1} , Eq. (B.19):

$$\boldsymbol{\sigma}_{\text{trial}}^k = {}^n J^{-1} {}^n \boldsymbol{\Lambda} {}^n \boldsymbol{\sigma} {}^n \boldsymbol{\Lambda}^T + {}^n J^{-1} {}^n \boldsymbol{\Lambda} \left(\Delta \boldsymbol{\sigma}_{\text{trial}}^k \right) {}^n \boldsymbol{\Lambda}^T$$

- *.- Update the configuration from Ω_n to Ω_{n+1} by using the incremental displacements of the current step
- 5.- Compute the final stresses $\boldsymbol{\sigma}^k$ at t_{n+1} by performing the plastic correction
- 6.- Compute the internal forces $\mathbf{f}_{\text{int}}^k$ by integrating the stresses $\boldsymbol{\sigma}^k$
- 7.- Check convergence. If it is not attained, recover configuration Ω_n and go back to step 1.

Box B.2 First stress update algorithm

FOR EVERY TIME-INCREMENT $[t_n, t_{n+1}]$;

FOR EVERY ITERATION k WITHIN THE TIME-INCREMENT;

1.- Compute the incremental displacements $\Delta \mathbf{u}^k$ by solving a linearized form of Eq. (B.14)

*.- Update the configuration from Ω_n to $\Omega_{n+\frac{1}{2}}$, Eq. (B.21)

2.- Compute the incremental strains $\Delta \boldsymbol{\varepsilon}^k$ as the symmetrized gradient of displacements:

$$\Delta \boldsymbol{\varepsilon}^k = \frac{1}{2} \left\{ \left[\frac{\partial(\Delta \mathbf{u}^k)}{\partial({}^{n+\frac{1}{2}}x)} \right] + \left[\frac{\partial(\Delta \mathbf{u}^k)}{\partial({}^{n+\frac{1}{2}}x)} \right]^T \right\}$$

3.- Compute the elastic trial incremental stresses $\Delta \boldsymbol{\sigma}_{\text{trial}}^k$ via the elastic modulus tensor:

$$\Delta \boldsymbol{\sigma}_{\text{trial}}^k = \mathbf{C} : \Delta \boldsymbol{\varepsilon}^k$$

*.- Compute the incremental deformation gradients ${}^n \boldsymbol{\Lambda}$ and ${}^{n+\frac{1}{2}} \boldsymbol{\Lambda}$, Eqs. (B.27) and (B.29), and their determinants ${}^n J$ and ${}^{n+\frac{1}{2}} J$

4.- Compute the elastic trial stresses at t_{n+1} , pushing forward ${}^n \boldsymbol{\sigma}$ and $\Delta \boldsymbol{\sigma}_{\text{trial}}^k$ to Ω_{n+1} , Eq. (B.24):

$$\boldsymbol{\sigma}_{\text{trial}}^k = {}^n J^{-1} {}^n \boldsymbol{\Lambda} {}^n \boldsymbol{\sigma} {}^n \boldsymbol{\Lambda}^T + {}^{n+\frac{1}{2}} J^{-1} {}^{n+\frac{1}{2}} \boldsymbol{\Lambda} \Delta \boldsymbol{\sigma}_{\text{trial}}^k {}^{n+\frac{1}{2}} \boldsymbol{\Lambda}^T$$

*.- Update the configuration from $\Omega_{n+\frac{1}{2}}$ to Ω_{n+1}

5.- Compute the final stresses $\boldsymbol{\sigma}^k$ at t_{n+1} by performing the plastic correction

6.- Compute the internal forces $\mathbf{f}_{\text{int}}^k$ by integrating the stresses $\boldsymbol{\sigma}^k$

7.- Check convergence. If it is not attained, recover configuration Ω_n and go back to step 1.

Appendix C

A note on a numerical benchmark test: an axisymmetric shell under ring loads

The same shell test that was used in Chapter 3 (to test various nonlinear solvers in combination with Lagrange multipliers) and in Chapter 4 (to compare the two stress update algorithms for large strains), is presented here as a possible tool to validate the implementation of an arc-length algorithm.

C.1 Introduction

Nonlinear equation solvers in a general sense (i.e. resolution strategies, acceleration techniques, path following methods, etc.) is a subject of much current research (Crisfield, 1991; Kelley, 1995; Riks, 1992; Soria & Pegon, 1993; Van der Boogaard & de Borst, 1994; Zienkiewicz & Taylor, 1991). Development and analysis of nonlinear techniques requires to use numerical benchmark tests involving both material and geometrical nonlinearities, recall that there are only a few nonlinear mechanics problems with analytical solution, (Lubliner, 1990). Thus, a collection of benchmark tests in nonlinear computational mechanics has been produced over the past decade, see Crisfield (1991), Zienkiewicz & Taylor (1991), Papadrakakis & Pantazopoulos (1993) among others. These tests are of extreme importance to researchers, because they can check and validate the implementation and performance of new or even classical algorithms. Moreover, it is important to notice that it is difficult to find benchmarks where the same problem can be solved using a displacement controlled technique and an arc-length

method to overcome limit points. That is, simple problems that present snap-through and snap-back depending on the controlled variable.

Here, a well-known example: “an axisymmetric shell under ring loads”, (Zienkiewicz & Taylor, 1991), is reviewed as a possible tool to verify the implementation of arc-length algorithms. This example presents under certain circumstances snap-through or snap-back depending on the controlled variable. Therefore, the easily obtained displacement controlled solution of the snap-through case can be employed, after simple post-processing, as a reference to validate an arc-length implementation which uses as controlling variable the one that presents snap-back. Moreover, the analysis is extended beyond published results and the applicability of displacement-controlled techniques.

C.2 Problem statement

An axisymmetric shell made of an elastic material is clamped at its border, and a ring load is applied to the shell, causing a deflection v of the apex. Figure C.1 shows a description of the problem. This problem is classically solved using a displacement-controlled technique on the apex, namely, by forcing increasing values of v , see Zienkiewicz & Taylor (1991).

Figure C.2 shows the load-deflection curves presented in Zienkiewicz & Taylor (1991), each of them associated to a different eccentricity of the ring load, $e = r/R_h$. Such a figure clearly shows that this example presents a snap-through tendency as the eccentricity increases.

C.3 An alternative to displacement control

The behaviour of the structure may also be examined by plotting the load versus the displacement of the loaded node, ω . This can be done as a simple post-processing of the displacement-controlled solution, i.e. the solution is obtained controlling v but curves P versus ω are plotted. Figure C.3 shows the curves load P versus displacement of the loaded node, ω , associated again to eccentricities 0, 0.25 and 0.42. It is important

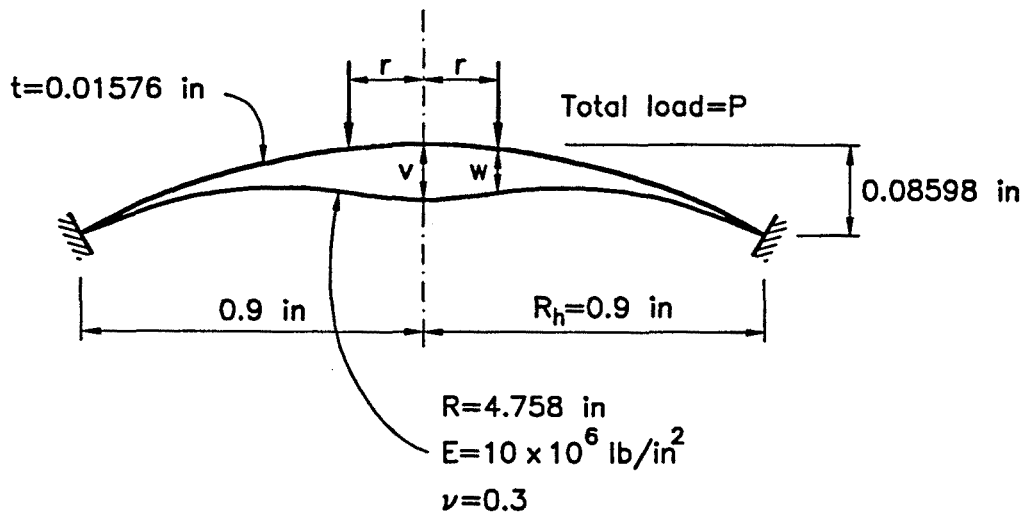


Figure C.1 Problem statement

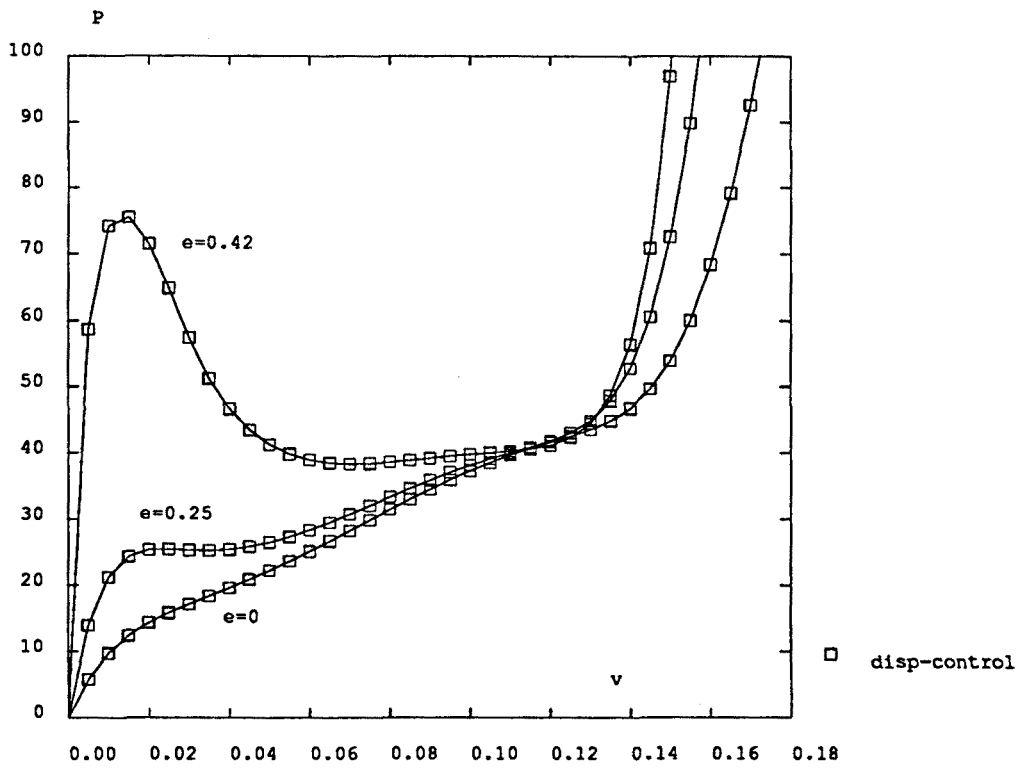


Figure C.2 Load P vs. deflection of apex v ; $e = 0$, $e = 0.25$, $e = 0.42$.

to note that the last one of these curves ($e = 0.42$) shows **snap-back**, which means that this particular problem cannot be solved using a displacement-controlled procedure on the displacement w of the loaded node. Therefore, for $e = 0.42$ the apex displacement-controlled solution may be employed as a reference to validate the implementation of arc-length techniques.

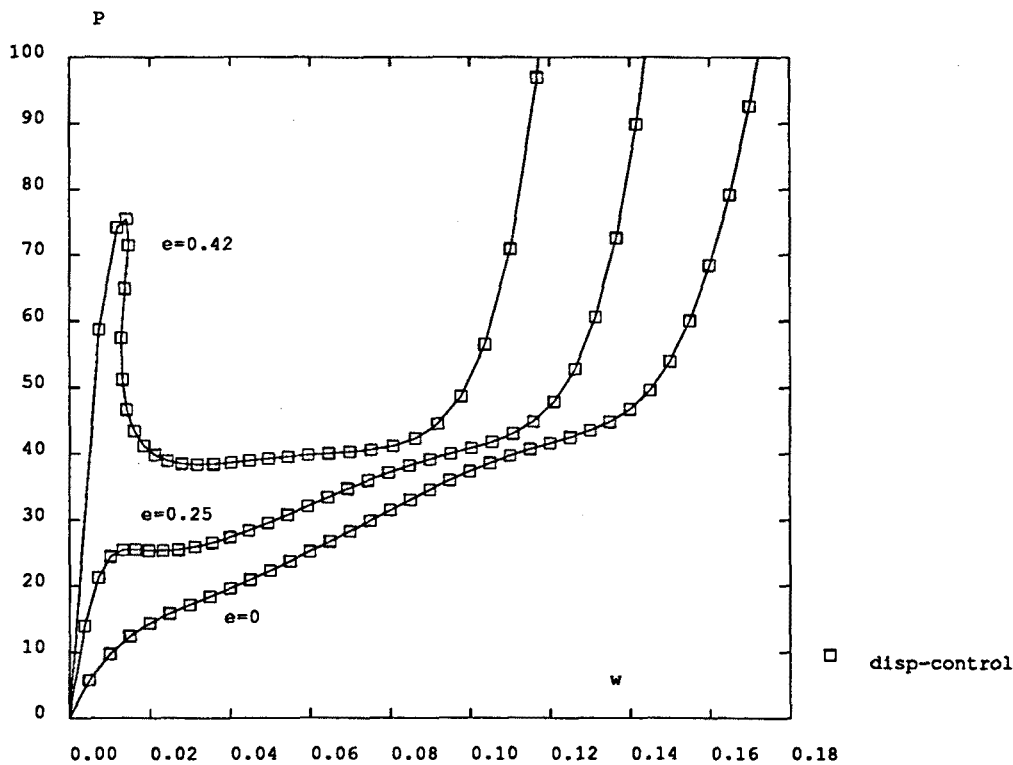


Figure C.3 Load P vs. deflection of loaded node; $e = 0$, $e = 0.25$, $e = 0.42$.

Furthermore, as it will be shown in the following examples, the growing complexity of the structure's behaviour (increasing number of limit points) is accentuated as the eccentricity of the load increases. In Figure C.4 the eccentricity is raised to a value of 0.50, and a qualitatively identical situation to the $e = 0.42$ case is obtained; that is, the $P - v$ curve presents snap-through and may thus be displacement-controlled, whereas the $P - w$ curve presents snap-back. Hence, for the $e = 0.42$ and $e = 0.50$ cases, arc-length techniques (restricted, for instance, to the $P - w$ plane) can be verified by

post-processing the data from a v -displacement-controlled computation. As a matter of fact, Figure C.4b shows the reference solution obtained after post-processing the data from a v -displacement-controlled computation, \square , and the arc-length results (restricted to the $P - \omega$ plane), \times , which as expected coincide with the reference.

C.4 Further considerations

In the previous Section the structure shows snap-through or snap-back depending on the controlled variable. However, when the eccentricity of the ring load is increased, for instance up to 0.60, this mixed behaviour is not present anymore. Figure C.5a shows the $P - v$ response obtained with a displacement-controlled technique, and the behaviour seems, at least qualitatively, correct. But if the $P - \omega$ curve is plotted as a post-process of the previous results, see Figure C.5b, serious doubts on the validity of the computations obviously appear. In this case, the **correct** response of the structure (i.e. initial **negative** deflections for the $P - v$ curve, and a smooth initial slope for the $P - \omega$ curve) must be determined with an arc-length algorithm, see Figure C.6. This solution can never be captured via a displacement-controlled strategy on any of the presented variables. In fact, if in an attempt to capture the correct response the initial displacement increments of v were drastically reduced, the computations would follow an equilibrium path with **negative** values of the load. Thus, an arc-length procedure must be employed. In this particular case, a cylindrical arc-length solution with automatic control of the arc length, (Crisfield, 1991), has been employed to determine Figure C.6.

Similar conclusions can be drawn if the eccentricity of the load is raised up to $e = 0.65$. Figure C.7 presents the unrealistic results obtained with a v -displacement-controlled strategy. On the other hand, Figure C.8 presents the correct behaviour obtained with an arc-length method. For even larger values of e , arc-length techniques are definitely necessary, see Figures C.9 and C.10.

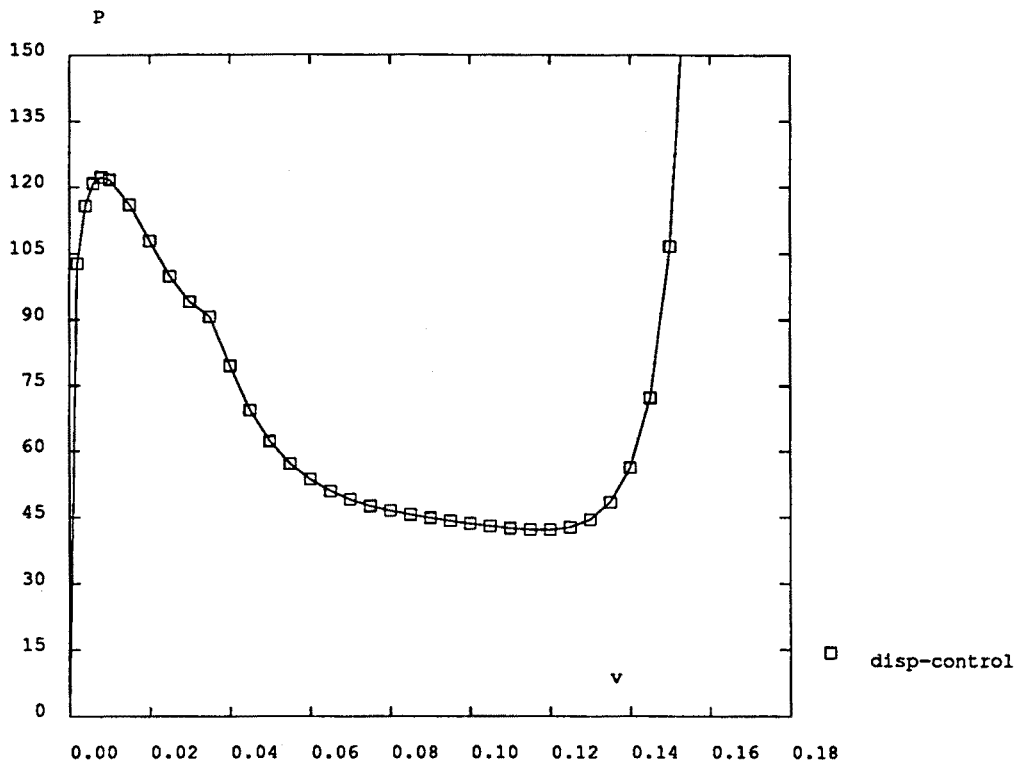


Fig. C.4a

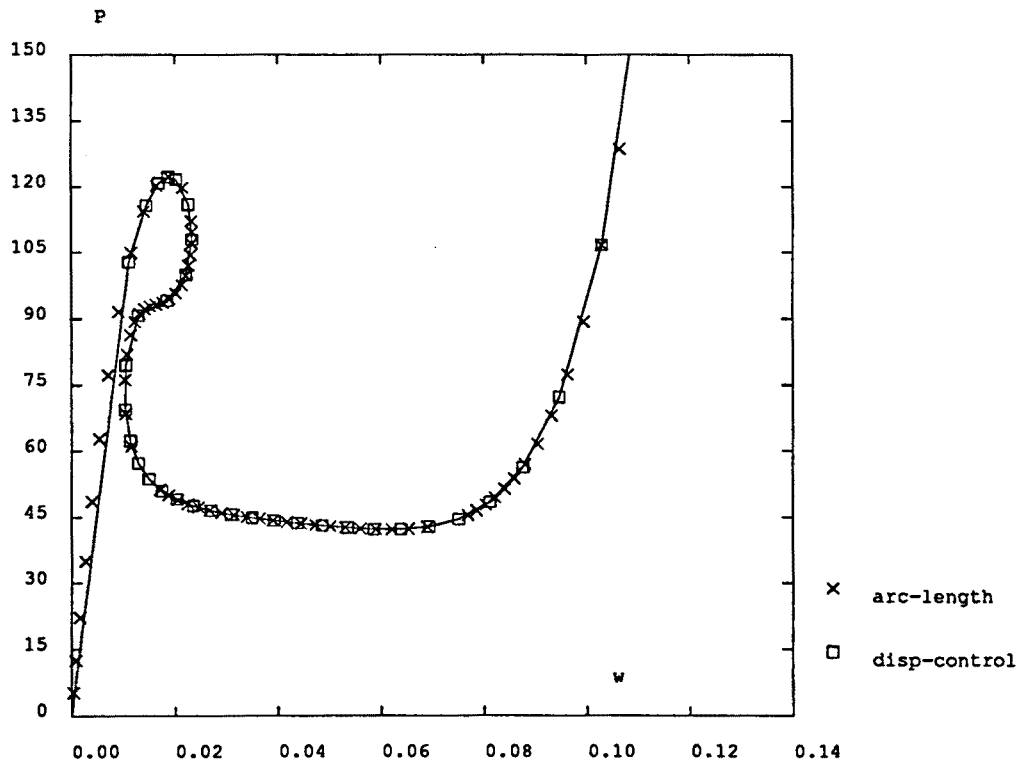


Fig. C.4b

Figure C.4 Response for an eccentricity $e = 0.50$: a) Load P vs. deflection of central node v ; b) Load P vs. deflection of loaded node w .

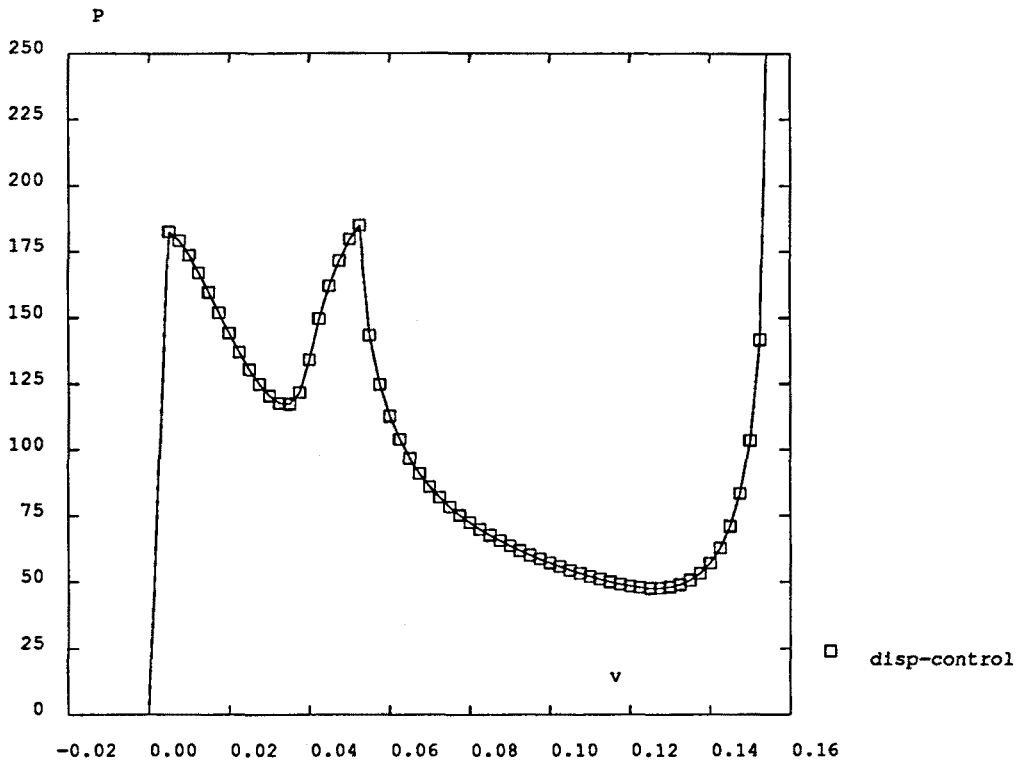


Fig. C.5a

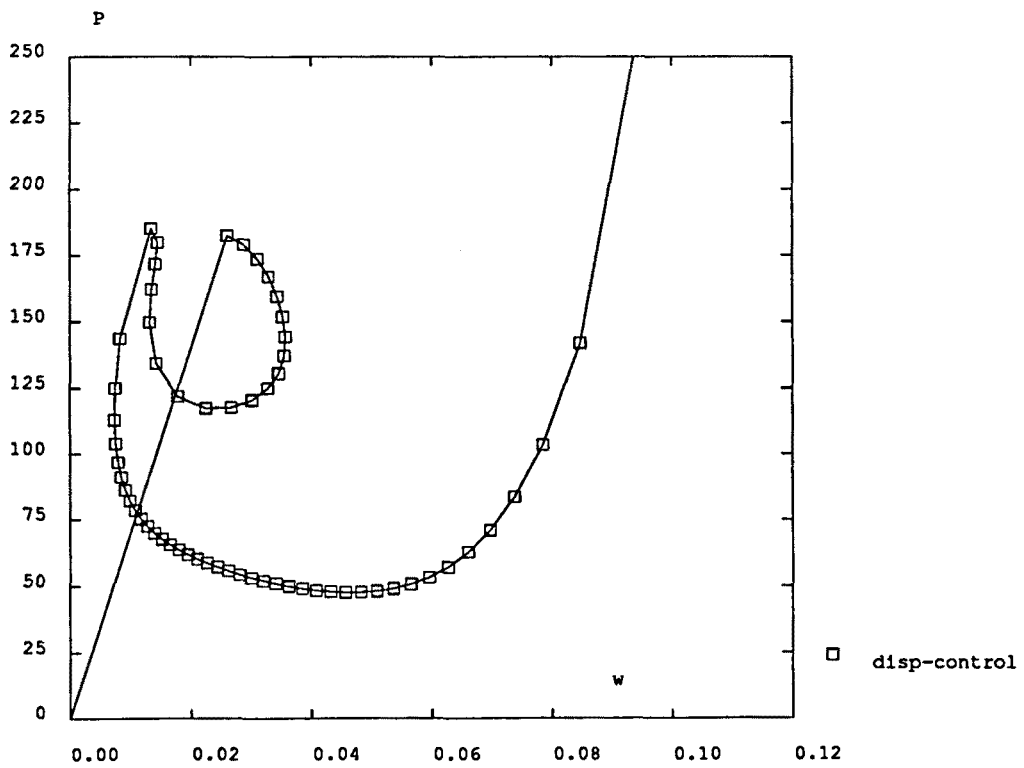


Fig. C.5b

Figure C.5 Displacement-controlled solution for an eccentricity $e = 0.60$: a) P versus v ; b) P versus ω .

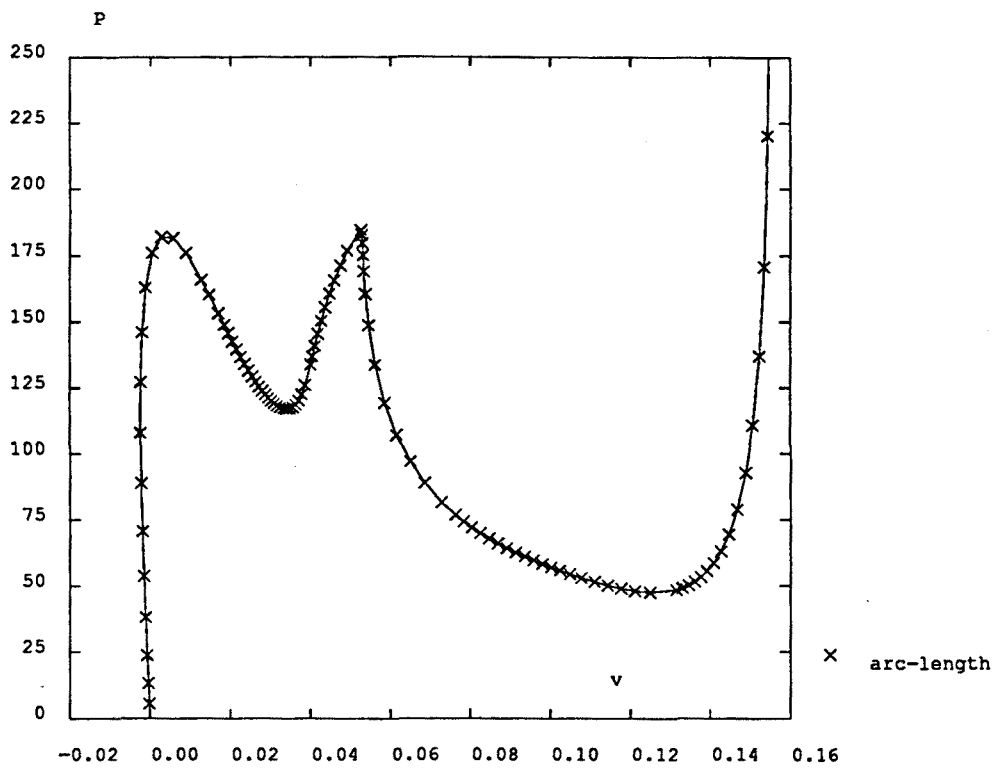


Fig. C.6a

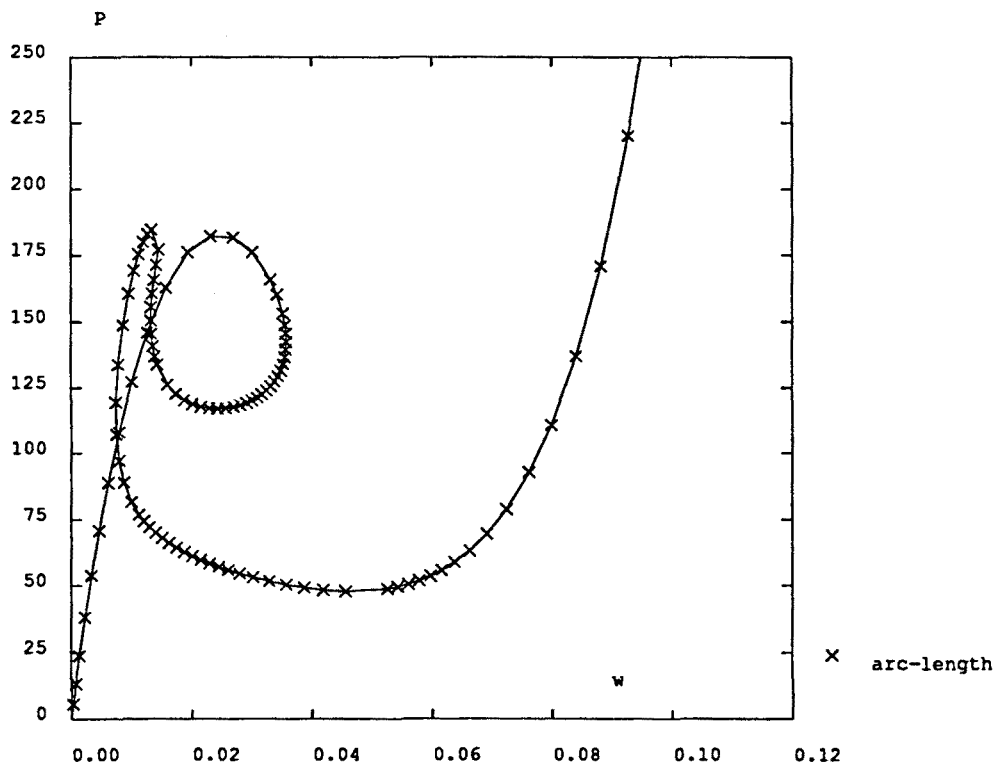


Fig. C.6b

Figure C.6 Arc-length solution for an eccentricity $e = 0.60$: a) P versus v ; b) P versus ω .

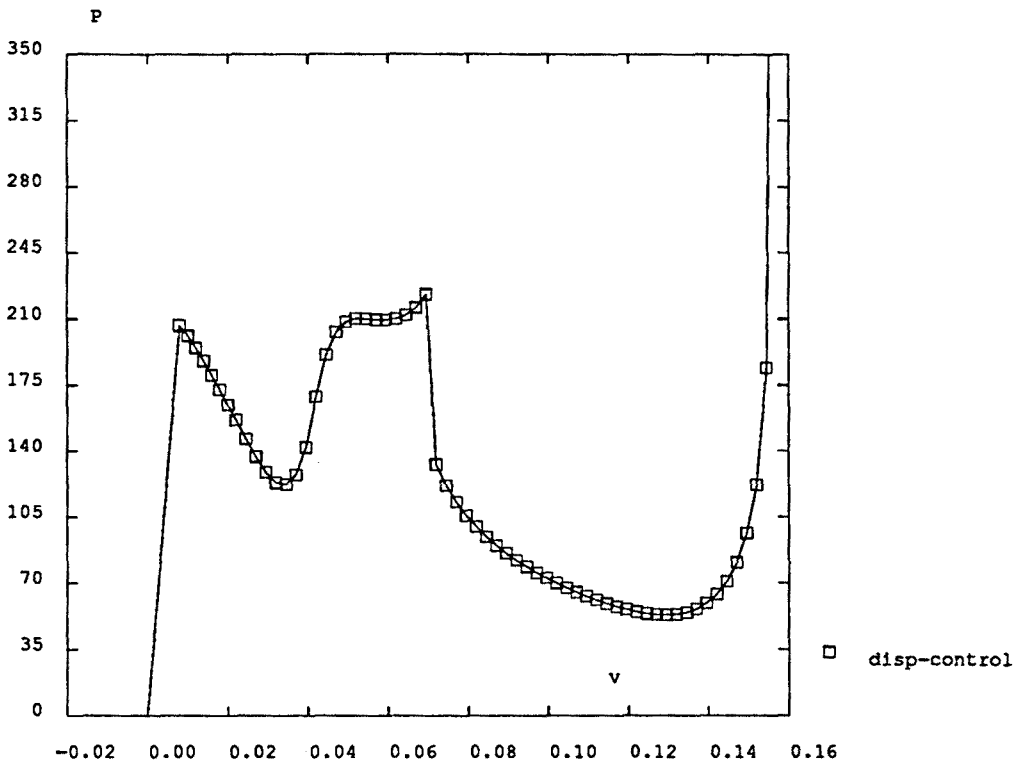


Fig. C.7a

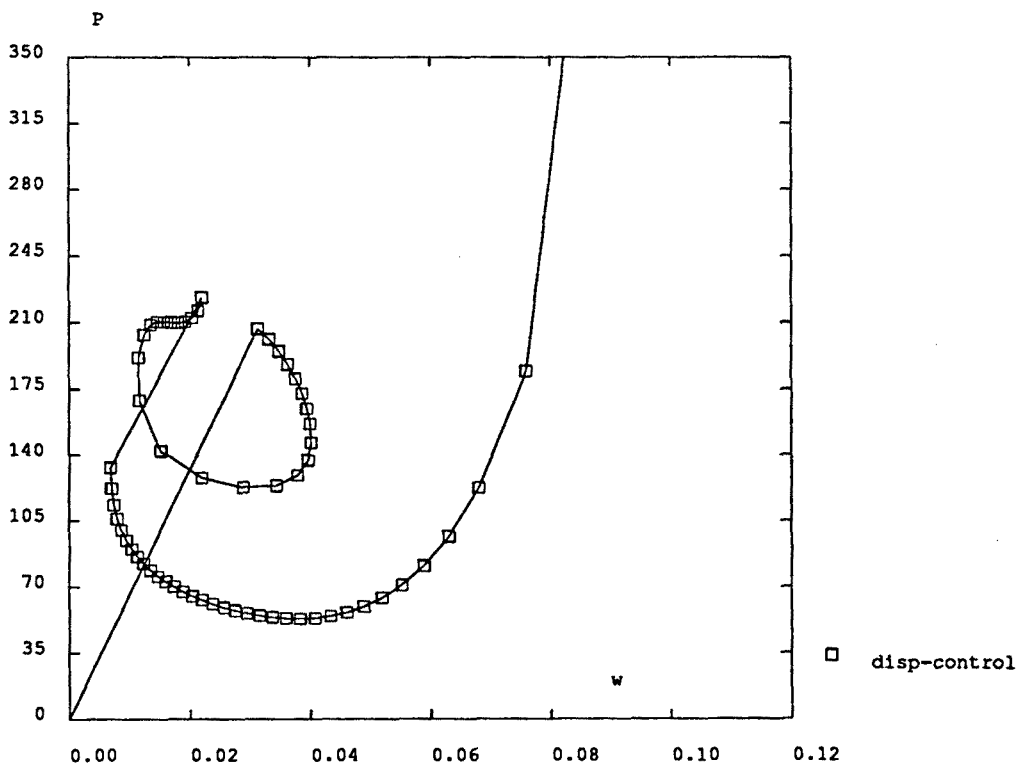


Fig. C.7b

Figure C.7 Displacement-controlled solution for an eccentricity $e = 0.65$: a) P versus v ; b) P versus w .

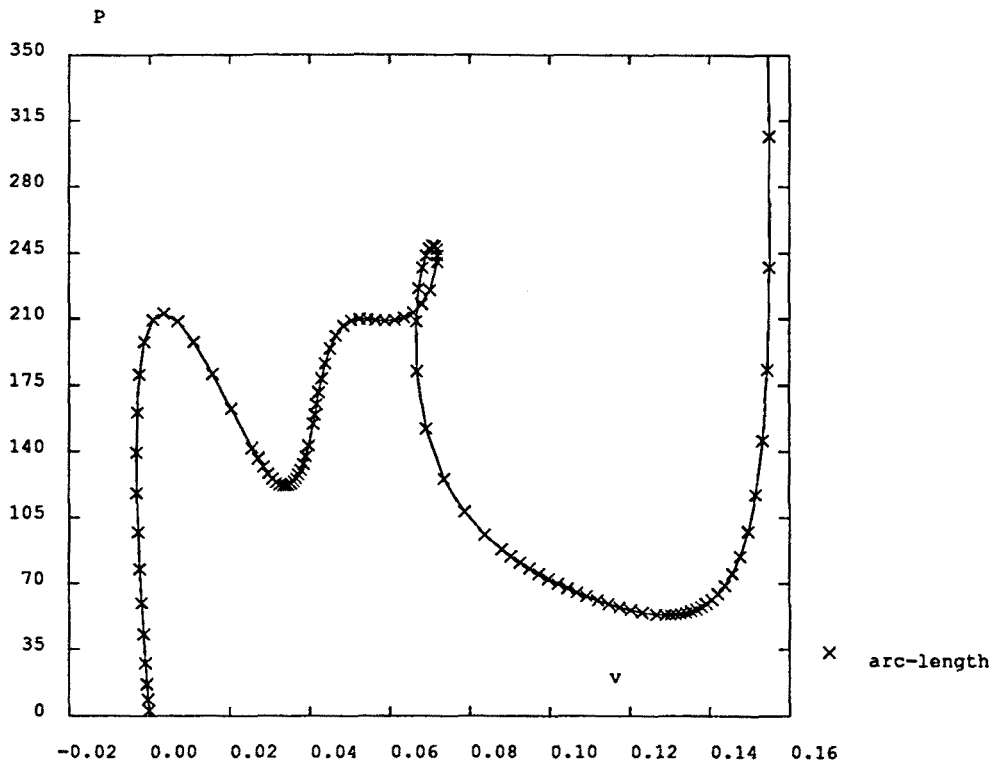


Fig. C.8a

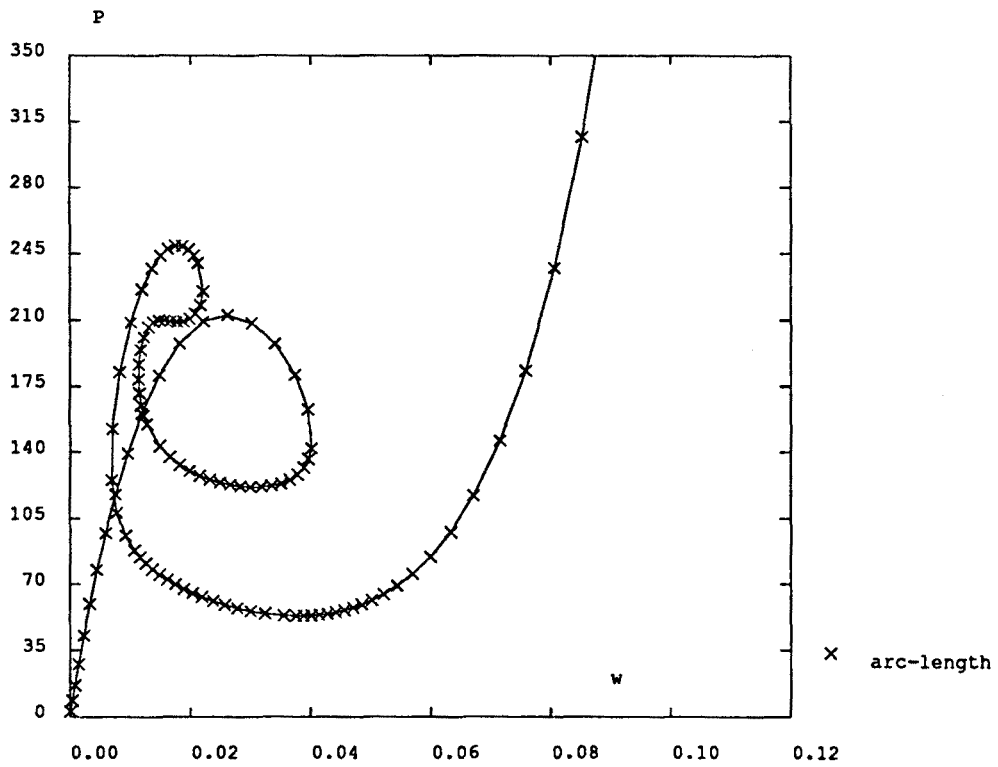


Fig. C.8b

Figure C.8 Arc-length solution for an eccentricity $e = 0.65$: a) P versus v ; b) P versus ω .

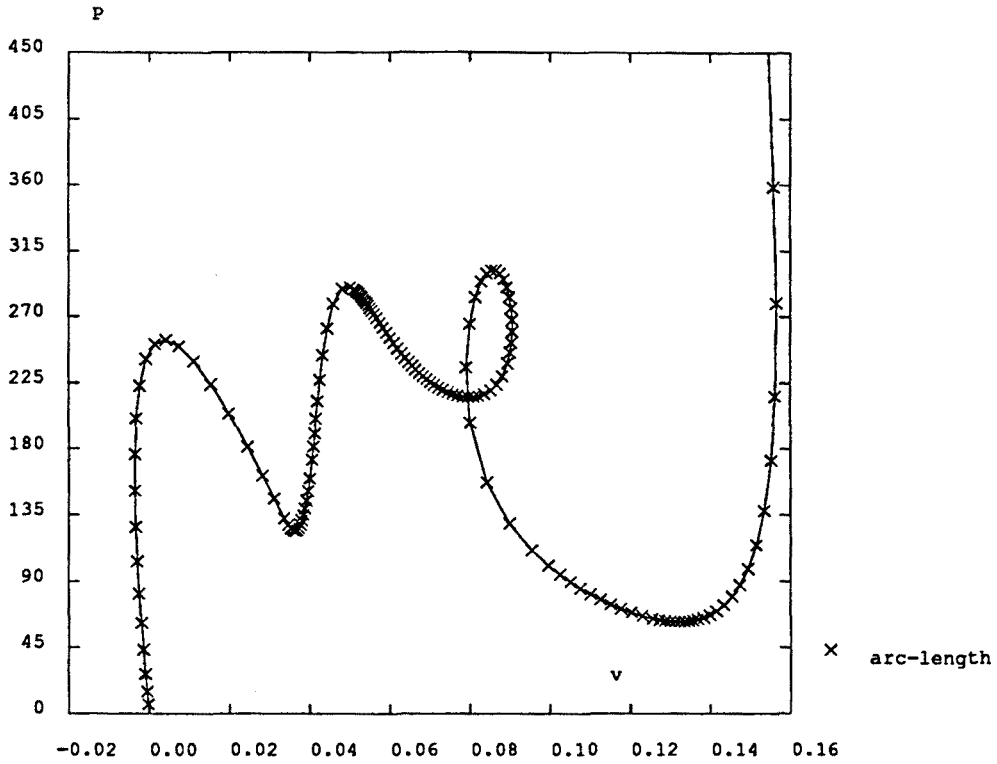


Fig. C.9a

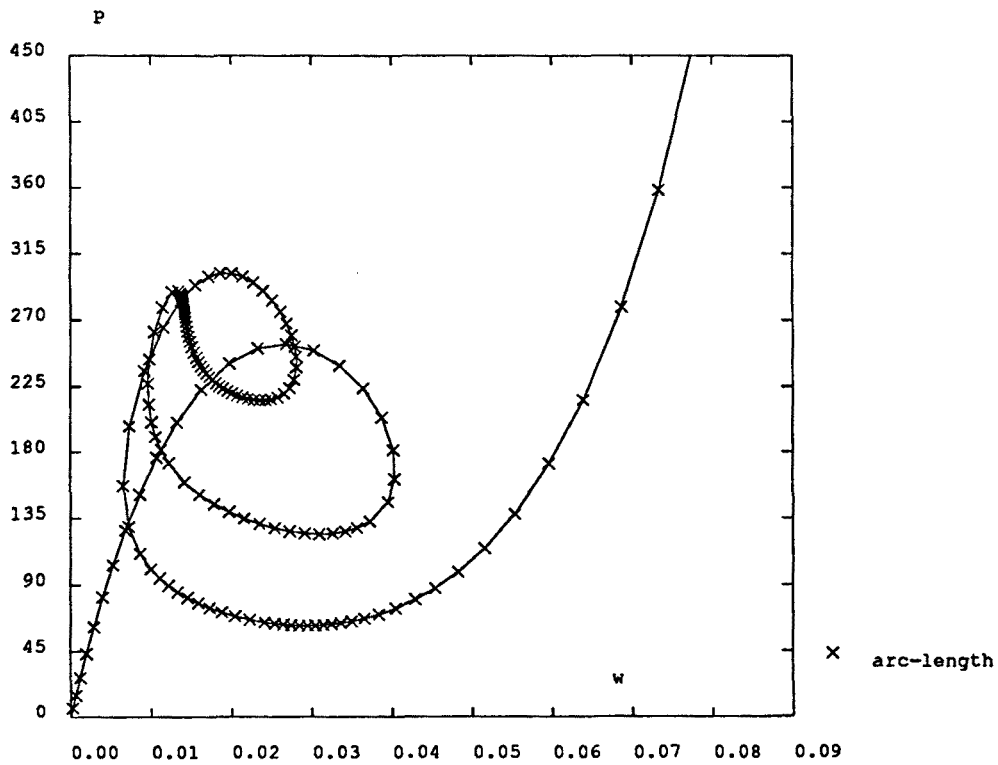


Fig. C.9b

Figure C.9 Response for an eccentricity $e = 0.70$: a) Load P vs. deflection of central node v ; b) Load P vs. deflection of loaded node ω .

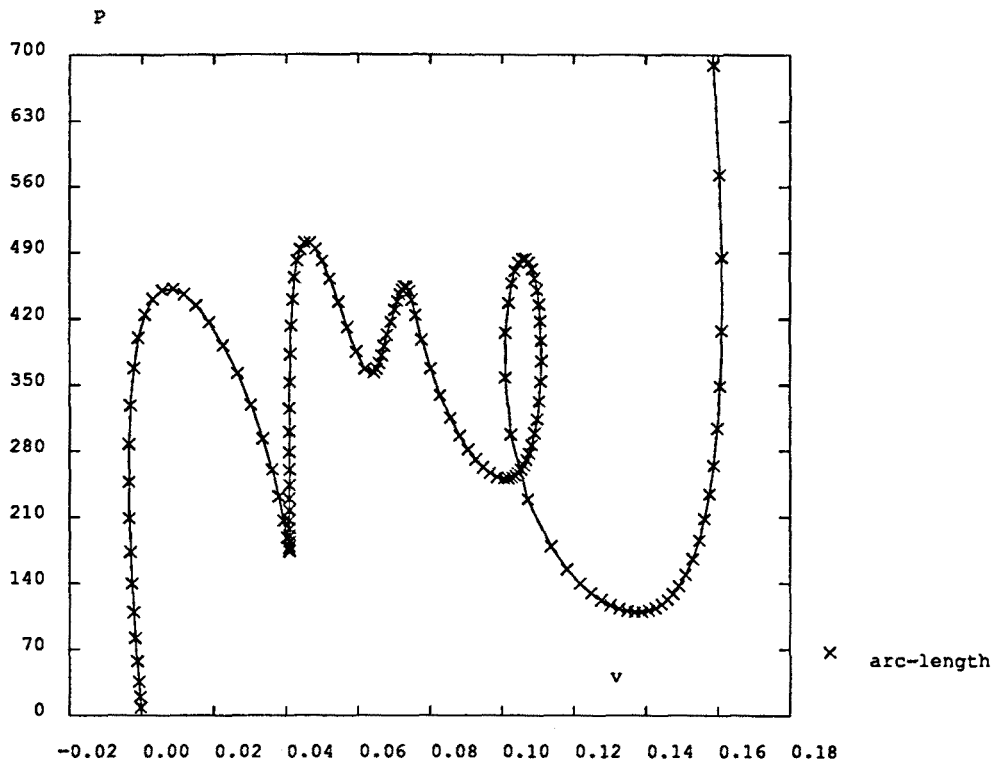


Fig. C.10a

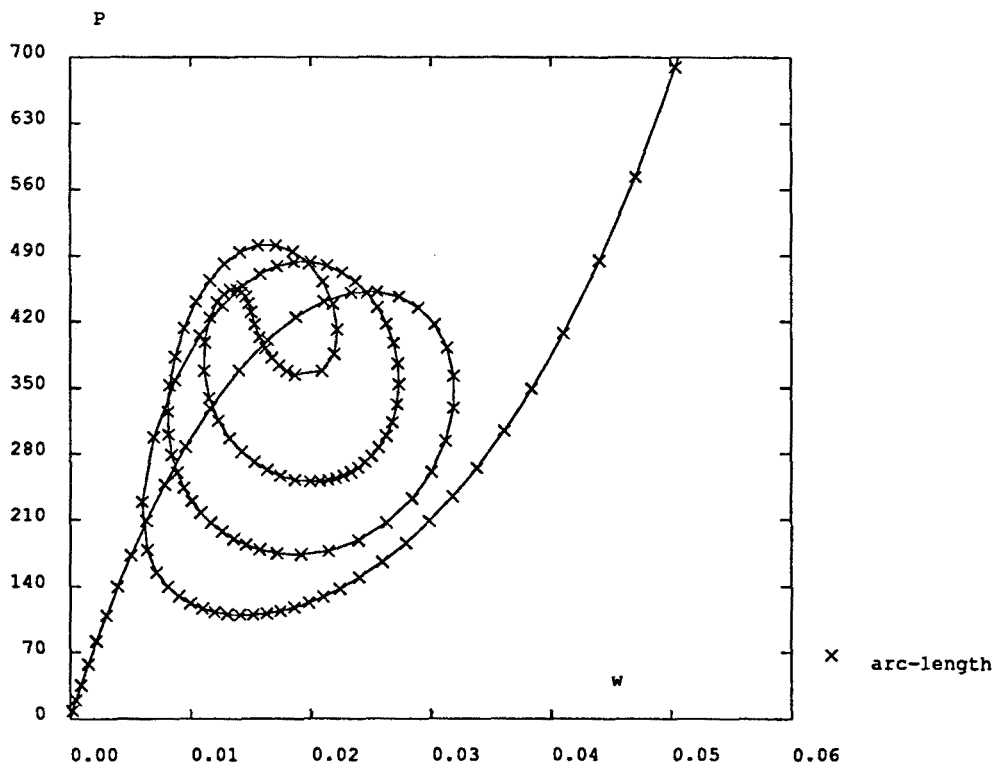


Fig. C.10b

Figure C.10 Response for an eccentricity $e = 0.80$: a) Load P vs. deflection of central node v ; b) Load P vs. deflection of loaded node w .

C.5 Analysis of the results and conclusions

A well-known spherical shell test has been analyzed from a numerical point of view for a complete range of eccentricities of the ring load, e . Notice that from a physical point of view other concerns (not addressed here) must be taken into account, apart from the numerical resolution of the nonlinear equations. Both the displacement of the apex, v , and the displacement under the ring load, w , have been employed as the controlling variables. Depending on the value of e the following results have been obtained, see Figure C.11:

- 1.— for low values of e , the problem can be solved with displacement control using either v or w , because the curves $P - v$ and $P - w$ do not show snap-back.
- 2.— for intermediate values of e , the displacement control is still valid for v , but the arc-length control is needed for w because the $P - w$ curve presents snap-back. Therefore, in the intermediate range, the displacement controlled solution (with v) can be employed to validate the arc-length implementation (with w).
- 3.— for large values of e , the arc-length method must be employed with either v or w , because both $P - v$ and $P - w$ curves show complex snap-back behaviour.

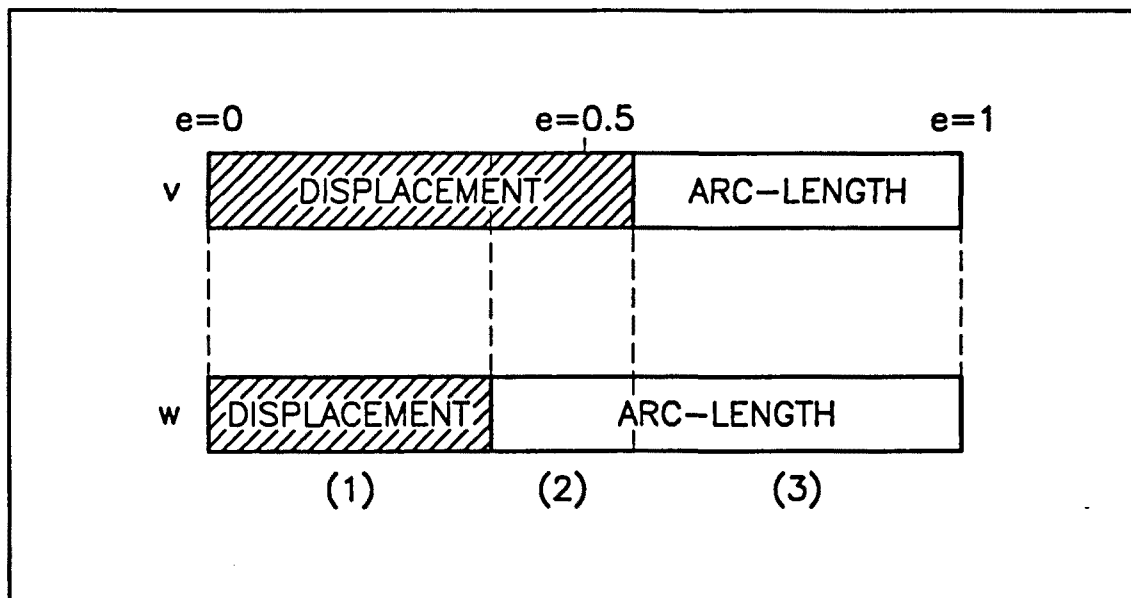


Figure C.11 Schematic representation of the controlling technique for each variable and the range of eccentricities.

REFERENCES

- Akkerman, R., Huétink, J. and van der Helm, P.N. (1995), "Finite Elements and Volumes in a Euler-Lagrange Formulation. Artificial Dissipation versus Limited Flux Schemes", *Proceedings of the Fourth International Conference on Computational Plasticity (COMPLAS IV)*, Barcelona, Vol. 2, pp. 2307-2318.
- Ausaverri, M. (1993), "Condiciones de contorno artificiales en dominios semiinfinitos. Aplicación a problemas de flujo en medio poroso", *Tesina de Especialidad*, E.T.S. de Ingenieros de Caminos, Universitat Politècnica de Catalunya, Barcelona (in Spanish).
- Baaijens, F.P.T. (1993), "An U-ALE Formulation of 3-D Unsteady Viscoelastic Flow", *International Journal for Numerical Methods in Engineering*, Vol. 36, pp. 1115-1143.
- Barlow, J. (1982), "Constraint Relationships in Linear and Nonlinear Finite Element Analyses", *International Journal for Numerical Methods in Engineering*, Vol. 18, pp. 521-533.
- Bathe, K.J. (1982), *Finite Element Procedures in Engineering Analysis*, Prentice Hall, New Jersey, USA.
- Bathe, K.J., Ramm, E., and Wilson, E.L. (1975), "Finite Element Formulations for Large Deformation Dynamic Analysis", *International Journal for Numerical Methods in Engineering*, Vol. 9, pp. 353-386.
- Bellini, P.X. and Chulya, A. (1987), "An Improved Automatic Incremental Algorithm for the Efficient Solution of Nonlinear Finite Element Equations", *Computers and Structures*, Vol. 26, pp. 99-110.

- Belytschko, T. (1983), "An Overview of Semidiscretization and Time Integration Procedures", Chapter 1 of *Computational Methods for Transient Analysis*, Eds. T. Belytschko and T.J.R. Hughes, Elsevier, New York, USA.
- Belytschko, T. and Kennedy, J.M. (1978), "Computer Models for Subassembly Simulation", *Nuclear Engineering and Design*, Vol. 49, pp. 17-38.
- Benson, D.J. (1989), "An Efficient, Accurate, Simple ALE Method for Nonlinear Finite Element Programs", *Computer Methods in Applied Mechanics and Engineering*, Vol. 72, pp. 305-350.
- Benson, D.J. (1992), "Computational Methods in Lagrangian and Eulerian Hydrocodes", *Computer Methods in Applied Mechanics and Engineering*, Vol. 99, pp. 235-394.
- Bretones, M.A. (1993), "Programación orientada al objeto. Una herramienta de ingeniería y desarrollo" *Tesina de Especialidad*, E.T.S. de Ingenieros de Caminos, Universitat Politècnica de Catalunya, Barcelona (in Spanish).
- Bretones, M.A., Rodríguez-Ferran, A. and Huerta, A. (1995), "La programación orientada al objeto aplicada al cálculo por el método de los elementos finitos", *Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería*, Vol. 11, pp. 423-449 (in Spanish).
- Bretones, M.A. (1996), *Personal communication*.
- Brillouin, L. (1949), *Les tenseurs en mécanique et en élasticité*, Masson, Paris (in French).
- Brodie, K.W., Gourlay, A.R. and Greenstadt, J. (1973), "Rank-one and Rank-two Corrections to Positive Definite Matrices Expressed in Product Form", *Journal of the Institute of Mathematics and its Applications*, Vol. 11, pp. 73-82.
- Broyden, C.G., Dennis, J.E. and Moré, J.J. (1973), "On the Local and Superlinear Convergence of Quasi-Newton Methods", *Journal of the Institute of Mathematics and its Applications*, Vol. 12, pp. 223-245.

- Casadei, F., Donéa, J. and Huerta, A. (1995), "Arbitrary Lagrangian-Eulerian Finite Elements in Non-Linear Fast Transient Continuum Mechanics", *Report EUR 16327 EN*, Institute for Safety Technology, Joint Research Centre Ispra, European Commission.
- Cervera, M. (1986), "Nonlinear Analysis of Reinforced Concrete Structures Using Three Dimensional and Shell Finite Element Models", *Ph.D. Thesis*, University College of Swansea, Swansea.
- Crisfield, M.A. (1980), "A Fast Incremental/Iterative Solution Procedure that Handles 'Snap-Through' ", *Computers and Structures*, Vol. 13, pp. 55-62.
- Crisfield, M.A. (1983), "An Arc-Length Method Including Line Searches and Accelerations", *International Journal for Numerical Methods in Engineering*, Vol. 19, pp. 1269-1289.
- Crisfield, M.A. (1991), *Non-linear Finite Element Analysis of Solids and Structures*, John Wiley and Sons Ltd., England.
- Dennis, J.E. and Moré, J.J. (1977), "Quasi-Newton Methods, Motivation and Theory", *SIAM Review*, Vol. 19, pp. 46-89.
- Dennis, J.E. and Schnabel, R.B. (1983), *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice Hall Series in Computational Mathematics, Englewood Cliffs, New Jersey, USA.
- Dennis, J.E. and Walker, H.F. (1981), "Convergence Theorems for Least-Change Secant Update Methods", *SIAM Journal on Numerical Analysis*, Vol. 18, pp. 949-987.
- Dieter, G.E. (1976), *Mechanical Metallurgy*, International Student Edition, Mc Graw-Hill Kogakusha Ltd., Tokyo.
- Díez, P., Egozcue, J.J. and Huerta, A. (1995), "A Posteriori Error Estimation for Standard Finite Element Analysis", *Publication CIMNE No. 75*, International Centre for Numerical Methods in Engineering, Barcelona.
- Donéa, J. (1983), "Arbitrary Lagrangian-Eulerian Finite Element Methods", Chapter 10 of *Computational Methods for Transient Analysis*, Eds. T. Belytschko and

- T.J.R. Hughes, Elsevier, New York, USA.
- Donéa, J. (1984), "A Taylor-Galerkin Method for Convective Transport Problems", *International Journal for Numerical Methods in Engineering*, Vol. 20, pp. 101-120.
- Donéa, J., Fasoli-Stella, P. and Giuliani, S. (1977), "Lagrangian and Eulerian Finite Element Techniques for Transient Fluid-Structure Interaction Problems", Paper B1/2, *Transactions of the 4th SMiRT Conference (Structural Mechanics in Reactor Technology)*, San Francisco, USA.
- Donéa, J. and Quartapelle, L. (1992), "An Introduction to Finite Element Methods for Transient Advection Problems", *Computer Methods in Applied Mechanics and Engineering*, Vol. 95, pp. 169-203.
- Dubois-Pèlerin, Y., Zimmermann, Th. and Bomme, P. (1992), "Object-Oriented Finite Element Programming Concepts", in *New Advances in Computational Structural Mechanics*, Eds. P. Ladevèze and O.C. Zienkiewicz, Elsevier, pp. 457-466.
- Engelman, M.S., Strang, G. and Bathe, K.J. (1981), "The Application of Quasi-Newton Methods in Fluid Mechanics", *International Journal for Numerical Methods in Engineering*, Vol. 17, pp. 707-718.
- Fafard, M. and Massicotte, B. (1993), "Geometrical Interpretation of the Arc-Length Method", *Computers and Structures*, Vol. 46, pp. 603-615.
- Fletcher, C.A.J. (1987a), *Computational Techniques for Fluid Dynamics. Volume I: Fundamental and General Techniques*, Springer Series in Computational Physics, Springer-Verlag, Berlin.
- Fletcher, R. (1987b), *Practical Methods of Optimization*, John Wiley and Sons, Chichester, England.
- Galindo, M. (1994), "FemLab Version 1.0", *Technical Report No. IT-114*, International Centre for Numerical Methods in Engineering, Barcelona.
- García Garino, C. (1993), "Un modelo numérico para el análisis de sólidos elastoplásticos sometidos a grandes deformaciones", *Doctoral Thesis*, E.T.S. de Ingenieros de Caminos, Universitat Politècnica de Catalunya, Barcelona (in Spanish).

- Gerardin, M., Hogge, M. and Idelsohn, S. (1983), "Implicit Finite Element Methods", Chapter 9 of *Computational Methods for Transient Analysis*, Eds. T. Belytschko and T.J.R. Hughes, Elsevier, New York, USA.
- Ghosh, S. and Kikuchi, N. (1991), "An Arbitrary Lagrangian-Eulerian Finite Element Method for Large Deformation Analysis of Elastic-Viscoplastic Solids", *Computer Methods in Applied Mechanics and Engineering*, Vol. 86, pp. 127-188.
- Halleux, J.P. and Casadei, F. (1987), "Transient Large-Strain Finite Element Analysis of Solids", in *Computational Methods for Non-Linear Problems*, Eds. C. Taylor, D.R.J. Owen and E. Hinton, Pineridge, Swansea.
- Hirt, C.W., Amsden, A.A. and Cook, J.L. (1974), "An Arbitrary-Lagrangian-Eulerian Computing Method for All Flow Speeds", *Journal of Computational Physics*, Vol. 14, 227.
- Huerta, A. and Casadei, F. (1991), "Arbitrary Lagrangian-Eulerian Formulation for Large Boundary Motion in Nonlinear Continuum Mechanics", *Technical Note No. 1.91.95*, Safety Technology Institute, Joint Research Centre Ispra, European Commission. Also in *Internal Report MA006/1991*, E.T.S. de Ingenieros de Caminos, Universitat Politècnica de Catalunya.
- Huerta, A. and Casadei, F. (1994), "New ALE Applications in Non-Linear Fast-Transient Solid Dynamics", *Engineering Computations*, Vol. 11, pp. 317-145.
- Huerta, A., Casadei, F. and Donéa, J. (1995), "ALE Stress Update in Transient Plasticity Problems", *Proceedings of the Fourth International Conference on Computational Plasticity (COMPLAS IV)*, Barcelona, Vol. 2, pp. 1865-1876.
- Huerta, A., Díez, P. and Egozcue, J.J. (1996), "An A-Posteriori Error Estimator for Classical Linear and Nonlinear Finite Elements", *Proceedings of the Second Ecomas Conference on Numerical Methods in Engineering*.
- Huerta, A. and Liu, W.K. (1988), "Viscous Flow with Large Free Surface Motion", *Computer Methods in Applied Mechanics and Engineering*, Vol. 69, pp. 277-324.
- Huerta, A. and Liu, W.K. (1989), "ALE Formulation for Large Boundary Motion",

- Transactions of the 10th SMiRT Conference (Structural Mechanics in Reactor Technology)*, Anaheim, California, USA, Vol. B, pp. 335-346.
- Huerta, A. and Liu, W.K. (1990), "Large Amplitude Sloshing with Submerged Blocks", *Journal of Pressure Vessel Technology, ASME*, Vol. 112, pp. 104-108.
- Huétink, J. (1986), "On the Simulation of Thermo-Mechanical Forming Processes", *Dissertation*, University of Twente, The Netherlands.
- Huétink, J., Vreede, P.T. and Van der Lugt, J. (1990), "Progress in Mixed Eulerian-Lagrangian Finite Element Simulation of Forming Processes", *International Journal for Numerical Methods in Engineering*, Vol. 30, pp. 1441-1457.
- Hughes, T.J.R. (1984), "Numerical Implementation of Constitutive Models: Rate-Independent Deviatoric Plasticity", Chapter 2 of *Theoretical Foundation for Large-scale Computations for Non-linear Material Behavior*, Eds. S. Nemat-Nasser, R.J. Asaro and G.A. Hegemier, Martinus Nijhoff Publishers, Dordrecht.
- Hughes, T.J.R. (1987), *The Finite Element Method*, Prentice Hall International, Stanford, USA.
- Hughes, T.J.R., Liu, W.K. and Zimmermann, T.K. (1978), "Lagrangian-Eulerian Finite Element Formulation for Incompressible Viscous Flows", *US-Japan Seminar on Interdisciplinary Finite Element Analysis*, Cornell University, Ithaca, New York, USA.
- Hughes, T.J.R., Liu, W.K. and Zimmermann, T.K. (1981), "Lagrangian-Eulerian Finite Element Formulation for Incompressible Viscous Flows", *Computer Methods in Applied Mechanics and Engineering*, Vol. 29, pp. 329-349.
- Hughes, T.J.R. and Winget, J. (1980), "Finite Rotation Effects in Numerical Integration of Rate Constitutive Equations Arising in Large-Deformation Analysis", *International Journal for Numerical Methods in Engineering*, Vol. 15, pp. 1862-1867.
- Kelley, C.T. (1995), *Iterative Methods for Linear and Nonlinear Equations*, SIAM, Philadelphia.

- Khan, A.S. and S. Huang (1995), *Continuum Theory of Plasticity*, John Wiley and Sons, New York.
- Kouhia, R. and Mikkola, M. (1989), "Tracing the Equilibrium Paths Beyond Simple Critical Points", *International Journal for Numerical Methods in Engineering*, Vol. 28, pp. 2923-2941.
- L.A.M.S. (1988), "CASTEM2000 – Manuel d'utilisation", Rapport 88/176, Laboratoire d'Analyse Mécanique des Structures, Commissariat à l'Énergie Atomique, France (in French).
- Lax, P.D. and Wendroff, B. (1960), "Systems of Conservation Laws", *Communications on Pure and Applied Mathematics*, Vol. 13, pp. 217-237.
- Lax, P.D. and Wendroff, B. (1964), "Difference Schemes for Hyperbolic Equations with High-Order of Accuracy", *Communications on Pure and Applied Mathematics*, Vol. 17, pp. 381.
- Le Veque, R.J. (1990), *Numerical Methods for Conservation Laws*, Lectures in Mathematics, ETH Zürich, Birkhäuser Verlag, Basel.
- Liu, W.K., Belytschko, T. and Chang, H. (1986), "An Arbitrary Lagrangian-Eulerian Finite Element Method for Path-Dependent Materials", *Computer Methods in Applied Mechanics and Engineering*, Vol. 58, pp. 227-245.
- Liu, W.K., Chang, H., Chen, J.S. and Belytschko, T. (1988), "Arbitrary Lagrangian-Eulerian Petrov-Galerkin Finite Elements for Nonlinear Continua", *Computer Methods in Applied Mechanics and Engineering*, Vol. 68, pp. 259-310.
- Lubliner, J. (1990) *Plasticity Theory*, Macmillan Publishing Company, New York, USA.
- Mackie, R.I. (1992), "Object-Oriented Programming of the FEM", *International Journal for Numerical Methods in Engineering*, Vol. 35, pp 425-436.
- Malvern, L.W. (1969), *Introduction to the Mechanics of a Continuous Medium*, Prentice-Hall Series in Engineering of the Physical Sciences, Englewood Cliffs, New Jersey, USA.

- Marsden, J.E. and Hughes, T.J.R. (1983), *Mathematical Foundations of Elasticity*, Prentice-Hall, Englewood Cliffs, New Jersey, USA.
- Matthies, H., and Strang, G. (1979), "The Solution of Nonlinear Finite Element Equations", *International Journal for Numerical Methods in Engineering*, Vol. 14, pp. 1613-1626.
- Montolío, N., Sarrate, J. and Huerta, A. (1995), "Generador recursivo de mallas cuadrangulares no estructuradas", *Proceedings del V Congreso Español de Informática Gráfica*, pp. 225-236 (in Spanish).
- Miller, G.R. (1991), "An Object-Oriented Approach to Structural Analysis and Design", *Computers and Structures*, Vol. 40, pp. 75-82.
- Noh, W.F. (1964), "CEL: A Time-Dependent Two-Space-Dimensional Coupled Eulerian-Lagrangian Code", in *Methods in Computational Physics*, Vol. 3, Eds. B. Alder, S. Fernbach and M. Rotenberg, Academic Press, New York.
- NUMIFORM (1995), "Simulation of Materials Processing: Theory, Methods and Applications", *Proceedings of the Fifth International Conference on Numerical Methods in Industrial Forming Processes*, Ithaca, New York.
- Oden, J.T. (1972), *Finite Elements of Nonlinear Continua*, Advanced Engineering Series, Mc Graw-Hill, New York.
- Oldroyd, J.G. (1950), "On the Formulation of Rheological Equations of State", *Proceedings of the Royal Society of London*, Vol. A200, pp. 523-541.
- Oñate, E. (1991), *Cálculo de estructuras por el método de los elementos finitos. Análisis estático lineal*, Centro Internacional de Métodos Numéricos en Ingeniería, Barcelona (in Spanish).
- Ortega, J.M., and Rheinboldt, W.C. (1970), *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, California, USA.
- Ortiz, S. (1993), "Simulació numèrica d'ones sòniques", *Tesina d'Especialitat*, E.T.S. de Ingenieros de Caminos, Universitat Politècnica de Catalunya, Barcelona (in Catalan).

- Papadrakakis, M. and Pantazopoulos, G. (1993), "A Survey of Quasi-Newton Methods with Reduced Storage", *International Journal for Numerical Methods in Engineering*, Vol. 36, pp. 1573-1596.
- Pegon, P. (1993), "Model Implementation in CASTEM 2000: Some General Considerations and a Simple Practical Realization", *Technical Note No. I.93.06*, Safety Technology Institute, Joint Research Centre Ispra, Commission of the European Communities.
- Pegon, P., and Anthoine, A. (1994), "Numerical Strategies for Solving Continuum Damage Problems Involving Softening: Application to the Homogenization of Masonry", *Proceedings of the Second International Conference on Computational Structures Technology*, Athens.
- Pegon, P. and Guélin, P. (1986), "Finite Strain Plasticity in Convected Frames", *International Journal for Numerical Methods in Engineering*, Vol. 22, pp. 521-545.
- Peraire, J. Vahdati, M., Morgan, K. and Zienkiewicz, O.C. (1991), "Adaptive Remeshing for Compressible Flow Computations", *Journal of Computational Physics*, Vol. 72, pp. 449-466.
- Pijaudier-Cabot, G., Bodé, L. and Huerta, A. (1995), "Arbitrary Lagrangian-Eulerian Finite Element Analysis of Strain Localization in Transient Problems", *International Journal for Numerical Methods in Engineering*, Vol. 38, pp. 4171-4191.
- Pinsky, P.M., Ortiz, M., and Pister, K.S. (1983), "Numerical Integration of Rate Constitutive Equations in Finite Deformation Analysis", *Computer Methods in Applied Mechanics and Engineering*, Vol. 40, pp. 137-158.
- Ponthot, J.P. (1995), "Traitement Unifié de la Mécanique des Milieux Continus Solides en Grandes Transformations par la Méthode des Éléments Finis", *Doctoral Thesis*, University of Liège (in French).
- Richtmyer, R.D. and Morton, K.W. (1967), *Difference Methods for Initial-Value Problems*, Interscience, New York.
- Riks, E. (1979), "An Incremental Approach to the Solution of Snapping and Buckling Problems", *International Journal of Solids and Structures*, Vol. 15, pp. 529-551.

- Riks, E. (1992) "On Formulations of Path-Following Techniques for Structural Stability Analysis", in *New Advances in Computational Structural Mechanics*, Eds. P. Ladevèze and O.C. Zienkiewicz, Elsevier, pp. 65-79.
- Rodríguez-Ferran, A. and Huerta, A. (1994), "A Comparison of Two Objective Stress Rates in Object-Oriented Codes", *Monograph No. 26*, International Centre for Numerical Methods in Engineering, Barcelona.
- Rodríguez-Ferran, A. and Huerta, A. (1995), "ALE Quasistatic Analysis in an Object-Oriented Code", *Proceedings of the Fourth International Conference on Computational Plasticity (COMPLAS IV)*, Barcelona, Vol. 2, pp. 2349-2360.
- Rodríguez-Ferran, A. and Huerta, A. (1996), "Comparing Two Algorithms to Add Large Strains to a Small Strain Finite Element Code", *Research Report No. 91*, International Centre for Numerical Methods in Engineering, Barcelona.
- Rodríguez-Ferran, A., Pegon, P., and Huerta, A. (1996), "Two Stress Update Algorithms for Large Strain Solid Mechanics. Part I: Derivation and Accuracy Analysis. Part II: Numerical Implementation and Validation", *Research Report No. 92*, International Centre for Numerical Methods in Engineering, Barcelona.
- Rodríguez-Ferran, A., Vila, A. and Huerta, A. (1996), "Adapting Nonlinear Solvers to Handle Linear Constraints Imposed via Lagrange Multipliers", *Proceedings of the Second Eccomas Conference on Numerical Methods in Engineering*.
- Sarrate, J. (1996), "Modelización numérica de la interacción fluido-sólido rígido: desarrollo de algoritmos, generación de mallas y adaptabilidad", *Doctoral Thesis*, E.T.S. de Ingenieros de Caminos, Universitat Politècnica de Catalunya, Barcelona (in Spanish).
- Sarrate, J., Gutiérrez, M.A. and Huerta, A. (1993), "A New Algorithm for Unstructured Quadrilateral Mesh Generation and Rezoning", *Proceedings of the Eighth International Conference on Finite Elements in Fluids*, Barcelona, Vol. 1, pp. 745-754.
- Schreurs, P. (1983), "Numerical Simulation of Forming Processes. The Use of the Arbitrary-Eulerian-Lagrangian (AEL) Formulation and the Finite Element

- Method", *Dissertation*, University of Eindhoven, The Netherlands.
- Schreyer, H.L. and Parsons, D.A. (1995), "Direct Application of Constraints to Symmetric Algebraic Systems", *Communications in Numerical Methods in Engineering*, Vol. 11, pp. 563-573.
- Schweizerhof, K.H. and Wriggers, P. (1986), "Consistent Linearization for Path Following Methods in Nonlinear FE Analysis", *Computer Methods in Applied Mechanics and Engineering*, Vol. 59, pp. 261-279.
- Shephard, M.S. (1984), "Linear Multipoint Constraints Applied Via Transformation as Part of a Direct Stiffness Assembly Process", *International Journal for Numerical Methods in Engineering*, Vol. 20, pp. 2107-2112.
- Simo, J.C. (1988), "A Framework for Finite Strain Elastoplasticity Based on Maximum Plastic Dissipation and the Multiplicative Decomposition. Part II: Computational Aspects", *Computer Methods in Applied Mechanics and Engineering*, Vol. 68, pp. 1-31.
- Soria, A. (1990), "Contribución al análisis de transitorios térmicos accidentales en los componentes de un reactor nuclear", *Doctoral Thesis*, Universidad Politécnica de Madrid (in Spanish).
- Soria, A. and Pegon, P. (1990), "Quasi-Newton Iterative Strategies Applied to the Heat Diffusion Equation", *International Journal for Numerical Methods in Engineering*, Vol. 30, pp. 661-677.
- Soria, A. and Pegon, P. (1992), "Some Introductory Remarks About CASTEM2000", *Personal communication*, Applied Mechanics Division, Joint Research Centre Ispra, Commission of the European Communities.
- Soria, A. and Pegon, P. (1993), "An Arc Length Control Procedure to Solve Parabolic Problems", *Computer Methods in Applied Mechanics and Engineering*, Vol. 106, pp. 27-50.
- Truesdell, C. (1953), Corrections and Additions to "The Mechanical Foundations of Elasticity and Fluid Dynamics", *Journal of Rational Mechanics and Analysis*, Vol. 2, pp. 505-616.

- Truesdell, C. (1955), "The Simplest Rate Theory of Pure Elasticity", *Communications on Pure and Applied Mathematics*, Vol. VIII, pp. 123-132.
- Truesdell, C. and Noll, W. (1965), *The Non-Linear Field Theories of Mechanics*, *Encyclopedia of Physics*, Ed. S. Flügge, Vol. III/3, Springer-Verlag, Berlin.
- Truesdell, C. and Toupin, R. (1960), *The Classical Field Theories*, *Encyclopedia of Physics*, edited by S. Flügge, Vol. III/1, Springer-Verlag, Berlin.
- Trulio, J.G. (1966), Air Force Weapons Laboratory, AFWL-Tr-66-19 (June 1966).
- Van der Boogaard, A.H. and de Borst, R. (1994), "An Adaptive Time-Stepping Algorithm for Quasistatic Processes", *Communications in Numerical Methods in Engineering*, Vol. 10, pp. 837-844.
- Verpeaux, P. and Millard, A. (1988), "De l'existence à l'essence. De CASTEM à CASTEM2000. Quelques considérations sur le développement de grans codes du calcul", *Rapport 88/179*, Laboratoire d'Analyse Mécanique des Structures, Commissariat à l'Énergie Atomique, France (in French).
- Vila, A., Rodríguez-Ferran, A. and Huerta, A. (1995), "Nonlinear Finite Element Techniques Using an Object-Oriented Code", *Monograph No. 31*, International Centre for Numerical Methods in Engineering, Barcelona.
- Vila, A., Rodríguez-Ferran, A., and Huerta, A. (1996), "A Note on a Numerical Benchmark Test: an Axisymmetric Shell Under Ring Loads", *Research Report No. 85*, International Centre for Numerical Methods in Engineering, Barcelona.
- Webb, J.P. (1990), "Imposing Linear Constraints in Finite-Element Analysis", *Communications in Numerical Methods in Engineering*, Vol. 6, pp. 471-475.
- Wempner, G.A. (1971), "Discrete Approximations Related to Nonlinear Theories of Solids", *International Journal of Solids and Structures*, Vol. 7, pp. 1581-1599.
- Zienkiewicz, O.C. (1984), "Flow Formulation for Numerical Solution of Forming Processes", in *Numerical Analysis of Forming Processes*, Eds. J.F.T. Pittman *et al.*, Wiley, Chichester, pp 1-44.

- Zienkiewicz, O.C. and Huang, G.C. (1990), "A Note on Localization Phenomena and Adaptive Finite-Element Analysis in Forming Processes", *Communications in Numerical Methods in Engineering*, Vol. 6, pp. 71-76.
- Zienkiewicz, O.C. and Morgan, K. (1983), *Finite Elements and Approximation*, John Wiley and Sons Ltd., England.
- Zienkiewicz, O.C. and Taylor, R.L. (1991), *The Finite Element Method, Vols. 1 and 2*, Mc Graw Hill, London.

