SELF-ADAPTING TIME INTEGRATION MANAGEMENT IN CRASH-WORTHINESS AND SHEET METAL FORMING COMPUTATIONS

L. NOELS, L. STAINIER and J.P. PONTHOT
University of Liège - LTAS-Thermomechanics - Bât. B52/3
Chemin des Chevreuils 1, B-4000 Liège 1, Belgium
Voice +32-4-3669310 - Fax +32-4-3669141
Email : jp.ponthot@ulg.ac.be (corresponding author)

Keywords: computer simulations, automatic time-stepping, contact-impact problems, plasticity, augmented Lagrangian method

Abstract

Variable step strategies are especially well suited to deal with problems characterized by high non-linearity and contact/impact, and resolved with an implicit scheme. Both phenomena are typical of dynamic simulations of contact-impact problems, as well as sheet metal forming. Constant step size strategies do not give a satisfactory answer for this kind of problems, since it is very difficult, if not impossible, for the user to find an appropriate time step that does not lead to divergence nor generate extremely costly computations. An automatic time stepping algorithm is proposed, which takes into account the recent history of accelerations in the deformable bodies under consideration. More precisely, the adaptation algorithm is based on estimators of the integration error of the differential dynamic balance equations. This allows for adaptation of the step size to capture correctly the transient phenomena, with characteristic times which can range from relatively long (after contact, or during sheet metal forming) to very short (during contact-impact), thus ensuring precision while keeping the computation cost to a minimum. Furthermore, we will see that this strategy can be used in explicit schemes. Additionally, the proposed algorithm automatically takes decisions regarding the necessity of updating the tangent matrix or stopping the iterations, further reducing the computational costs especially, when the Augmented Lagrangian method is used. As an illustration of the capabilities of this algorithm, several numerical simulations (shock absorber devices for vehicle crash-worthiness or sheet metal forming) problems will be presented. Other simulations pertaining to the sheet metal forming for vehicle structures will also be investigated, thus demonstrating the versatility, the capabilities and the efficiency of the proposed strategy.

Table of notations

b	body force per unit mass (N/m^3)
B	matrix of the derivatives of the FEM shape function
CO	counter limit for reducing time step size
CT	counter limit for augmenting time step size
C_T	tangent damping matrix (Ns/m)
e	integration error
$ERRO$	maximum error of last steps before reducing it
$ERRT$	maximum error of last steps before augmenting it
f	surface forces (N/m^2)
F_n^{cont}	force of contact at node n
F^{ext}	vector of external forces
F^{int}	vector of internal forces
K_T	tangent stiffness matrix
g_n	gap of node n
M	mass matrix
N	matrix of the FEM shape function
p	penalty of contacts
$PRCU$	tolerance of integration error
r	non dimensional residual of the equations of motion
R	residual vector
$RAPRES$	ratio between two successive non dimensional residues
RAT	ratio between two time step size
$RDOWN$	ratio between two time step size when divergence occurs
$REJL$	maximum integration error tolerated
S	Hessian matrix (N/m)
$S(t)$	current surface of the body (m^2)
S_0	initial surface of the body (m^2)
$TRHLD$	integration error limit below which time step size could be augmented
t	time (s)
$V(t)$	current volume of the body (m^3)
V_0	initial volume of the body (m^3)
$VALRF$	ratio between CPU time of an iteration with updating of the Hessian matrix and without updating
x	vector of nodal positions (m)
$\dot{x}$	vector of nodal velocities (m/s)
$\ddot{x}$	vector of nodal accelerations (m/s^2)
α_M	first free parameter balancing sampling time around $[t_n, t_{n+1}]$ for averaging inertia terms
α_F	second free parameter balancing sampling time around $[t_n, t_{n+1}]$ for averaging forces
β	first Newmark parameter
γ	second Newmark parameter
γ_s	security coefficient on time step size for an explicit scheme
Δt	time step size: $\Delta t = t_{n+1} - t_n$ (s)
Δt_{crit}	time step size limit to have a stable explicit scheme
δ	tolerance of non dimensional residual
ε	integration error of an one degree of freedom linear system
η	parameter to estimate new time step size
ρ	current mass density of the material (kg/m^3)
ρ_0	initial mass density of the material (kg/m^3)

σ	Cauchy stress tensor (N/m^2)
ω	pulsation of an one degree of freedom linear system $(1/s)$
Ω	non dimensional pulsation $(\omega \Delta t)$

1 Introduction

Non-linear dynamics problems integrated in time can be solved with two kind of time stepping algorithms: explicit or implicit. For an explicit algorithm, the elements of solution at time t_{n+1} depend only on the solution at time t_n, while for an implicit algorithm, they also implicitly depend on other elements of the solution at time t_{n+1} itself. The problem must then be solved in an iterative fashion. Stability (i.e. positive damping of initial perturbations) imposes different restrictions on those two families of algorithms and, with a proper choice of parameters, the time step size can be much larger for an implicit algorithm than for an explicit algorithm that is conditionally stable. The total number of time steps in an implicit scheme will thus generally be smaller. Then, even though the cost of a time step is higher, as a consequence of the need for computing and inverting a Hessian matrix, the total computation time for an implicit scheme is often more interesting than for an explicit scheme. In this context, for an implicit scheme, if the time step size is chosen too small, the calculation cost is very expensive, while if it is chosen too large, the integration is not accurate enough or the iterations diverge (when solving the balance equations). Therefore, time step size should be carefully evaluated. Since the problem evolves with time, the time step size should be continuously adapted to this evolution. An automatic time stepping algorithm is then the only solution to accurately solve the problem in a reasonably short computation time. Furthermore, for some problems computed with an explicit scheme, a guarantee of accuracy is needed. In such a problem the critical time step size (that ensure stability of the problem)is too large. Therefore, a estimation of the integration error can resolve this problems of convergence.

For an industrial problem that has a large number of degrees of freedom, the most expensive operation of an implicit code is the inversion of the Hessian Matrix. For non-linear problems, the Hessian matrix normally evolves every iteration, but the Newton-Raphson iterations can sometimes converge while using the old inverted matrix. Still, this inverted matrix must be regularly recomputed to avoid divergence. In a classical strategy, this inversion occurs at the beginning of each time step and for some iterations selected by the user. But if the Hessian matrix is not inverted for too many iterations, the problem diverges, while if the inversion occurs too frequently, the problem becomes too expensive. According to the evolution of the problem with time, an algorithm automatically selecting if the inverted Hessian matrix must be recalculated or not can significantly reduce the total computation cost. This is especially efficient with quasi-linear problems where Hessian matrix evolves only a little with iterations or when the augmented Lagrangian method is used to treat contact. In this last case, once the iterations has converged, Lagrangian multipliers are actualized and then the iterative process is restarted. For problems with strong non linearities, updating of the Hessian matrix is necessary at each iteration. But once the time step has converged, those non linearities (such as contact) do not evolve very much. Therefore, the Hessian matrix can be kept constant for some iterations after the first augmentation.

Assuming the inverse Hessian matrix is updated at an acceptable frequency, the Newton-Raphson iterations can still diverge. The time step is then rejected and the time step size is reduced. A problem is to determine when iterations diverge. Divergence can result from a negative Jacobian. In this case, divergence detection is trivial. But even when there is no negative Jacobian, convergence is not ensured. Indeed, the residual is never guaranteed to decrease. In this case, divergence detection is more difficult. Usually, a maximum number

of iterations is defined. If this number is too small, a time step can be rejected while the problem slowly converges. If this number is chosen too large, some iterations are needlessly computed when the divergence actually occurs. It is then interesting to determine if divergence occurs on the basis of the evolution of the residual. The maximum number of iterations is more difficult to be correctly determined when the inverted matrix is not computed at each iteration. Indeed, this number depends on how frequently the inverted matrix is computed.

This paper proposes an automatic time step control algorithm based on the measure of the integration error. This algorithm modifies the time step size only if durable physical changes occur in the problem evolution. Estimation of the error is made independent of the implicit scheme's parameters. Three estimators are compared. This algorithm is extended to explicit schemes. An algorithm choosing if the Hessian matrix is to be recomputed is also proposed. This determination is based on residual evolution with iterations. Finally, a divergence criterion based on this residual evolution is implemented. Academic and industrial numerical examples are then presented to illustrate these new algorithms.

2 Numerical integration of transient problems

In this section, the equations of motion and both implicit and explicit schemes of integration are rapidly explained. Next the penalty contact method and the augmented Lagrangian method are shortly reviewed.

2.1 Equations of motion

FEM (space) semi-discretization of the equations of motion of a nonlinear structure leads to the coupled set of second order nonlinear differential equations [1], [2], [3], [4], [5], [6]:

$$R = M\ddot{x} + F^{int}\left(x, \dot{x}\right) - F^{ext}\left(x, \dot{x}\right) = 0 \tag{1}$$

Where R is the residual vector, x the vector of nodal positions, $\dot{x}$ the vector of nodal velocities, $\ddot{x}$ the vector of nodal accelerations. M is the mass matrix, F^{int} the vector of internal forces resulting from body's deformation and F^{ext} the vector of external forces. Both vectors are non-linear in x and in $\dot{x}$ due to phenomena of contact, plastic deformations, geometrical non-linearity. The set of equations (1) is completed by two sets of given initial conditions at time zero:

$$x_0 = x\left(t = 0\right) \quad \dot{x}_0 = \dot{x}\left(t = 0\right) \tag{2}$$

Internal and external forces can be written:

$$F^{int}\left(x, \dot{x}\right) = \int_{V(t)} \left[B\right]^T \left\{f\right\} dV \tag{3}$$

$$F^{ext}\left(x, \dot{x}\right) = \int_{V(t)} \left[N\right]^T \left\{b\right\} dV + \int_{S(t)} \left[N\right]^T \left\{f\right\} dS \tag{4}$$

Note that expression (4) collects all types of loading (applied through local or distributed actions, in a follow-up way or not, reactions to imposed displacements and contact situations) and that the consistent mass matrix reads

$$M = \int_{V(t)} \rho \left[N\right]^T \left[N\right] dV = \int_{V_0(t)} \rho_0 \left[N\right]^T \left[N\right] dV_0 \tag{5}$$

2.2 Implicit schemes

Implicit scheme are classically designed for vibrations and low speed dynamics of structures. For an implicit scheme, the elements of the solution at time t_{n+1}, implicitly depend on other elements of the solution at time t_{n+1} itself. The problem must then be solved in an iterative fashion. Stability (i.e. positive damping of initial perturbations) imposes different restrictions on this family of algorithms and, with a proper choice of parameters. The time step size does not need to be lower than a limit to have a stable time step since the scheme is conditionally stable.

2.2.1 Implicit schemes: The generalized-α trapezoidal scheme

The most general scheme for implicit integration of (1) is a generalized trapezoidal scheme [1], [2], [7] where updating of positions and velocities is based on "averaged" accelerations stemming from associated values between t_n and t_{n+1}. It reads for instance

$$\dot{x}_{n+1} = \dot{x}_n + (1-\gamma)\,\Delta t\ddot{x}_n + \gamma\Delta t\ddot{x}_{n+1} \tag{6}$$

$$x_{n+1} = x_n + \Delta t\dot{x}_n + \left(\frac{1}{2}-\beta\right)\Delta t^2\ddot{x}_n + \beta\Delta t^2\ddot{x}_{n+1} \tag{7}$$

or equivalently

$$\ddot{x}_{n+1} = \frac{1}{\beta\Delta t^2}\left[x_{n+1} - x_n + \Delta t\dot{x}_n - \left(\frac{1}{2}-\beta\right)\Delta t^2\ddot{x}_n\right] \tag{8}$$

$$\dot{x}_{n+1} = \frac{\gamma}{\beta\Delta t}\left[x_{n+1} - x_n + \left(\frac{\beta}{\gamma}-1\right)\Delta t\dot{x}_n + \left(\frac{\beta}{\gamma}-\frac{1}{2}\right)\Delta t^2\ddot{x}_n\right] \tag{9}$$

The discretized equations of motion (1) can be rewritten under the form proposed by Chung and Hulbert [7]:

$$\begin{aligned} R_{n,n+1} \quad = \quad & \tfrac{1-\alpha_M}{1-\alpha_F}M\ddot{x}_{n+1} + \tfrac{\alpha_M}{1-\alpha_F}M\ddot{x}_n + \left(F^{int}_{n+1} - F^{ext}_{n+1}\right) \\ & + \tfrac{\alpha_F}{1-\alpha_F}\left(F^{int}_n - F^{ext}_n\right) = 0 \end{aligned} \tag{10}$$

Particular choices of parameters lead to well-known [1], [2], [7] schemes such as

- $\alpha_M = \alpha_F = 0$ for Newmark scheme
- $\alpha_M = 0$ for Hilber-Hughes-Taylor scheme
- $\alpha_F = 0$ for Wood-Bossak-Zienkiewicz scheme

Unconditional stability and second order accuracy of the scheme, for linear problems [7], require that

$$\begin{array}{lcl} \gamma & \geq & \frac{1}{2} - \alpha_M + \alpha_F \\ \alpha_M & \leq & \frac{1}{2} \\ \beta & \geq & \frac{1}{4}\left(1 + \alpha_F - \alpha_M\right)^2 \end{array} \tag{11}$$

Associated rules for Newmark scheme are

$$\begin{array}{lcl} \gamma & \geq & \frac{1}{2} \\ \beta & \geq & \frac{1}{4}\left(\gamma + \frac{1}{2}\right)^2 \end{array} \tag{12}$$

It's worth pointing out that though classical schemes require $0 < \alpha_F < 1/2$ (i.e. sampling the force in the second half of $[t_n, , t_{n+1}]$), here, no such rule is followed for α_M, the sampling parameter for the inertia terms: it might be negative for instance, thus leading to an extrapolation at t_{n+1} instead of an interpolation.
Iterative solution of the nonlinear system (10) first requires the elimination of acceleration and velocity at time t_{n+1} with the help of (8) and (9) and, secondly, the writing of the Hessian matrix of the system, i.e.

$$S = \left[\frac{1}{\beta \Delta t^2} \left(\frac{1-\alpha_M}{1-\alpha_F} \right) M + \frac{\gamma}{\beta \Delta t} C_T + K_T \right] \tag{13}$$

where K_T, C_T are respectively the tangent stiffness and damping matrices defined by:

$$K_T = \frac{\partial}{\partial x} \left(F^{int} - F^{ext} \right) \tag{14}$$

$$C_T = \frac{\partial}{\partial \dot{x}} \left(F^{int} - F^{ext} \right) \tag{15}$$

The residual after iteration number i is defined by:

$$\begin{aligned} R \quad = \quad & \tfrac{1-\alpha_M}{1-\alpha_F} M \ddot{x}^i_{n+1} + \tfrac{\alpha_M}{1-\alpha_F} M \ddot{x}_n + \\ & \left[F^{int}_{n+1} \left(x^i_{n+1}, \dot{x}^i_{n+1} \right) - F^{ext}_{n+1} \left(x^i_{n+1}, \dot{x}^i_{n+1} \right) \right] \\ & + \tfrac{\alpha_F}{1-\alpha_F} \left[F^{int}_n - F^{ext}_n \right] \end{aligned} \tag{16}$$

Using equations (13) to (16) and a Newton-Raphson technique, the iterative solution of system (8), (9) and (10) can be written as:

$$S \Delta x = -R \tag{17}$$

Iterations stop when the non-dimensional residual r becomes lower than the accuracy tolerance δ that is defined by the user. Therefore, the following relation is verified:

$$r = \frac{\|R\|}{\|F^{int}\| + \|F^{ext}\|} \leq \delta \tag{18}$$

2.2.2 Implicit schemes: The generalized-θ mid-point scheme (GMP)

An alternative to the previous scheme is a generalized midpoint scheme with constant acceleration over the time step [3], [4], [5]. In this case, the equations of motion (1) are solved at the sampling time: $t_{n+\theta} = t_n + \theta(t_{n+1} - t_n)$ with $\theta > 0$, i.e.

$$R_\theta = M \ddot{x}_{n+\theta} + F^{int} \left(x_{n+\theta}, \dot{x}_{n+\theta} \right) - F^{ext} \left(x_{n+\theta}, \dot{x}_{n+\theta} \right) = 0 \tag{19}$$

where

$$\ddot{x}_{n+\theta} = \frac{2}{(\theta \Delta t)^2} \left[x_{n+\theta} - x_n - \theta \Delta t \dot{x}_n \right] \tag{20}$$

$$\dot{x}_{n+\theta} = \frac{2}{\theta\Delta t}\left[x_{n+\theta} - x_n - \frac{\theta\Delta t}{2}\dot{x}_n\right] \tag{21}$$

Iterative solution of the nonlinear system (19) requires the evaluation of the Hessian matrix of the system given by:

$$S = \left[\frac{2}{(\theta\Delta t)^2}M + \frac{2}{\theta\Delta t}C_T + K_T\right] \tag{22}$$

Let us stress some features of the θ-GMP scheme with respect to α-family:

- All forces, even contact ones, are exactly estimated at time $t_{n+\theta}$ instead of being averaged between the values at t_n and t_{n+1} as in (10).
- The present scheme is $\ddot{x}_n$-independent, thus yielding the final acceleration as a post-treatment result given by (19)

$$\ddot{x}_{n+1} = \frac{\dot{x}_{n+1} - \dot{x}_n}{\Delta t} = \ddot{x}_{n+\theta} \tag{23}$$

- Since, for θ=1, it corresponds to the Newmark scheme with γ=1 and β=0.5, we can state that it is conditionally stable and exhibits strong numerical damping. However, this scheme has proved to be very efficient provided θ is taken larger than unity, i.e. choosing a sampling point out of the range $[t_n, t_{n+1}]$, thus producing some backward evaluation of the final status of the system at the end of the time step, with respect to the sampling point for residual evaluation.

2.3 Explicit scheme

This is the most advocated scheme [1], [2] for integrating (1) in case of wave propagation and impact problems, i.e. high speed dynamics. For an explicit algorithm, the elements of solution at time t_{n+1} depend only on the solution at time t_n. Therefore, the resolution does not need to be iterative. Stability (i.e. positive damping of initial perturbations) imposes that the time step size be lower than a limit. The scheme is conditionally stable. It reads for a time step $\Delta t = t_{n+1} - t_n$:

$$\dot{x}_{n+\frac{1}{2}} = \dot{x}_{n-\frac{1}{2}} + \Delta t\ddot{x}_n \tag{24}$$

$$x_{n+1} = x_n + \Delta t\dot{x}_{n+\frac{1}{2}} \tag{25}$$

$$\ddot{x}_{n+1} = M^{-1}\left(F^{ext}_{n+1} - F^{int}_{n+1}\right) \tag{26}$$

Stability imposes the time step size to be lower than a critical time step [1] that depends on the maximum pulsation ω of the body. A security coefficient γ_s is introduced to take into account the fact that problem is not linear (theory to estimate Δt_{crit} is based on a linear theory) :

$$\Delta t = \gamma_s \Delta t_{crit} = \gamma_s \frac{2}{\omega_{max}} \tag{27}$$

2.4 Treatment of contact

The penalty method is used to treat contacts. When contact is detected between two bodies or between a body and a rigid tool, forces of contact are introduced into the external forces. These forces are proportional to the penetration of the node n (gap: g_n) and are normal (and tangential if there is sticking). Defining p the penalty, forces of contact at node n are rewritten [3], [8]:

$$\begin{array}{lll} F_n^{cont} & = -pg_n & \text{if } g_n < 0 \\ & = 0 & \text{if } g_n \geq 0 \end{array} \tag{28}$$

With an implicit scheme, the Augmented Lagrangian method can be used to ensure accuracy [8]. Before the first augmentation, the contact forces are computed as in (28) until the iterations converge and relation (18) is verified. But at this point no accuracy on the penetration is ensured. Therefore a solution that constrain g_n below a limit can be researched. Let us define superscript $i = 0$ to refer to the solution before the first augmentation and superscript $i > 0$ to refer to the solution after the augmentation i. The contact forces are stoked in the Lagrangian vector λ after the solution has converged (i.e. 18 is verified). It comes therefore for node n :

$$\lambda_n^i = F_n^{cont,i} \tag{29}$$

Then iterations restart with the new forces of contact:

$$\begin{array}{lll} {F_n^{cont}}^{i+1} & = \lambda_n^i - pg_n & \text{if } g_n < \frac{\lambda_n^i}{p} \\ & = 0 & \text{if } g_n \geq \frac{\lambda_n^i}{p} \end{array} \tag{30}$$

and stop when relation (18) is verified. Augmentations occurs until the gap for all the contacts is lower than a given limit.

3 Parameters control for implicit schemes

First the time step size control problem is studied. Next an algorithm deciding if Hessian matrix need to be updated is proposed. The problem of determining convergence is then exposed. Finally numerical examples are exposed to validate the proposed algorithms.

3.1 Automatic time step size control

A relatively simple method proposed by Ponthot [3] aims at an optimal number of iterations. If the number of iterations exceeds this optimal number, new time step size is reduced, while, if the number of iterations is lower than the optimal number, time step size is augmented. Givoli and Henisberg [9] propose to modify the time step size to keep the displacements difference between two successive times lower than a given limit. Géradin [10] (Figure 1) estimates the integration error from the accelerations and the inertial forces difference between two successive times multiplied by the square of time step size. This error is divided by a constant depending on the initial positions and by a constant that is the average error for a one-degree-of-freedom linear system (defined as in section 3.1.1), resulting the non-dimensional integration error ϵ. The error must be lower than a given tolerance ($PRCU$). If the error is higher, the time step is rejected and its size divided by two. If the error is lower than the tolerance but higher than half the tolerance, the time step is divided by the ratio between the error and the half tolerance to the power one third. If the error

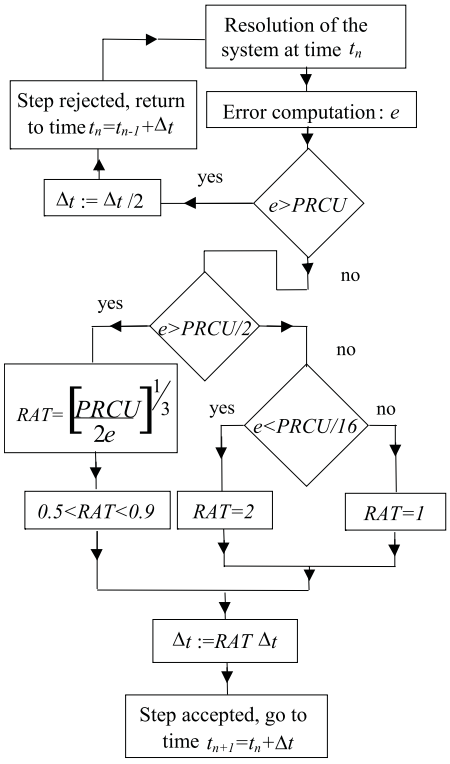

Figure 1: *Time step size control proposed by Géradin [10]*

is lower than the tolerance divided by sixteen, the time step size is multiplied by two. For Cassano and Cardona [11], the time step control is the same than for Géradin, but the error is calculated only from the accelerations difference and is not divided by a constant depending on the initial positions but by a term that evolves with positions. Hulbert and Jang [12] (Figure 2) estimate the error from the accelerations difference multiplied by the square of time step size. This error is divided by a term that depends on the positions difference. Time step control is characterized by two tolerances ($TOL1$ and $TOL2$) and by a counter of maximal index $LCOUNT$. If the error is higher than $TOL2$, then the step is rejected and time step size is reduced. If error is lower than $TOL2$ and higher than $TOL1$, the time step is accepted and time step size is kept constant. If the error is lower than $TOL1$ then time step is accepted. If it occurs successively $LCOUNT$ times, then the time step size is augmented. The counter is introduced to avoid undesirable changes in time step size due to the periodic nature of the local error. Dutta and Ramakrishnan [13] also calculate the error from the accelerations difference multiplied by the square of the time step size. It is made non-dimensional by dividing it by the maximum norm of the positions vector, for the previous time step. The time interval is divided in sub-domains. In each sub-domain there are a certain number of time steps of constant size. Once the time marching scheme has gone through a whole sub-domain, an average error is calculated. A time step size for the next sub-domain is then computed from this average error.

In this paper, the automatic control scheme is based on the algorithm proposed by Géradin [10]. Nevertheless, due to the non-linear characteristics of the problems we are interested in, we will assume that the time step size reacts only on evolution in physical mode and not on numerical mode. Changes in time step size will also occur only if the new time step size

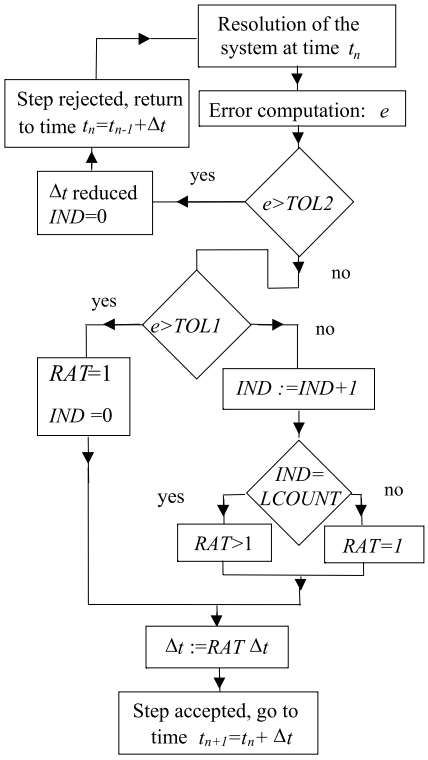

Figure 2: *Time step size control proposed by Hulbert and Jang [12]*

can be kept constant for several steps. On the other hand, the error estimator based on the inertial forces difference (proposed by Géradin [10] and established for a linear theory) and the error estimator based on the acceleration difference (established for linear and non-linear problem) are compared. It will appear that for non-linear problems a linear theory is not adequate.

3.1.1 Error estimator

Estimation error comes from the truncation error of equations (6) and (7) or equations (20, 21). Indeed, the error is of the third order: $e = O\left(\frac{1}{6}\Delta t^3 \dddot{x}\right) \simeq O\left(\frac{1}{6}\Delta t^2 \ddot{x}\right)$. Therefore, we can write:

$$e = \frac{\Delta t^2}{6} \|\Delta \ddot{x}\| \tag{31}$$

First, this expression must be available for each problem. Then it is rendered non-dimensional (x_0 is the vector of the initial positions):

$$e = \frac{\Delta t^2}{6\,\|x_0\|} \|\Delta \ddot{x}\| \tag{32}$$

To ensure the error estimator can be used for each implicit scheme (the generalized-α trapezoidal scheme or the generalized-θ mid-point scheme), and for each parameter (α_F,

α_M, β, γ or θ) without modifying tolerance on the error (see section 3.1.2), expression (32) is divided by a reference. This reference is the average error (on a period) for a one-degree of freedom linear oscillator. Assuming time step size is constant, and pulsation is ω, we can define the non-dimensional pulsation as $\Omega = \omega \Delta t$. For such a problem, equations (8), (9) et (10) can be rewritten:

$$\left\{ \begin{array}{l} x_{n+1} = x_n + \Delta t \dot{x}_n + \left(\frac{1}{2} - \beta\right) \Delta t^2 \ddot{x}_n + \beta \Delta t^2 \ddot{x}_{n+1} \\ \dot{x}_{n+1} = \dot{x}_n + (1-\gamma)\, \Delta t \ddot{x}_n + \gamma \Delta t \ddot{x}_{n+1} \\ (1-\alpha_M)\, \ddot{x}_{n+1} + \alpha_M \ddot{x}_n + (1-\alpha_F)\, \omega^2 x_{n+1} + \alpha_F \omega^2 x_n = 0 \end{array} \right. \tag{33}$$

or:

$$\begin{pmatrix} x_{n+1} \\ \Delta t \dot{x}_{n+1} \\ \Delta t^2 \ddot{x}_{n+1} \end{pmatrix} = A(\Omega) \begin{pmatrix} x_n \\ \Delta t \dot{x}_n \\ \Delta t^2 \ddot{x}_n \end{pmatrix} \tag{34}$$

with:

$$A(\Omega) = \frac{1}{D(\Omega)} \begin{pmatrix} D(\Omega) - \Omega^2 \beta & 1 - \alpha_M & \frac{1 - \alpha_M - 2\beta}{2} \\ -\gamma \Omega^2 & D(\Omega) - \Omega^2 \gamma (1 - \alpha_F) & D(\Omega) - \gamma \left[1 + \frac{\Omega^2}{2}(1 - \alpha_F)\right] \\ -\Omega^2 & (\alpha_F - 1)\, \Omega^2 & -\alpha_M + (1 - \alpha_F)\left(\beta - \frac{1}{2}\right) \Omega^2 \end{pmatrix} \tag{35}$$

where:

$$D(\Omega) = 1 - \alpha_M + (1 - \alpha_F)\, \Omega^2 \beta \tag{36}$$

and finally:

$$\begin{pmatrix} \Delta x \\ \Delta t \Delta \dot{x} \\ \Delta t^2 \Delta \ddot{x} \end{pmatrix} = [A(\Omega) - I] \begin{pmatrix} x_0 \cos(\omega t) \\ -\Omega x_0 \sin(\omega t) \\ -\Omega^2 x_0 \cos(\omega t) \end{pmatrix} \tag{37}$$

Therefore the error (32) for the one-degree of freedom linear oscillator can be expressed as:

$$e = \frac{\Delta t^2}{6 \|x_0\|} \|\Delta \ddot{x}\| = \frac{(1 - \alpha_F)\, \Omega^2 \left\| \sin(\omega t) + \frac{\Omega}{2} \cos(\omega t) \right\|}{6 \left[1 - \alpha_M + (1 - \alpha_F)\, \Omega^2 \beta\right]} \tag{38}$$

The reference is the average error for a period. It can be noted ε:

$$\varepsilon = \frac{\omega}{2\pi} \int_{t=0}^{t=\frac{2\pi}{\omega}} e\, dt \tag{39}$$

Using (38), expression (39) is explicitly calculated:

$$\varepsilon(\Omega) = \frac{(1 - \alpha_F)\, \Omega^3 \sqrt{1 + \frac{\Omega^2}{4}}}{3\pi \left[1 - \alpha_M + (1 - \alpha_F)\, \Omega^2 \beta\right]} \tag{40}$$

If α_M=0, then we recover the expression calculated by Géradin [10] for HHT implicit scheme. Expression (40) is established for the generalized-α trapezoidal scheme, while for the generalized-θ mid-point scheme, system (33) is replaced by:

$$\begin{cases} x_{n+\theta} = x_n + \theta\Delta t\dot{x}_n + \frac{\theta^2\Delta t^2}{2}\ddot{x}_{n+\theta} \\ \dot{x}_{n+\theta} = \dot{x}_n + \theta\Delta t\ddot{x}_{n+\theta} \\ \ddot{x}_{n+\theta} + \omega^2 x_{n+\theta} = 0 \end{cases} \tag{41}$$

with:

$$\begin{cases} x_{n+1} = x_n + \Delta t\dot{x}_n + \frac{\Delta t^2}{2}\ddot{x}_{n+\theta} \\ \dot{x}_{n+1} = \dot{x}_n + \Delta t\ddot{x}_{n+\theta} \\ \ddot{x}_{n+1} = \ddot{x}_{n+\theta} \end{cases} \tag{42}$$

Therefore matrix $A(\Omega)$ in expression (37) becomes:

$$A(\Omega) = \frac{2}{2+\theta^2\Omega^2}\begin{pmatrix} 1+\frac{\Omega^2}{2}\left(\theta^2-1\right) & 1+\frac{\Omega^2}{2}\left(\theta^2-\theta\right) & 0 \\ -\gamma\Omega^2 & 1+\frac{\Omega^2}{2}\left(\theta^2-2\theta\right) & 0 \\ -\Omega^2\theta^2 & -\Omega^2\theta & 0 \end{pmatrix} \tag{43}$$

Finally, the reference error (40) is rewritten:

$$\varepsilon(\Omega) = \frac{\Omega^2\sqrt{\left[\theta^2\Omega^2 + 2\left(1-\theta^2\right)\right]^2 + 4\theta^2\Omega^2}}{3\pi\left(2+\theta^2\Omega^2\right)} \tag{44}$$

Error (32) is then divided by ε (expression 40 or 44) to have an expression independent of the used scheme. However, Ω need to be known to estimate ε. For the one-degree of freedom linear oscillator, ten time steps in a period gives a good accuracy with a relatively low computation cost. Therefore with the non-dimensional pulsation, which corresponds to a $0.1Hz$ frequency, given by $\Omega_k = 0.6$, we define, using (32):

$$e_1 = \frac{\Delta t^2}{6\varepsilon(\Omega_k)\,\|x_0\|}\,\|\Delta\ddot{x}\| \tag{45}$$

For linear systems, Géradin demonstrated [10] that the error can be evaluated as expression (46). This error filters high frequency modes (as numerical modes). However, for non-linear systems, no advantage is gained (see numerical examples in section 3.4) in replacing the acceleration difference by a term depending on the accelerations and the inertial forces difference as in (46), yielding:

$$e_2 = \frac{\Delta t^2}{6\varepsilon(\Omega_k)\left[x_0^T M x_0\right]^{\frac{1}{2}}}\left[\Delta\ddot{x}^T\Delta\left(M\ddot{x}\right)\right]^{\frac{1}{2}} \tag{46}$$

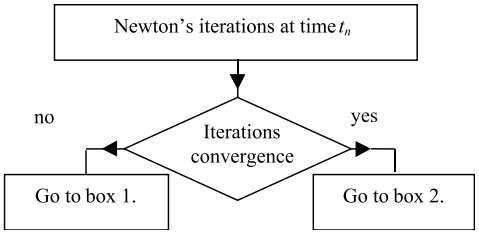

Figure 3: *Iterations convergence test*

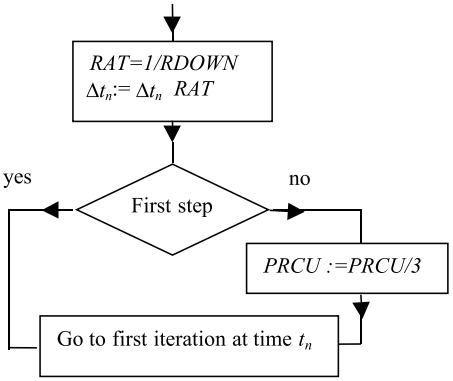

Figure 4: *Description of box 1, step size control when iterations diverge*

This last expression is similar to expression (45) but inertial force difference is used instead of acceleration difference. Another possibility (Cassano and Cardona [11]) to evaluate the error is to keep the maximum acceleration difference and not the vector norm. We define e_3, with *ndof* the number of degrees of freedom:

$$e_3 = \frac{\Delta t^2}{6\varepsilon\left(\Omega_k\right) \max\limits_{j=1,ndof} \left(x_0\right)_j} \max_{i=1,ndof} \left(\Delta\ddot{x}\right)_i \tag{47}$$

In this paper those three error indicators are compared on academic cases.

3.1.2 Time step size control

The computed error must be of the order of a user-defined tolerance that is noted *PRCU*. A value of this tolerance leading to a good accuracy-price ratio is typically 1E-4. A low *PRCU* gives a good accuracy but a longer computation time. A higher *PRCU* gives a shorter computation time but a lower accuracy. If *PRCU* is too high, the time step size can result in an error lower than *PRCU* but can be not small enough to allow for iterations to converge.

Therefore, if a problem of convergence appears (Figure 3),the algorithm goes to box 1 and reduces *PRCU* (Figure 4). More, time step size is divided by *RDOWN* that is initialized at 3 by default. After some time step without convergence problems, tolerance *PRCU* can be augmented. This number of time step is large enough to avoid oscillation in *PRCU* value. It could depend on divergence occurrences.

If the iterations converge, the algorithm goes to box 2 and tries to adjust the time step size to have an error equal to one half of *PRCU*. There exist three possibilities (Figure 5):

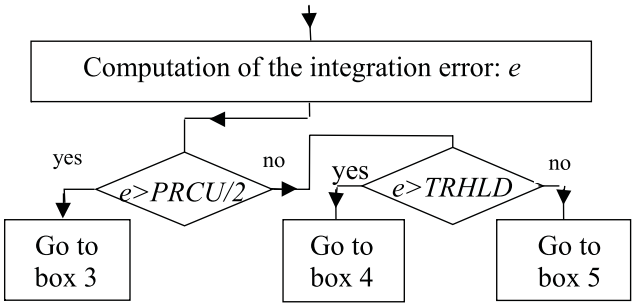

Figure 5: *Description of box 2, step size control when iterations converge*

- The error is larger than *PRCU/2*, the algorithm goes to box 3. It is considered to be too high. To ensure a good accuracy, next time step size must be reduced.
- The error is in the interval *[TRHLD, PRCU/2]*, the algorithm goes to box 4. The time step size ensures a good accuracy with a relatively low computation cost. Time step size is kept constant.
- The error is smaller than a limit *TRHLD*, the algorithm goes to box 5. It is considered to be too small. To ensure a reduced computation cost, time step size must be augmented.

Let us first examine the problem of too high an error (Figure 6). The next time step size must therefore be reduced. But to avoid needless changes of time step, we will make sure that the variation of the integration error is due to a durable and physical evolution in the problem. The time step is then reduced only if there are a number (CO) of successive time steps for which integration error is larger than *PRCU/2*. This number CO can be taken equal to three. The factor by which the time step size is reduced depends on the maximum error ($ERRO$) of CO successive time steps. Géradin [10] demonstrates that for a linear one-degree-of-freedom system, the factor by which the time step size needs to be multiplied to reduce the error from e to *PRCU/2* can be written:

$$RAT = \left[\frac{PRCU}{2e}\right]^{\frac{1}{\eta}}, \eta \in [2,3] \tag{48}$$

For non-linear systems η can be out of this interval. To ensure that the time step size is sufficiently reduced, h is taken smaller than two. The factor that finally multiplies the time step size is $RAT = \left[\frac{PRCU}{2ERRO}\right]^{2/3}$. But if there is a rapid change in the physical problem (impact ...), the time step is not immediately adapted. Therefore, if the error e is larger than $PRCU$, the time step size is immediately multiplied by $RAT = \left[\frac{PRCU}{2e}\right]^{2/3}$. If the error e is larger than a limit $REJL$, the time step is rejected and its new value is size is multiplied by $RAT = \left[\frac{PRCU}{2e}\right]^{2/3}$. $REJL$ can be taken equal to $\frac{3PRCU}{2}$.

If the error is smaller than $\frac{PRCU}{2}$ and higher than $TRHLD$, the time step is kept constant (Figure 7). Typical values for $TRHLD$ are discussed at next paragraph.

Now, let us examine the problem of too small an error (Figure 8). The time step size could be augmented without degrading the solution. To avoid needless time step size changes, another counter is introduced. If CT successive time steps give an error lower then the limit $TRHLD$, the time step size is then augmented. $ERRT$ is the maximal error of those CT steps. To ensure the time step size is not augmented too much, η from equation (48) is taken larger

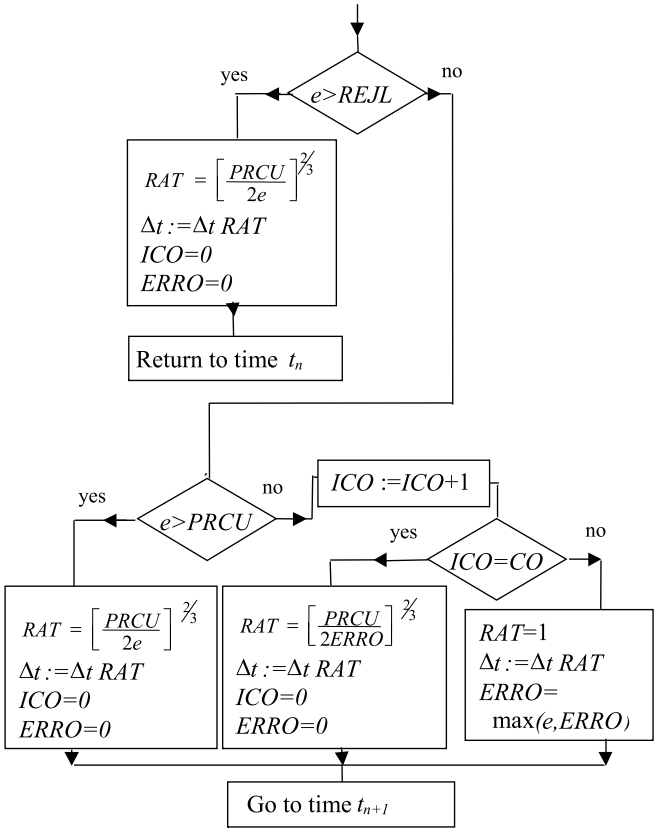

Figure 6: *Description of box 3, step size control when error is too large*

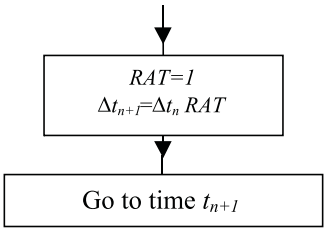

Figure 7: *Description of box 4, step size control when error is correct*

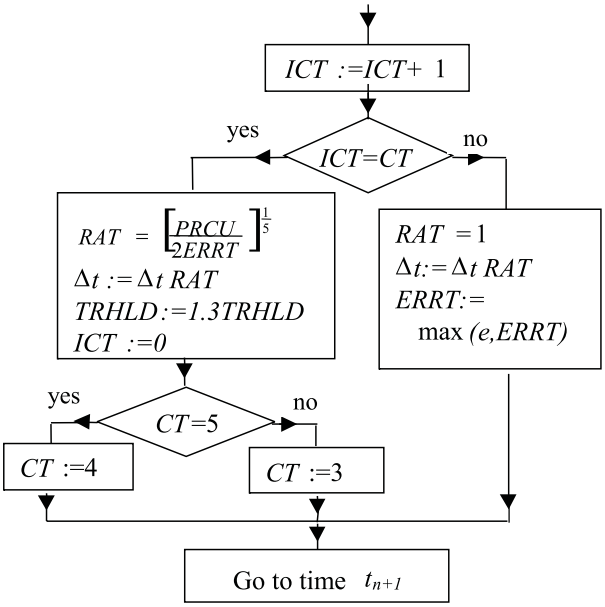

Figure 8: *Description of box 5, step size control when error is too small*

than 3. The factor multiplying the time step size is finally $RAT = \left[\frac{PRCU}{2ERRT}\right]^{\frac{1}{5}}$. A problem due to the introduction of a counter occurs when the solution becomes smoother (external forces diminish ...). Indeed, $TRHLD$ must be taken small (e.g. $\frac{PRCU}{16}$) and CT relatively large (e.g. 5) to ensure a good accuracy. In those conditions, time step size augments slowly. To reduce the computation cost, $TRHLD$ can be increased and CT can be decreased when the time step size is augmented. $TRHLD$ can be multiplied by 1.3 while CT is reduced to 4 first and to 2 next. Once a time step size is reduced, $TRHLD$ and CT are set back to their respective initial values $\frac{PRCU}{16}$ and 5. In some problems (translation at constant velocities), the error becomes nil. To avoid a division by zero, $ERRT$ is limited by $TRHLD\frac{PRCU}{10}$.

To complete boxes 1 to 5, let us note that: parameters ICO and $ERRO$ are re-initialized to their initial value if the scheme goes in box 1, 4 or 5 and parameters ICT and $ERRT$ if the scheme goes in box 1, 3 or 4.

3.2 Selective updating of the inverse Hessian matrix

For non-linear problems, if the Hessian matrix is not recomputed and inverted, the convergence of Newton-Raphson iterations is slower than if the Hessian matrix were recomputed and inverted at each iteration. For some step, divergence could also occur. Therefore, the criterion must consider two facts:

- Convergence of the iterations must be ensured.

- Not updating the Hessian matrix must reduce the total computation cost. Indeed, a problem with a small number of degrees of freedom and with strong non-linearities can converge in a few iterations when the Hessian matrix is updated at each iteration, but converge with more iterations when the Hessian matrix is not updated. When the number of degrees of freedom is reduced, an iteration without recalculation is not much less expensive. The total cost is then reduced when the Hessian matrix is often recalculated. On the other hand, if the problem has a large number of degrees of freedom and only a few non-linear elements, not updating the Hessian matrix can then reduce the computation cost.

3.2.1 The proposed algorithm

The evolution of the non-dimensional residual r (18) could indicate if the problem converges or not. While r decreases, iterations converge even if the Hessian matrix is not recalculated and not inverted. An indication of how it could be interesting not to recalculate the Hessian matrix is the ratio $VALRF$ between the time needed for an iteration with re-calculation and an iteration without re-calculation. This ratio indicates how much an iteration without re-calculation could advantageously replace an iteration with re-calculation. This value $VALRF$ is an integer in the range 2 to 9.

The proposed algorithm is the following:

- The Hessian matrix is recalculated at the first iteration if the time step size has changed. Indeed, S depends on Δt (13 or 22). Therefore, a modification of the time step size needs to the Hessian matrix to be updated to avoid divergence.
- If the number of the iterations is greater than $VALRF$, the next iteration is made with re-calculation of the Hessian matrix. Then, iterations occur without re-calculation only if it is less expensive.
- If the number of iterations is lower than $VALRF$, the Hessian matrix is recalculated only if the non-dimensional residual r has not been reduced by a ratio chosen equal to $RAPRES = \frac{VALRF}{10} \in [0.2, 0.85]$.
- If the non-dimensional residual has not been divided by $RAPRES$, the next iteration then needs updating of the Hessian matrix. But, if the residue has augmented, this iteration does not take as initial values $(x,\ \dot{x},\ \ddot{x})$ the values at the end of the previous iteration, but the value at the end of the last iteration which has converged (i.e. the penultimate iteration). Some divergences of the iterations are then avoided.

This algorithm avoids some needless re-calculations and inversions of the Hessian matrix. For strongly non-linear problems with a small number of degrees of freedom, this algorithm updates at most of the iteration and is at worst a little expensive (due to some few expansive rejected iterations without updating) than an algorithm with updating at each iteration. For problems with more degrees of freedom, this algorithm is less expensive than an algorithm where the user decides, more or less arbitrary, of the number of the iterations with re-calculation [14]. In fact, this algorithm allows a lot of iterations without re-calculation when possible, and recalculates frequently the Hessian matrix when needed.

3.2.2 Extension to the augmented Lagrangian method

When using the augmented Lagrangian method (see section 2.4), the previously exposed algorithm of selection of Hessian matrix updating is also enabled for each iteration of each augmentation. Nevertheless, a modification could be included. As seen in previous section, for a strong non-linear (a lot of (auto)contacts...) problem with a small number of degree

of freedom, the proposed algorithm updates the Hessian matrix at each iteration which is a little more expensive (due to some cheap rejected iterations without updating) than directly deciding of updating at each iterations. But this view becomes false after the first augmentation. Indeed after the iterations have converged before the first augmentation, the contact configuration for this time step won't change much for the next augmentations. Assuming that the most severe non-linearities are detection of new contacts or loss of old contacts and that this configuration does not evolve much, we can consider the problem as quasi-linear for iterations after the first augmentation. Therefore, for this kind of problem, the developed algorithm becomes efficient after the first augmentation.

3.3 Criterion of divergence

Two problems of divergence can occur. The first case occurs when an element has a negative Jacobian. This kind of divergence is easy to detect. A more difficult problem is to detect divergence when all Jacobians are positive, but when the residual evolution in the Newton-Raphson iterations does not lead to a residual lower than the defined tolerance. Usually, the user specifies a maximum number of iterations. If upon reaching this number, the non-dimensional residual r is not lower than the tolerance d, the time step is rejected and the time step size is divided. But when the residual r decreases slowly, the maximum number of iteration is exceeded before r becomes lower than δ (see relation 18). On the other hand, the process can diverge after a few iterations. More iterations are then needless. Finally, if we accept the problem to be solved without re-calculation of the Hessian matrix, the number of iterations is higher than when frequent re-calculations occur. Considering that the residue can presents oscillations, we consider that divergence occurs if the average value increases. A criterion that consider the last four iterations is used. The last residue is noted r_n, the previous one r_{n-1}, the penultimate one r_{n-2} and the ante-penultimate one r_{n-3}. Therefore, we consider that there is divergence if $r_n > r_{n-2}$ and $r_{n-1} > r_{n-3}$. Moreover, to avoid a risk of oscillation, we consider that divergence occurs if the non-dimensional residual has not been divided by two after 5 successive iterations with updating of the Hessian matrix. Several iterations need to be considered, because when divergence occurs, the non-dimensional residual usually presents some oscillations.

3.4 Numerical examples

Numerical examples are studied. The time step size control scheme developed is compared with the one proposed by Ponthot [3] that is described in section 3.1. Three error estimators are also compared for the proposed algorithm. The first one is defined by expression (45), the second one by expression (46) and the third one by expression (47). The problems considered are solved within the formalism of large deformations and displacements. All problems were computed in the research code METAFOR [3] in which the automatic time step size control algorithm has been implemented. Criterion of automatic updating and of divergence are also introduced.

3.4.1 Case 1: Contact of an elastic bar

An elastic bar [15] in plane strain (properties in Table 1) with an initial velocity (Figure 9) of -5 m/s (minus sign comes from the orientation of the x-axis), enters into contact with a rigid matrix initially at a distance of 0.25 mm. Due to a Poisson coefficient equal to zero, and due to vertical displacements fixed to zero, the problem is uni-dimensional. An analytic solution of this problem is known. In the interval [0 μs, 50 μs], the bar is in translation at constant velocity to wards the wall. Contact occurs at 50 μs and the velocity of the left edge becomes equal to zero. The celerity of the elastic wave, generated by the impact in

Height	$d = 40mm$
Length	$l = 247.65mm$
Initial distance to matrix	$d_i = 0.25mm$
Density	$\rho = 7895kg/m^3$
Young modulus	$E = 206.84E9N/m^2$
Poisson coefficient	$\nu = 0.0$
Initial velocity	$\dot{x}_0 = 5m/s$
Number of elements	20

Table 1: *Properties of elastic bar*

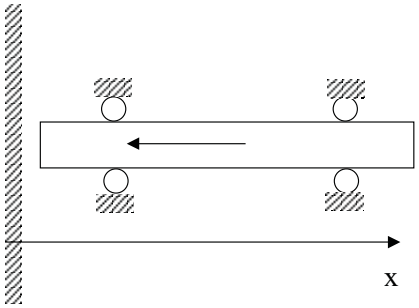

Figure 9: *Model of contact of an elastic bar*

the bar is $\sqrt{\frac{E}{\rho}} = 5120\ m/s$. It takes 100 μs for the wave to go from the left edge to the right edge and back from the right edge to the left edge. Thus the velocity of left edge is equal to zero during interval $[50\ \mu s, 150\ \mu s]$ and becomes equal (due to conservative laws of an elastic problem) to 5 m/s after 150 μs. This problem is resolved with the proposed time step control algorithm. Error indicators employed are successively those defined by relation (45), relation (46) and relation (47), respectively denoted e_1, e_2 and e_3. Due to the difficulty for this kind of problem to be integrated with the generalized-α trapezoidal scheme, a tolerance $PRCU$=10^{-4} is used for the numerical examples (if not stated otherwise). Finally, the problem is also solved with Ponthot's [3] method described in section 3.1. For the same reason that with the new algorithm, the optimal number of iteration is taken equal to 2, but for other problems it is taken equal to 4 (if not stated otherwise). This solution is called *opti*. The problem is solved with the generalized-α trapezoidal scheme ($\alpha_M = -0.997$ and other parameters automatically computed to have a stable scheme, i.e. $\alpha_F = 0.05$, $\gamma = 1.997$ and $\beta = 1.558$).These parameters are "energetically conservative". Numerical dissipation must be reduced as most as possible to ensure accuracy of the solution. Contact is treated with the penalty method (see section 2.4) without augmentation. Updating of the Hessian matrix occurs at each iteration.

Evolution of left edge's velocity is illustrated at Figure 10. Oscillations in the end of the computation are a typical numerical problem of an implicit scheme. To reduce them, more dissipative parameters are to be chosen but they will reduce the accuracy of the solution with the introduction of numerical dissipation at lower frequency. The most precise solution is the one obtained with the new algorithm and integration error e_3. The other ones are more dissipative (the average velocity at the third interval is decreasing below $5m/s$) and jumps obtained in velocity are less sharp. Relative computation costs are illustrated at Figure 11.

The most accurate solution (with the new algorithm and integration error e_3) is about

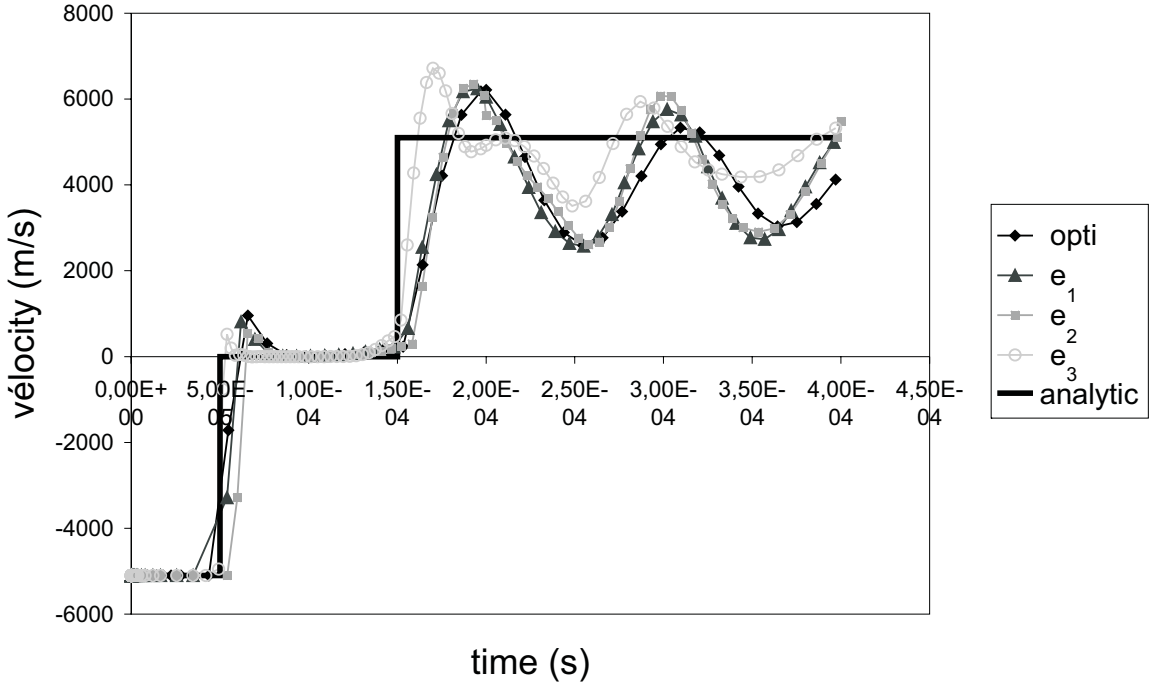

Figure 10: *Velocity of left edge for contact of an elastic bar*

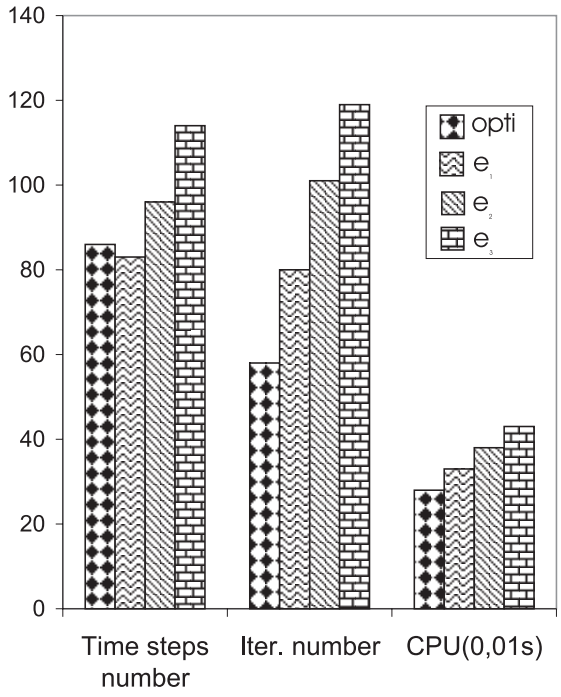

Figure 11: *Computation costs for contact of an elastic bar*

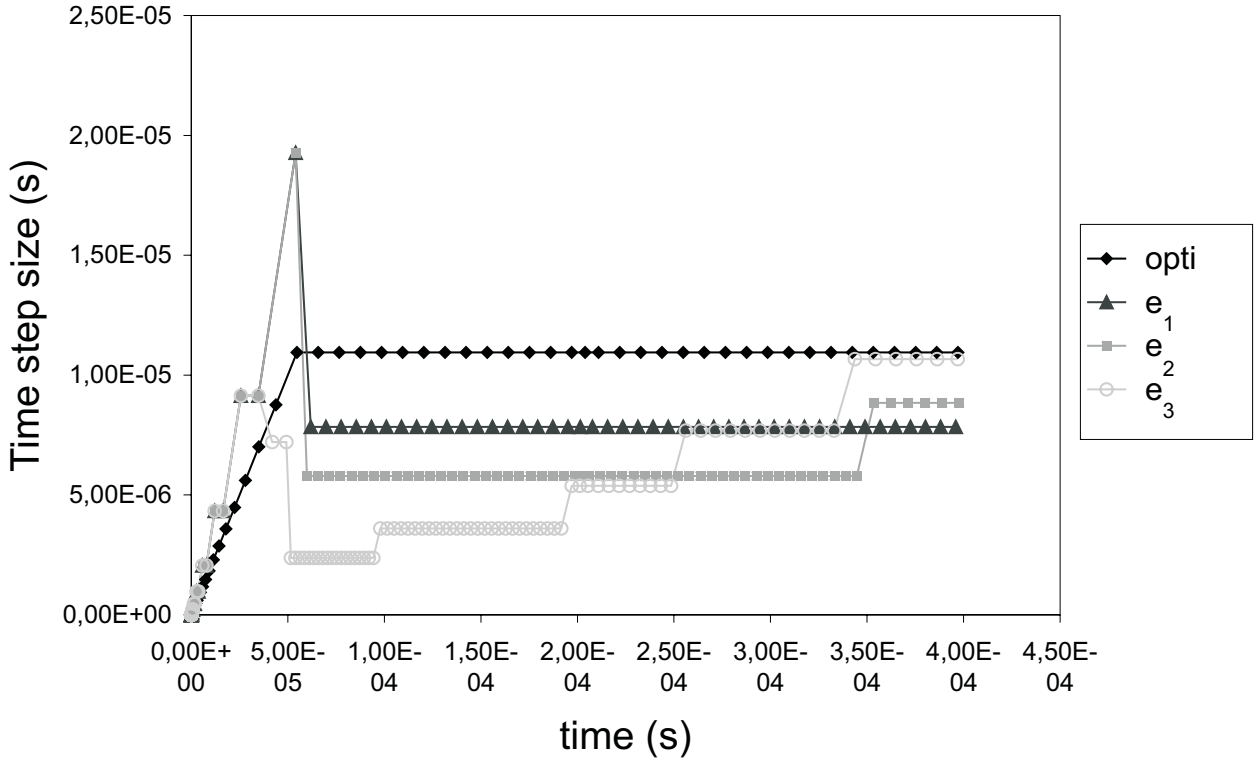

Figure 12: *Evolution of time step size for contact of an elastic bar*

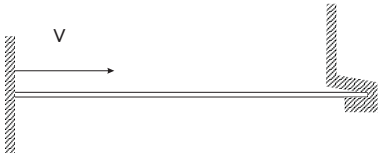

Figure 13: *Model of dynamic buckling of a cylinder*

70% more expensive than the cheaper one (*opti*) in term of CPU. Next come e_2 (40 % more expensive than *opti*) and e_1 (15 % more expensive than *opti*). The time step size evolution is given on Figure 12. The time step size is is reduced for the four computations when the bar enters into contact (at 50 μs). The method that reduces the less the time step size is the *opti* one. The one that reduces the most the time step size is the new algorithm with error e_3. This method is also the most accurate and, since the time step size augments with time, it is not too expensive. When error e_1 is used, the time step size is a slightly greater than with error e_2. Computation cost is then a little lower while the solution shows a similar level of accuracy (see Figure 10).

3.4.2 Case 2: Dynamic buckling of a cylinder

A hollow cylinder [8], [16], [17] (properties in Table 2) enters into contact with a rigid matrix (Figure 13). The left edge of the cylinder is constrained to move with a constant velocity of 27 m/s. A reference computation is defined. This reference is a computation of the problem with a small constant time step ($\Delta t = 0.1\ \mu s$). Evolution of the configuration at 0.475 ms intervals is shown on Figure 14. The solution obtained after 4.75 ms is illustrated

Internal diameter	$d_i = 27mm$
External diameter	$d_e = 31.17mm$
Length	$l = 180mm$
Density	$\rho = 7850kg/m^3$
Young modulus	$E = 210E9N/m^2$
Poisson coefficient	$\nu = 0.3$
Yield stress	$\sigma_0 = 700N/mm^2$
Hardening parameter	$h = 808N/mm^2$
Matrix velocity	$v = 27m/s$

Table 2: *Properties of dynamic buckling if a cylinder*

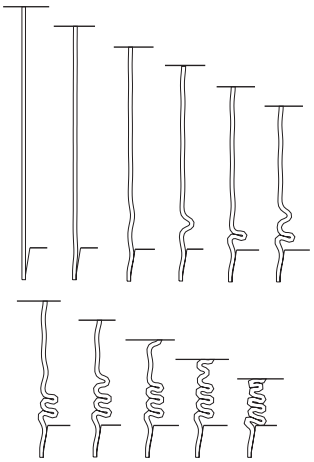

Figure 14: *Configuration (at 0.475 ms intervals) for dynamic buckling of a cylinder*

at Figure 15. Then the problem is resolved with the proposed time step control algorithm. The error indicator employed is successively relation (45), relation (46) and relation (47), respectively denoted e_1, e_2 and e_3. The tolerance parameter $PRCU$ is also equal at 10^{-4}. Finally, the problem is also solved with Ponthot's [3] method described in section 3.1. Three computations are represented. They are respectively a computation with an initial time step size equal to 0.1 ms, with an initial time step size equal to 10 μs and with an initial time step size equal at 1 μs . These solutions are respectively called $opti_4$, $opti_5$ and $opti_6$. The need to compute three solutions is due to the fact that the final configuration is radically different for the three solutions (see the orientation and the number of loops in Figure 16). With the developed algorithm, the first time step is rejected until the integration error is lower than the requested accuracy (and not until the step converges as in *opti* method). Therefore no trials on initial time step size is needed. The problem is solved with the generalized-α trapezoidal scheme (α_M = -0.87 and other parameters automatically computed to have a stable scheme, i.e. $\alpha_F = 0.1$, $\gamma = 1.48$ and β =0.98). Contact is treated with the penalty method (see section 2.4) without augmentation. Updating of the Hessian matrix occurs at each iteration.

Those solutions obtained after 4.75 ms is illustrated at Figure 16. When the *opti* is used, the final configuration depends on the choice of the initial time step size. It must be chosen under 1 μs to lead to the correct configuration. With the new developed algorithm and error e_1, there is a difference of 5 % in the Von-Mises stress with the reference configuration (Figure 15). This difference is about 2 % with error e_2 and 5 % with error e_3. Nevertheless, that is

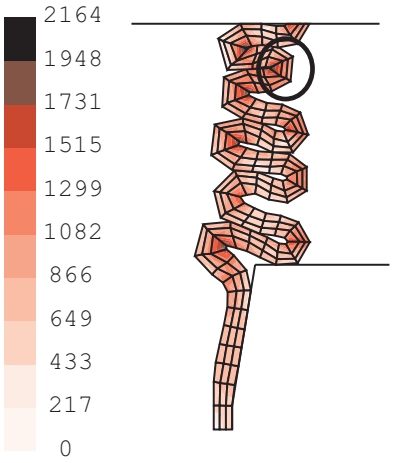

Figure 15: *Reference configuration and Von-Mises stress (N/mm^2) for dynamic buckling of a cylinder*

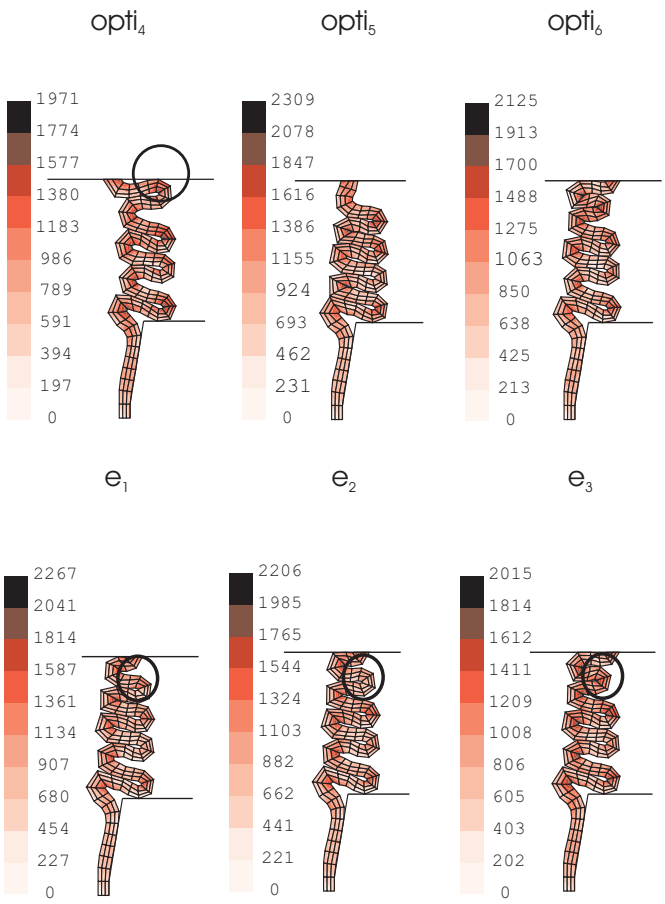

Figure 16: *Configuration and Von-Mises stress (N/mm^2) for dynamic buckling of a cylinder*

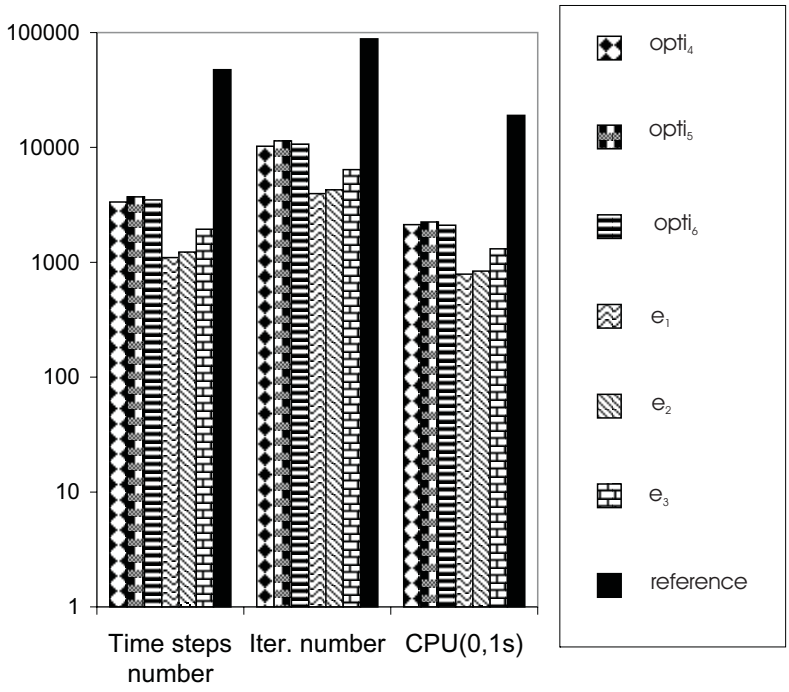

Figure 17: *Computation costs for dynamic buckling of a cylinder*

solution e_3 that is closer to the reference (see circles on Figures 15 and 16).

Relative computation costs are illustrated at Figure 17. Automatic time step size control allows reducing the computation cost. In this example, the number of iterations obtained with the proposed algorithm is approximately 60 % (note the logarithmic scale) lower than the number of iterations with the *opti* method. Computation time (CPU) is 5% lower with error e_1 than with error e_2 and 40 % lower than with error e_3.

3.4.3 Case 3: Metal sheet forming of an U

It consist of forming an U from a sheet of metal (Figure 18). Properties of this problem are reported in Table 3. During the first 0.1 ms, the blank-holder is moving down by 2 nm to establish contact. Afterwards, the punch is moving down by $70mm$, pulling the sheet metal in the rigid die. This is a plane strain problem that is almost quasi-static. It can then validate our algorithm when the accelerations are small (remember that the error estimator is accelerations depending (31)). A reference computation is defined. This reference is a computation of the problem with a small constant time step ($\Delta t = 50\mu s$). This computation is represented at Figure 19. Then the problem is resolved with the proposed time step control algorithm. The error indicator employed is successively relation (45), relation (46) and relation (47), respectively denoted e_1, e_2 and e_3. The tolerance parameter $PRCU$ is equal at 10^{-3}. Finally, the problem is also solved with Ponthot's [3] method described in section 3.1. The optimal number of iteration is taken equal to 4. Those parameters, leading the time step strategy, are taken a little higher than in previous cases, due to the fact that the problem is smoother than previous ones. The problem is solved with the generalized-θ mid-point scheme ($\theta = 1.1$). Contact is treated with the penalty method (see section 2.4) without augmentation. Updating of the Hessian matrix occurs at each iteration.

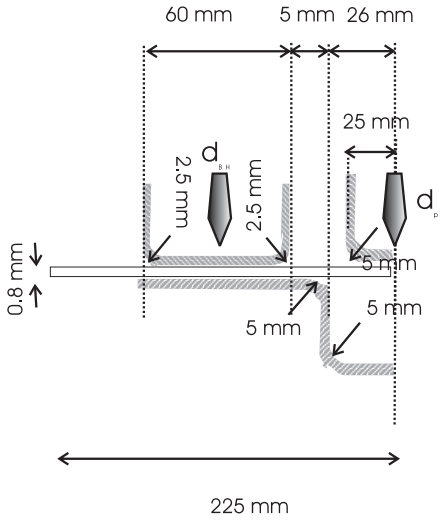

Figure 18: *Schema of metal sheet forming of an U*

Height	$h = 0.8mm$
Length	$l = 225mm$
Density	$\rho = 7850kg/m^3$
Young modulus	$E = 206E9N/m^2$
Poisson coefficient	$\nu = 0.3$
Yield stress	$\sigma_0 = 700N/mm^2$
Hardening parameter	$h = 808N/mm^2$
Blank-holder displacement	$d_BH = 2nm$
Blank-holder displacement time	$t_BH = 0.1s$
Punch displacement	$d_P = 70mm$
Punch displacement time	$t_BH = 1s$

Table 3: *Properties of metal sheet forming of an U*

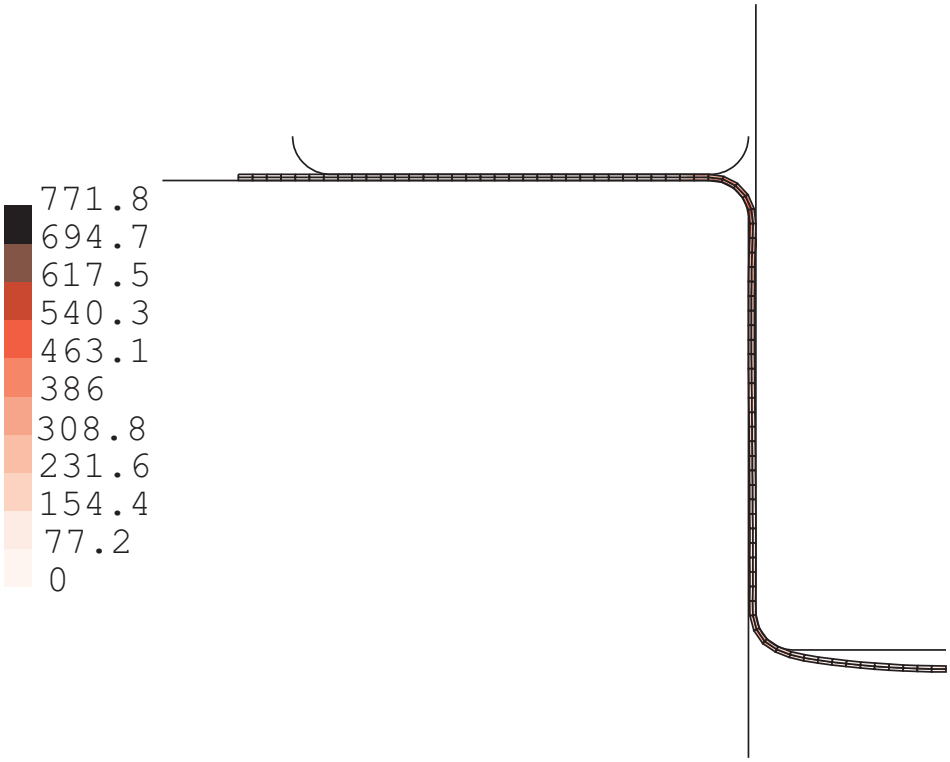

Figure 19: *Configuration and Von-Mises stress (N/mm^2) for metal sheet forming of an U*

method	Maximal Von-Mises stress $\sigma_0^{max}(N/mm^2)$
opti	771.8
e_1	771.8
e_2	771.8
e_3	771.8

Table 4: *Maximal Von-Mises stress for metal sheet forming of an U*

The maximal Von-Mises stresses obtained are reported in Table 4. Since the deformations are similar to the reference configuration, they are not reported. Therefore, all the solutions can be said to be similar.

Relative computation costs are represented on Figure 20. The method that minimize the number of time step is the *opti* one. Nevertheless, since the new developed algorithm with error e_1 minimize the number of iteration, it minimize also the CPU time (10 %) less than the *opti* method). It is then the one that adapt the more efficiently the time step size with the evolution of the problem. The newly developed algorithm with error e_1 is a little more expensive than with e_2 (2 %) and when error e_3 is used, the algorithm is 10 % more expensive than with e_2.

3.4.4 Case 4: 3D-Taylor's bar.

A cylindrical bar (properties in Table 5) with an initial velocity enters into contact with a rigid wall. There is no practical interest in a three dimensional model but it will serve to

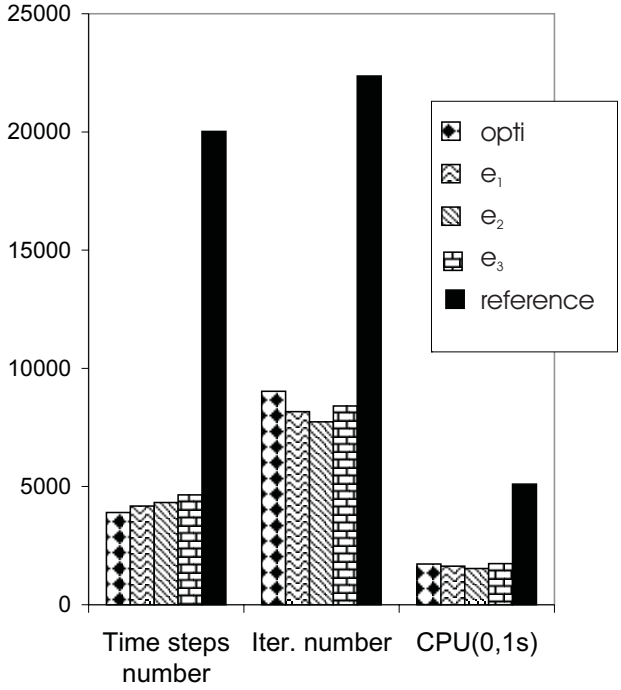

Figure 20: *Computation costs for sheet metal forming of an U*

External diameter	$d_e = 6.4mm$
Length	$l = 32.4mm$
Density	$\rho = 8930kg/m^3$
Young modulus	$E = 117E9N/m^2$
Poisson coefficient	$\nu = 0.35$
Yield stress	$\sigma_0 = 400N/mm^2$
Hardening parameter	$h = 100N/mm^2$
Initial velocity	$\dot{x}_0 = 227m/s$

Table 5: *Properties of 3D-Taylor's bar*

validate our algorithms. A reference computation is defined. This reference is a computation of the problem with a small constant time step ($\Delta t = 0.05\ \mu s$). The problem is resolved with the proposed time step control algorithm. Error indicators employed are successively those defined by relation (45), relation (46) and relation (47), respectively denoted e_1, e_2 and e_3. Due to the difficulty for this kind of problem to be integrated with the generalized-α trapezoidal scheme, a tolerance $PRCU$=10^{-3} is used for this numerical example. Finally, the problem is also solved with Ponthot's [3] method described in section 3.1. For the same reason that with the new algorithm, the optimal number of iteration is taken equal to 4. This solution is called *opti*. The problem is solved with the generalized-α trapezoidal scheme ($\alpha_M = -0.997$ and other parameters automatically computed to have a stable scheme, i.e. $\alpha_F = 0.05$, $\gamma = 1.997$ and $\beta = 1.558$). Updating of the Hessian matrix occurs at each iteration.

The solution obtained after 80 μs is illustrated at Figure 21. The maximum stress obtained with the new algorithm is similar to the reference configuration, respectively for error indicators e_1, e_2 and e_3 to within about 0.5 %, 2% and 0.1 %. Difference between Ponthot's

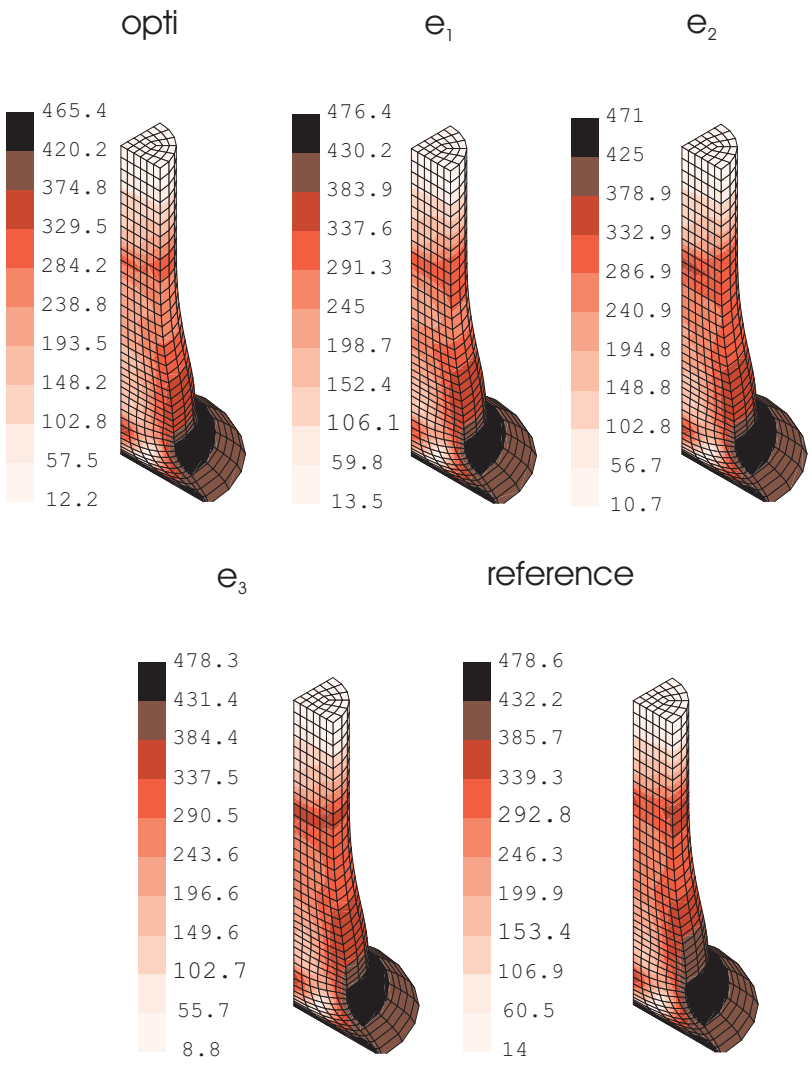

Figure 21: *Configuration and Von-Mises stress (N/mm^2) for 3D-Taylor's bar*

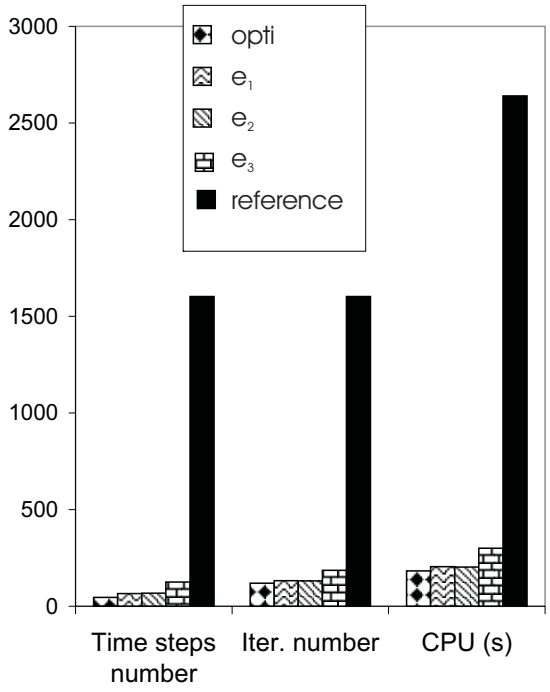

Figure 22: *Computation costs for 3D-Taylor's bar*

method and reference is 3 %.

Relative computation costs are illustrated at Figure 22. The method which minimizes the number of iterations and CPU cost is the *opti* one but it is also the less accurate. Next come the newly developed algorithm with error e_1 or e_2 that is 10 % more expensive. Finally the newly developed algorithm with error e_3 is 16 % more expensive than *opti* method. Therefore, the solution which gives the better results (good accuracy with a low computation cost, i.e. CPU time) is the newly developed algorithm with error e_1. Time step size evolution is illustrated on Figure 23. It appears that the automatic time stepping allows to work with a time step size very small during the impact ($t \simeq 0\ s$) as with the constant time step strategy. Nevertheless, after that impact, while the velocity and the evolution of non-linearities decrease quickly, the time step size increase. The newly developed algorithm that increase the most the time step size is the one with error e_1. Let us note that the automatic time stepping developed lead to constant time steps size for long time intervals.

3.4.5 Preliminary conclusions on the time step size control

From the studied cases, we can already say that the automatic time step size control algorithm developed is more accurate than *opti* method. Indeed, a guarantee of accuracy is introduced (that depends on the *PRCU* the user has defined). Computation costs can be mush lower (see section 3.4.2) than with the *opti* method. Moreover, the tolerance parameter *PRCU* has a physical sense by opposition at the optimal number of iteration of the *opti* method. Better results were generally obtained when estimator error e_1 (45) is used instead of e_2 (46).Indeed, in case 1 (see section 3.4.1, error e_1 gives a sharper jumps of velocity and a lower CPU time, in case 2 (see section 3.4.2) error e_1 gives a lower CPU time (5 %) for a nearly similar configuration (1%), in case 4 (see section 3.4.4) error e_1 give a better configuration (1%) in the same CPU time. For case 3 (see section 3.4.3), error e_2 gives a

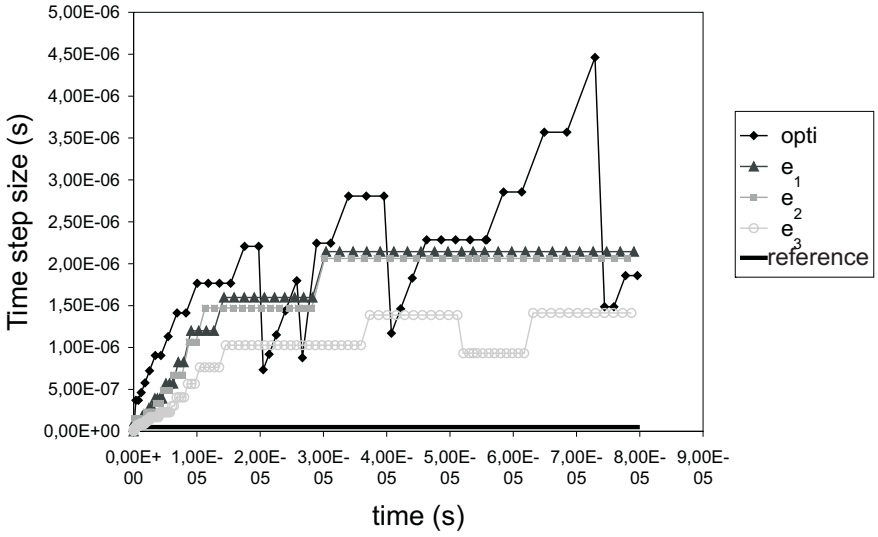

Figure 23: *Time step size evolution for 3D-Taylor's bar*

lower CPU time, but since the solutions are strictly identical for all computations, it is not very significant. Moreover, with error e_2, since the inertial forces and the accelerations of two successive time step are used (46) to compute the error, it is a little more expensive than with error e_1, where only the accelerations is used (45). Error e_3 (47) is more severe but is also more expensive than e_1. In fact, indicator error e_2 was developed for linear problems [10] and indicators e_1 or e_3 remain more appropriate for non-linear cases, the cost of e_3 always being greater than e_1.

Now the automatic criterion of Hessian matrix updating is introduced (see section 3.2.1). Regarding the previous conclusions, the time step size control used is the developed one with the error estimator e_1. Then three solutions are compared. The first one is a computation with updating at each iteration that is noted UPD_1. The second one is a computation with the automatic criterion (see section 3.2.1) for each iterations (before and after the first augmentations of the Lagrangian) and it is noted UPD_{auto}. The third one is the one developed at section 3.2.2. Then before the first augmentation, there is updating at each iteration and after the first augmentation, the automatic criterion of updating is used. This solution is noted $UPD_{1,auto}$. Naturally, the augmented Lagrangian method is employed for the three computations.

3.4.6 Case 5: Hemming

The problem is the one of clipping together two sheets of metal (Figure 24). The three matrix (M_i, i=1,...,3) move successively to clip the two sheets. Matrix displacements (d_i, i=1,..,3) are represented on Figure 25. The direction of the displacements are the ones represented by arrows on Figure 24. Properties of this problem are reported in Table 6. The two sheets are similar except in their length. The time step size control used is the developed one with the error estimator e_1, with $PRCU$ taken equal to 10^{-3}. The tolerance on the gaps is 0.1 nm. There is no projection between the augmentations. Three solutions are compared. The first one is a computation with updating at each iteration that is noted UPD_1. The second one is a computation with the automatic criterion (see section 3.2.1) for each iterations (before and after the first augmentations of the Lagrangian (see section 2.4)).

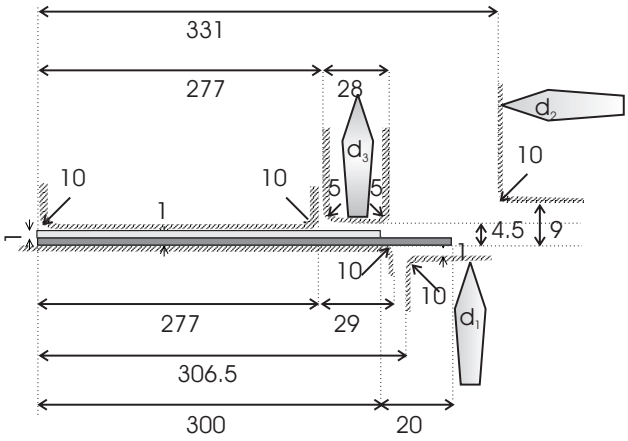

Figure 24: *Scheme of hemming (dimensions in mm)*

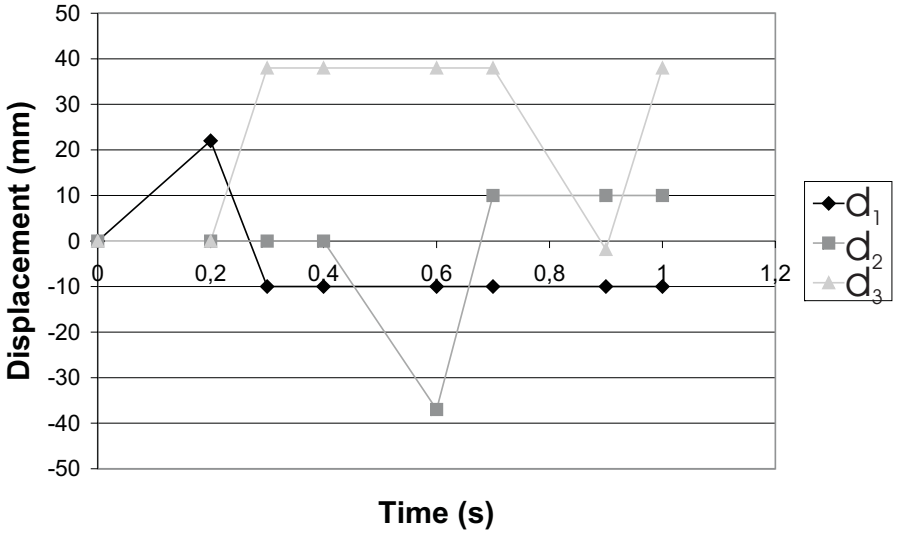

Figure 25: *Matrix displacements of hemming (dimensions in mm)*

Thickness	$e = 1mm$
Length of long sheet	$l_l = 320mm$
Length of short sheet	$l_s = 300mm$
Density	$\rho = 7850kg/m^3$
Young modulus	$E = 206E9N/m^2$
Poisson coefficient	$\nu = 0.3$
Yield stress	$\sigma_0 = 700N/mm^2$
Hardening parameter	$h = 808N/mm^2$

Table 6: *Properties of hemming*

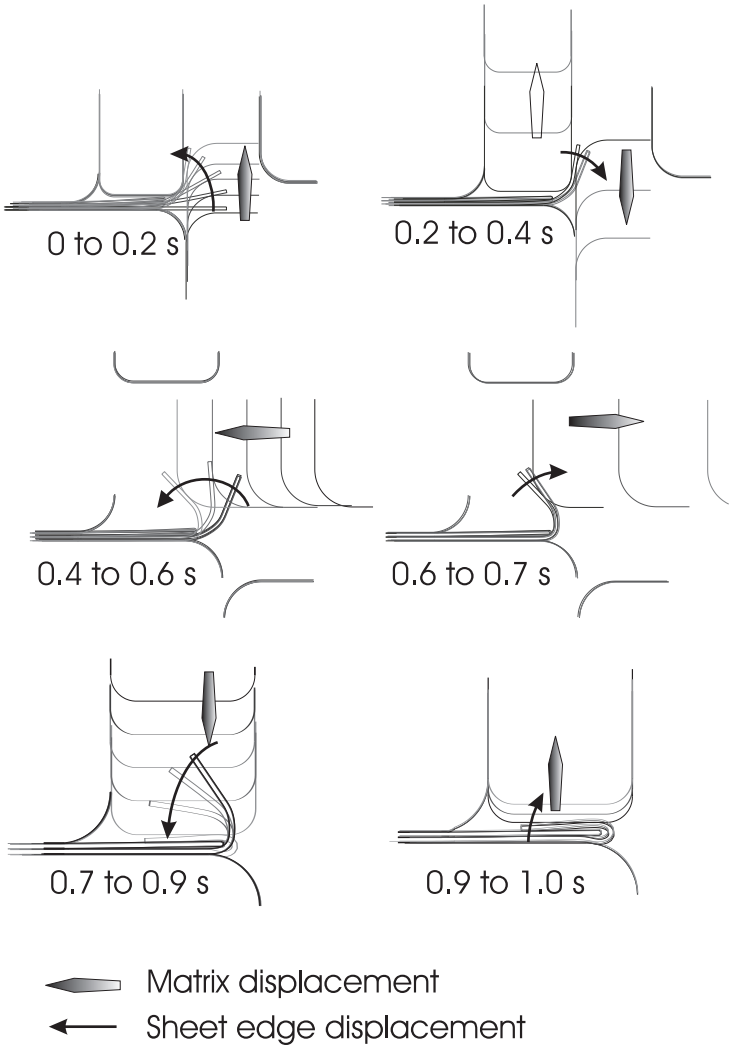

Figure 26: *Configurations (at 0.05 s intervals) for hemming*

It is noted UPD_{auto}. The third one is the one developed at section 3.2.2. Then before the first augmentation, there is updating at each iteration and after the first augmentation, the automatic criterion of updating is used. This solution is noted $UPD_{1,auto}$. Evolution of the configuration at 0.05 s intervals is shown in figure 26.

Final configuration and Von-Mises stress for UPD_1 solution are represented on Figure 27. The configuration is similar for the three computations. The maximal Von-Mises stress obtained after 1 s for each solution is reported in Table 7. Therefore, the three solutions can be said to be similar.

Relative computation costs are represented on Figure 28. The solution that maximize the number of iterations is UPD_{auto} (10 % more than with UPD_1 method). nevertheless, a lot of those iterations are less expensive than for the UPD_1 method, due to the fact that the Hessian matrix is not updated. Then the CPU time of method UPD_1 is higher (20% more) than for UPD_{auto}. When method $UPD_{1,auto}$ is used, the number of iterations is the same

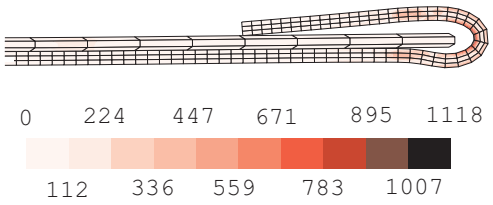

Figure 27: *Configuration and Von-Mises stress* (N/mm^2) *for hemming*

method	Maximal Von-Mises stress $\sigma_0^{max}(N/mm^2)$
UPD_1	1118
UPD_{auto}	1119
$UPD_{1,auto}$	1118

Table 7: *Maximal Von-Mises stress for hemming*

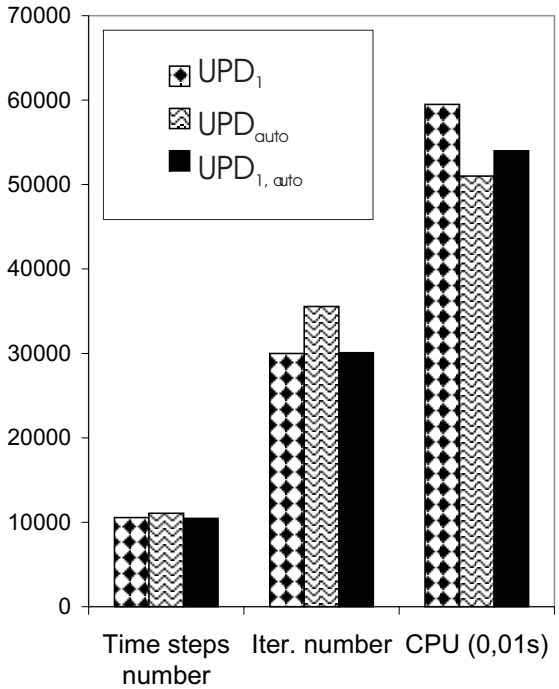

Figure 28: *Computation costs for hemming*

Length	$l = 600mm$
Density	$\rho = 8900kg/m^3$
Young modulus	$E = 200E9N/m^2$
Poisson coefficient	$\nu = 0.3$
Yield stress	$\sigma_0 = 200N/mm^2$
Hardening parameter	$h = 630N/mm^2$
Matrix velocity	$\dot{z_0} = 25m/s$

Table 8: *Properties of dynamic buckling of a 3D-bar*

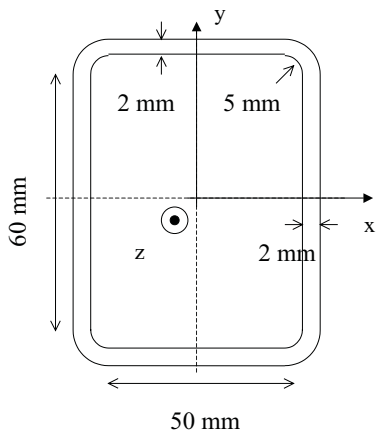

Figure 29: *Section of buckling of a 3D-bar*

than for UPD_1 method. The CPU cost is lower (about 12%) with $UPD_{1,auto}$ method than with UPD_1 method, but is higher (about 8 %) than with UPD_{auto} method. It is due to the fact that the needless iterations after the first augmentation are avoided but not the needless iterations before the first augmentation.

When studying the time step numbers, it appears that it is not constant. The reasons is that since the solutions after convergence depends on the update or not of the Hessian matrix (iterations stop when the residue is lower than a limit and not when it is equal, see relation (18)), the integration error is not the same. Therefore, time step size is not exactly the same and there is a difference of about 1 % of the number of time steps.

Now a problem with stronger non-linearities is studied, to see if the proposed algorithm is also efficient for such this of problem.

3.4.7 Case 6: Dynamic buckling of a 3D-bar

The problem is the dynamic buckling of a straight prism with a height of 600 mm (Table 8) with a uniform section (Figure 29). Properties of the bar are given in Table 8. The two external sections are fitted.

The problem is the one of a automobile stringer during a frontal crash. The bar has an initial velocity (25 m/s) parallel to the axis when it enters into contact with a rigid wall. But to simulate the vehicle inertia, the opposite edge of the bar is kept moving at a constant speed. As in the previous case, only error e_1 is used. The tolerance on the gaps is taken equal to 15 nm. Tolerance $PRCU$ is taken equal to 10^{-4}. Three solutions are compared. The first

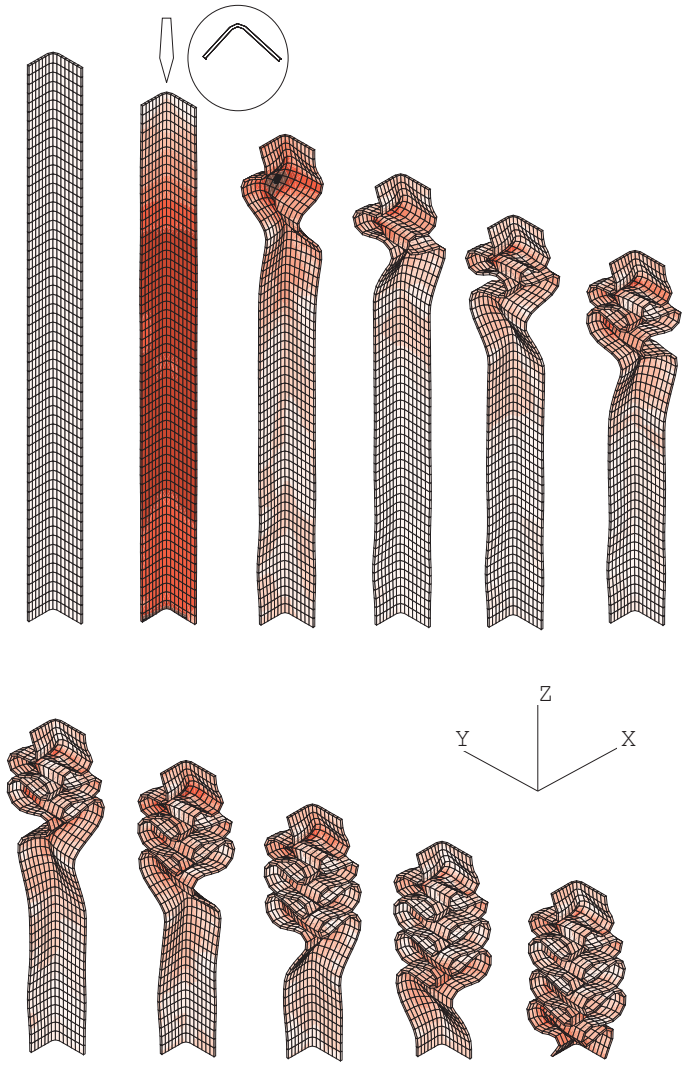

Figure 30: *Configuration (at 1.7 ms intervals) for dynamic buckling of 3D-bar (representation of a fourth of the bar)*

one is a computation with updating at each iteration that is noted UPD_1. The second one is a computation with the automatic criterion (see section 3.2.1) for each iterations (before and after the first augmentations of the Lagrangian (see section 2.4)). It is noted UPD_{auto}. The third one is the one developed at section 3.2.2. Then before the first augmentation, there is updating at each iteration and after the first augmentation, the automatic criterion of updating is used. This solution is noted $UPD_{1,auto}$. The problem is solved with the generalized-α trapezoidal scheme ($\alpha_M = -0.997$ and other parameters automatically computed to have a stable scheme, i.e. $\alpha_F = 0.05$, $\gamma = 1.997$ and $\beta = 1.558$). Evolution of the configuration at 1.7 *ms* intervals is shown in Figure 30.

The solution obtained after 17 *ms* is illustrated at Figure 31. The three configurations are similar and the maximal Von-Mises stress are nearly the same (about 3 % of variation).

Relative computational costs are represented on Figure 32. The UPD_1 method is the most

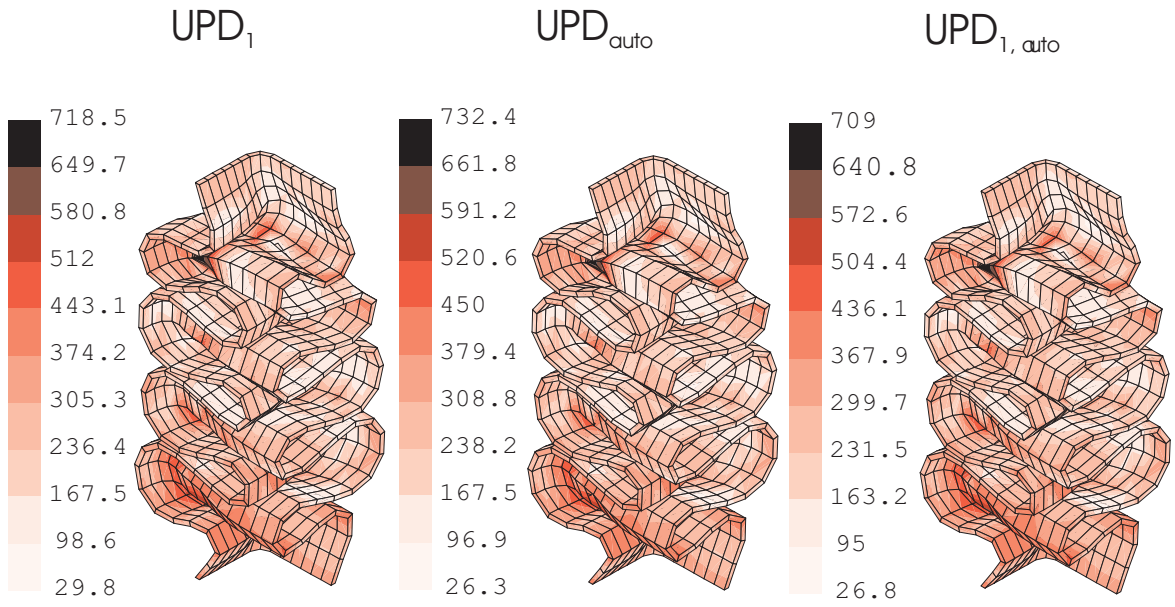

Figure 31: *Final configuration and Von-Mises stress (N/mm^2)for dynamic buckling of 3D-bar (representation of a fourth of the bar)*

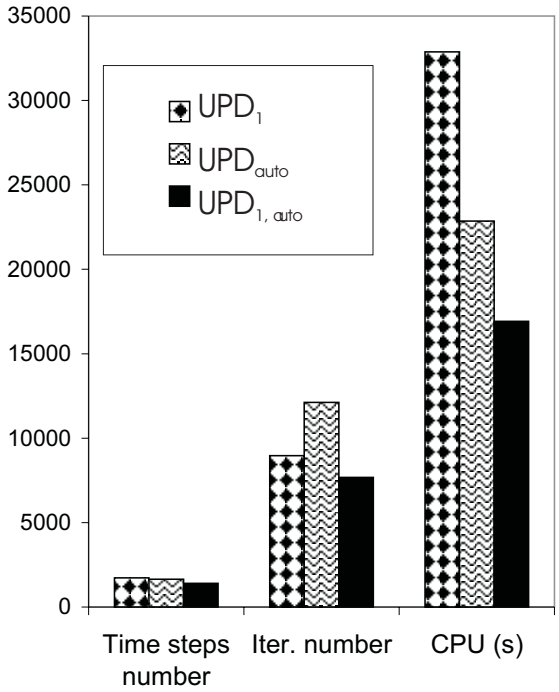

Figure 32: *Relative computation costs for dynamic buckling of 3D-bar*

expensive one in term of CPU. This method is about 50 % more expensive than UPD_{auto} method and is twice more expensive than $UPD_{1,auto}$. Since the number of strong non-linearities are important due to the auto-contact, the $UPD_{1,auto}$ method is more efficient than the UPD_{auto} method. In fact, before the first augmentation, most of iterations without updating are rejected and are replaced by iteration with updating (see 3.2.2). This phenomena could be expensive for some time steps and is avoided with the $UPD_{1,auto}$ method. Note that the number of time steps is not constant as explained in section 3.4.6.

3.4.8 Preliminary conclusions on automatic criterion of Hessian matrix updating

The automatic criterion of updating can be used for each iterations (UPD_{auto}) or just for iterations after the first augmentation while updating at each iteration can occurs before the first augmentation ($UPD_{1,auto}$). Both solution has provided better results than updating at each iterations. But the solution $UPD_{1,auto}$ has provide better result than UPD_{auto} method (and than UPD_1) method for a problem with a lot of strong non-linearities (see section 3.4.7). A future advance will be to change from UPD_{auto} method to $UPD_{1,auto}$ method and vice versa in the evolution of a problem. A method could be to modify the $RAPRES$ value (see section 3.2.1)with the augmentation number.

Let us note that the $UPD_{1,auto}$ method make the augmented Lagrangian method more attractive than usual. The augmented Lagrangian method allows to work with small penalty coefficient for a lower gap. But the number of iterations increase, and the advantage of a better conditioned Hessian matrix (due to the lower penalty coefficient) is balanced [8]. But with the proposed method, those iterations are very less expensive and the augmented Lagrangian method is more efficient

4 Parameters control for explicit schemes

Explicit scheme are conditionally stable. The time step size must be lower than a limit: the critical time step size (see section 2.3). Nevertheless, if this time step ensure stability for a linear problem, it is not necessarily true for a non-linear problem. Then the integration error can be used to solve this situation by modifying the security coefficient (γ_s) on the time step size. This solution is then used on the problem of dynamic buckling of a cylinder.

4.1 Control of the security coefficient (γ_s)

As seen in section 2.3, the stability imposes the time step size to be lower than a critical time step [1] that depends on the maximum pulsation ω of the body. A security coefficient γ_s is introduced to take into account the fact that the problem is non-linear (the theory to estimate Δt_{crit} is based on a linear scheme). Then it comes from (27) : $\Delta t = \gamma_s \Delta t_{crit} = \gamma_s \frac{2}{\omega_{max}}$.

The maximal pulsation ω_{max} is evaluated by the power iteration method [18]. It allows to really evaluate ω_{max} and not an approximation. Nevertheless, a security coefficient γ_s must be introduced to have a stable integration. A problem is how to evaluate γ_s and possibly to modify it with the problem evolution. From relations (24) and (25), we have:

$$x_{n+1} = x_n + \Delta t \dot{x}_{n-\frac{1}{2}} + \Delta t^2 \ddot{x}_n \tag{49}$$

Indeed, the error is of the third order as in an implicit scheme (see section 3.1.1): $e =$

$O\left(\frac{1}{6}\Delta t^3 \dddot{x}\right) \simeq O\left(\frac{1}{6}\Delta t^2 \ddot{x}\right)$. Therefore we can also write:

$$e = \frac{\Delta t^2}{6} \left\| \Delta \ddot{x} \right\| \tag{50}$$

As for an implicit scheme, this expression must be available for each problem. Then it is made adimensional (x_0 is the vector of the initial positions):

$$e = \frac{\Delta t^2}{6 \left\| x_0 \right\|} \left\| \Delta \ddot{x} \right\| \tag{51}$$

To process similarly as with the implicit scheme, expression (51) is divided by a reference. This reference is the average error (on a period) for a one-degree of freedom linear oscillator. Assuming time step size is constant, and pulsation is ω, we can define the non-dimensional pulsation as $\Omega = \omega \Delta t$. For such a problem, equations (24), (25) et (26) can be rewritten:

$$\begin{cases} x_{n+1} = x_n + \Delta t \dot{x}_{n+\frac{1}{2}} \\ \Delta t \dot{x}_{n+\frac{1}{2}} = \Delta t \dot{x}_{n-\frac{1}{2}} + \Delta t^2 \ddot{x}_n \\ \Delta t^2 \ddot{x}_{n+1} + \Omega^2 x_{n+1} = 0 \end{cases} \tag{52}$$

or:

$$\begin{pmatrix} x_{n+1} \\ \Delta t \dot{x}_{n+\frac{1}{2}} \\ \Delta t^2 \ddot{x}_{n+1} \end{pmatrix} = A\left(\Omega\right) \begin{pmatrix} x_n \\ \Delta t \dot{x}_{n-\frac{1}{2}} \\ \Delta t^2 \ddot{x}_n \end{pmatrix} \tag{53}$$

with:

$$A\left(\Omega\right) = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ -\Omega^2 & -\Omega^2 & -\Omega^2 \end{pmatrix} \tag{54}$$

And finally, assuming:

$$\begin{pmatrix} x_n \\ \Delta t \dot{x}_{n-\frac{1}{2}} \\ \Delta t^2 \ddot{x}_n \end{pmatrix} = \begin{pmatrix} x_0 \cos\left(\omega t\right) \\ -\Omega x_0 \sin\left(\omega t - \frac{\Omega}{2}\right) \\ -\Omega^2 x_0 \cos\left(\omega t\right) \end{pmatrix} \tag{55}$$

it leads to:

$$\begin{pmatrix} \Delta x \\ \Delta t \Delta \dot{x} \\ \Delta t^2 \Delta \ddot{x} \end{pmatrix} = \left[A\left(\Omega\right) - I\right] \begin{pmatrix} x_0 \cos\left(\omega t\right) \\ -\Omega x_0 \sin\left(\omega t - \frac{\Omega}{2}\right) \\ -\Omega^2 x_0 \cos\left(\omega t\right) \end{pmatrix} \tag{56}$$

Therefore the error (51) for the one-degree of freedom linear oscillator can be expressed as:

$$e = \frac{\Omega^2 \left\| -\cos\left(\omega t\right) + \Omega \sin\left(\omega t - \frac{\Omega}{2}\right) + \left(\Omega^2 + 1\right) \cos\left(\omega t\right) \right\|}{6} \tag{57}$$

or:

$$e = \frac{\Omega^3 \left\| \sin(\omega t) \cos\left(\frac{\Omega}{2}\right) + \left(\Omega - \sin\left(\frac{\Omega}{2}\right)\right) \cos(\omega t) \right\|}{6} \tag{58}$$

The reference is the average error for a period. It can be noted ε:

$$\varepsilon = \frac{\omega}{2\pi} \int_{t=0}^{t=\frac{2\pi}{\omega}} e dt \tag{59}$$

Using (58), expression (59) is explicitly calculated:

$$\varepsilon(\Omega) = \frac{\Omega^4}{3\pi} \sqrt{1 - \frac{\sin\left(\frac{\Omega}{2}\right)}{\frac{\Omega}{2}}} \tag{60}$$

Error (51) is then divided by ε (expression (60)). However, Ω need to be known to estimate ε. For the one-degree of freedom linear oscillator, assuming ω=ω_{max}, if the critical time step size ($\Delta t = \frac{2}{\omega_{max}}$) is used, then we have Ω=2 and ε=68%. That lead to integrate the system with about three step on a period and then with an error of 68%. If 10 time steps in a period are imposed, the obtained error is about 0.1%. It gives a good accuracy with a relatively low computation cost. Therefore with the non-dimensional pulsation, which corresponds to a $0.1Hz$ frequency, given by $\Omega_k = 0.6$, we define, using (51):

$$e_{exp} = \frac{\Delta t^2}{6\varepsilon(\Omega_k) \|x_0\|} \|\Delta \ddot{x}\| \tag{61}$$

Then we remark that for an one-degree-of-freedom linear system, the time step size must be lower than the critical time step size to ensure accuracy. For a real problem with more than one degree of freedom, the relation ω=ω_{max} is false if we consider ω as the pulsation of a mode having a significant energy. Nevertheless, there is no guarantee of accuracy using the critical time step. Therefore, the time step size could be evaluated to have an integration error (61) lower than a tolerance parameter $PRCU$ but must remain lower than the critical time step size (27) to ensure stability. A solution consist on modifying the security coefficient γ_s to keep the error lower than the tolerance and keeping security coefficient lower than 1 to ensure stability. It leads then:

$$\Delta t = \gamma_s \frac{2}{\omega_{max}} \quad \text{with} \quad \gamma_s < 1 \text{: } e_{exp} \leq PRCU \tag{62}$$

The method to evaluate γ_s is the one that permit to evaluate the time step size in section (3.1.2) with two modifications:

- The ratio RAT obtained is applied on the security coefficient γ_s
- The exponent η of relation (48) is taken equal at 4 since $e_{exp} \sim \div\Omega^4$ (see relation 60).

Now a numerical example is computed to validate these developments.

4.2 Numerical example

A strongly non-linear problem is considered. We will see that for this problem a security coefficient (27) lower than 0.3 must be used to solved it. But with the adaptive algorithm

γ_s	Time steps number	CPU (min)
0.9	*interrupted*	*interrupted*
0.4	*interrupted*	*interrupted*
0.3	*interrupted*	*interrupted*
0.2	192214	21.2
adaptive	148992	15.7

Table 11: *Computation costs for dynamic bucking of a cylinder with an explicit scheme*

developed, the security coefficient is adjusted (relation 62) to minimize the number of time steps without degrading accuracy.

4.2.1 Case 1: Dynamic buckling of a cylinder

This problem is the one previously studied in section 3.4.2 but solved with an explicit scheme. Five solutions are compared:

- A constant security coefficient of $\gamma_s = 0.9$ is used.
- A constant security coefficient of $\gamma_s = 0.4$ is used.
- A constant security coefficient of $\gamma_s = 0.3$ is used.
- A constant security coefficient of $\gamma_s = 0.2$ is used.
- The adaptive algorithm is used with an initial security coefficient of $\gamma_s = 0.9$ and a tolerance $PRCU$=1E-4

The first computation was interrupted after 0.7 ms due to an instability. The second one was interrupted after 2.5 ms for the same reason. The third computation was interrupted after 4.64 ms. The fourth computation was successful with 192214 time steps. Finally the fifth computation was also successful with 148992 time steps.

Those computation costs are resumed in Table 11. Figure 33 shows that with the new adaptive algorithm, the time step size (through security coefficient γ_s) is as high as possible but decreases with time to keep stability and accuracy.

The final configuration of the computation is illustrated on Figure 34. It appears that for solution $\gamma_s = 0.4$ and $\gamma_s = 0.3$, the Von-Mises stress exceed 7000 N/mm^2. This is due to the fact that an instability has appeared just before the computation was interrupted. The two solutions which were successful (fixed γ_s =0.2 and adaptive γ_s) give a solution nearly equal to the reference of the implicit scheme computation (see Figure 15) at 1 %. Therefore, the developed algorithm has reduced the number of time steps with in about 25 % (comparing with a strategy with fixed γ_s). Nevertheless, however the problem was highly dynamical, computation with an implicit scheme had needed 1.3 min (see Figure 17) versus 15 min with the explicit scheme. Indeed, the time step size of an implicit scheme could be hundred times greater for the same accuracy, and the number of degrees of freedom is small (463). Therefore, the iterations cost time of an implicit time step remains smaller than the cost time of hundred time steps of an explicit scheme. It will not be the same for problems with more degrees of freedom.

5 General conclusions

A new time step size control algorithm has been presented for implicit schemes. This algorithm is based on the measure of an integration error. By introducing counters, the time

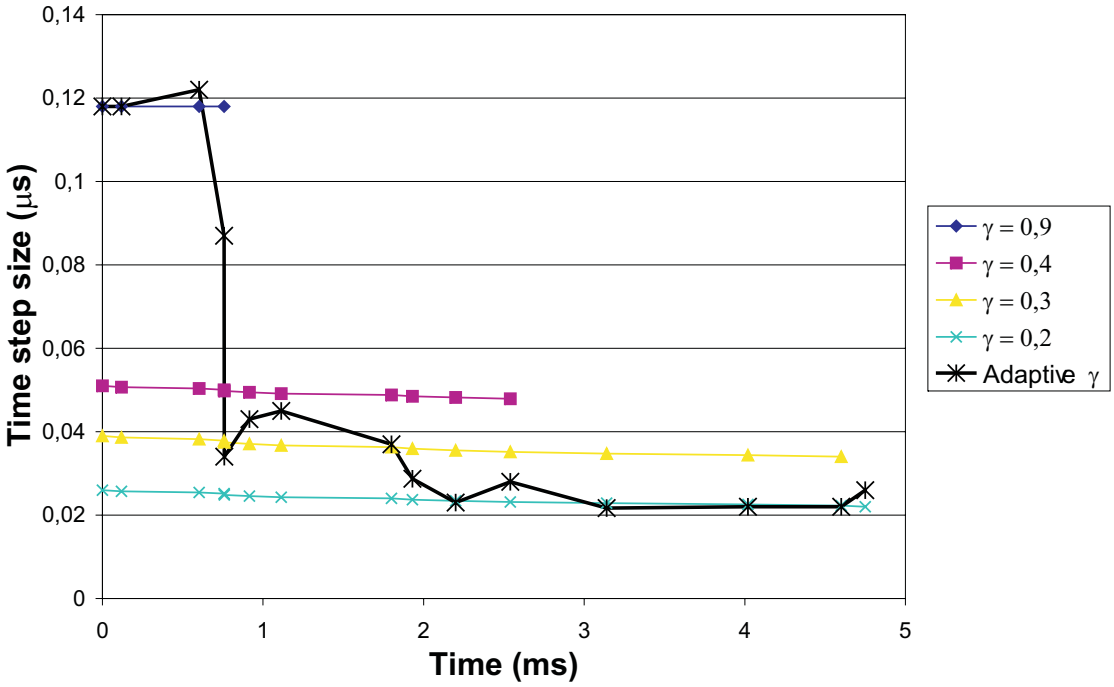

Figure 33: *Evolution of time step size for dynamic buckling of a cylinder with an explicit scheme*

step size is modified only for physical and durable variations in dynamical problems. But for a sudden change such as an impact or a contact, the integration error increases in one time step and the algorithm reduces instantaneously the time step size. By modifying the error threshold under which the time step can be augmented, when the problem becomes smoother, the time step size can increase rapidly. Thus this algorithm gives a good accuracy with a low computation time and a constant time step for long periods. If problems of convergence occur, tolerance on the integration error is reduced to adapt the time step size. Time step which are costly but nevertheless rejected are thus avoided. This algorithm is then applied to problems with contacts and large deformations. Associated to an estimator of the integration error based on the average acceleration jump (relation 45), the algorithm has been shown to ensure accuracy at a relatively low cost on very dynamical problems but also on quasi-static problems.

Next, an algorithm deciding if the Hessian matrix must be re-evaluated has been proposed. This algorithm re-computes the Hessian matrix only if it is necessary for convergence. If not, the old Hessian matrix is used in the iterative process and the overall computation time is reduced. This algorithm has been validated on strongly non-linear problems. It is especially efficient for quasi-linear problems. For problems with strong non-linearities and a few number of degrees of freedom this algorithm corresponds to updating at each iteration with the disadvantage of having to reject some iterations without updating. But when the augmented Lagrangian method is used for contact, such a problem becomes quasi-linear after the first augmentation. Therefore, the developed algorithm can be used after the first augmentation. The proposed method does not update the Hessian matrix when the iterations converge quickly enough so that the increase in the number of iterations is balanced by the diminution of their cost. A criterion of divergence was also implemented. It considers that the problem does not converge if the non-dimensional residual does not globally decrease

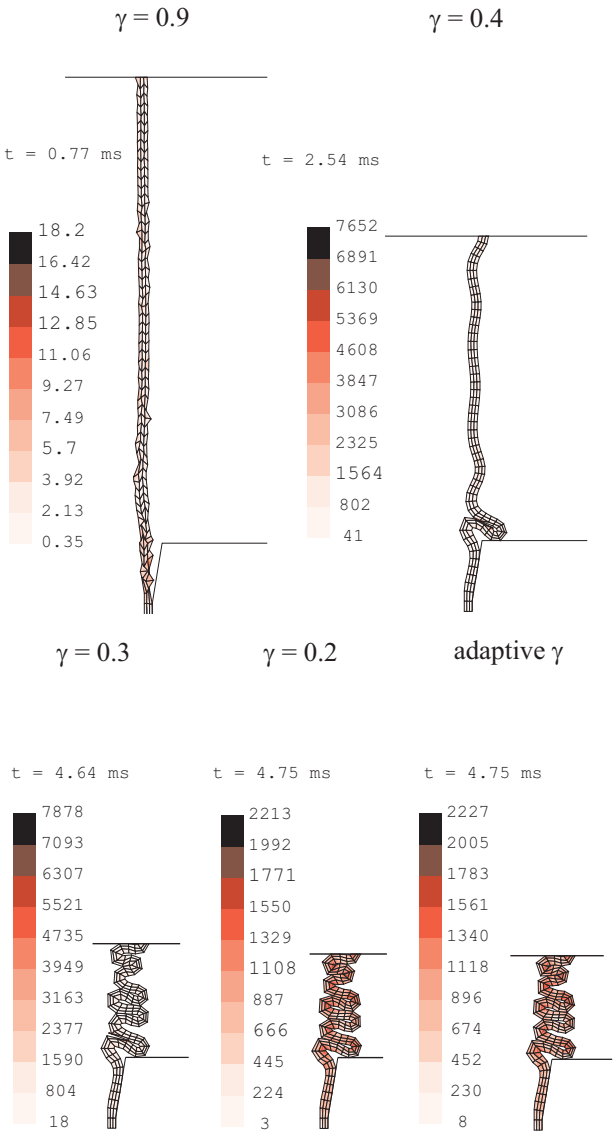

Figure 34: *Configuration and Von-Mises stress (N/mm^2) for dynamic buckling of a cylinder with an explicit scheme*

when iterating. In fact, it considers the evolution of the five last non-dimensional residual. A lot of needless iterations are thus avoided.

Finally, the proposed time step algorithm is used on explicit schemes to consider the presence on non-linearities. The time step size is the critical one, but to ensure accuracy and stability, it is multiplied by a security coefficient that depends on the integration error. This algorithm allows to work with a large security coefficient ($\gamma_s \sim 1$) when it is possible, and to reduce it when the non-linearities lead to instabilities. Then, this algorithm is less expensive than one with a constant small security coefficient.

References

[1] T. Belytschko and T.J.R. Hughes. *Computational methods for transient analysis.* North Holland, 1983.

[2] T.J.R. Hughes. *The finite element method.* Prentice Hall, 1987.

[3] J.-P. Ponthot. *Traîtement unifié de la mécanique des milieux continus solides en grandes transformations par la méthode des éléments finis (in French).* PhD thesis, université de Liège, 1995.

[4] J.-P. Ponthot and M. Hogge. On relative merits of implicit schemes for transient problems in metal forming simulation. In *International conference on numerical methods for metal forming in industry*, 1994.

[5] M. Hogge and J.-P. Ponthot. Efficient implicit schemes for transient problems in metal forming simulation. In *NUPHYMAT'96, Numerical and physical study of material forming processes*, 1996.

[6] M. Géradin and D. Rixen. *Mechanical vibrations (Theory and applications to structural dynamics).* Masson, 1994.

[7] J. Chung and J.M. Hulbert. A time integration algorithms for structural dynamics with improved numerical dissipations: the generalized-α method. *Journal of Applied Mechanics*, 60, 1993.

[8] D. Graillet. *Modélisation tridimensionnelle du contact entre structures à paroies minces dans les phénomènes d'impacts et de mise à forme (in French).* PhD thesis, Université de Liège, To appear.

[9] D. Givoli and I. Henisberg. A simple time-step control scheme. *Communications in numerical methods in engineering*, 9, 1993.

[10] M. Géradin. *Analyse, simulation et conception de systèmes polyarticulés et déployables.* Cours IPSI, 1997.

[11] A. Cassano and A. Cardona. A comparaison between three variable-step algorithms for the integration of the equations of motion in structural dynamics. *Latin American research*, 21, 1991.

[12] G.M. Hulbert and I. Jang. Automatic time step control algorithms for structural dynamics. *Computer methods in applied mechanics and engineering*, 126, 1995.

[13] A. Dutta and C.V. Ramakrishnan. Accurate computation of design sensitives for structures under transient dynamic loads using time marching scheme. *International journal for numerical methods in engineering*, 41, 1998.

[14] SAMTECH. User manual of samcef, v8.0. Technical report.

[15] L. Noels. Détermination automatique de la taille du pas de temps pour les schémas implicites en dynamique non-linéaire (in french). Travail de fin d'étude, université de Liège, 2000.

[16] D. Graillet and J.-P. Ponthot. Efficient implicit schemes for the treatment of the contact between deformable bodies: Applications to shock-absorber devices. *IJCrash*, 4(3):173–286, 1999.

[17] T.A. Laursen. *Formulation and treatment of frictional contact problems using finite elements*. PhD thesis, Department of mechanical engineering Stanford University, 1992.

[18] D. J. Benson. Stable time step estimation for multi-material eulerian hydrocodes. *Computer methods in applied mechanics and engineering*, 167, 1998.