Automatic time stepping algorithms for implicit numerical simulations of non-linear dynamics

Ludovic Noels [(1)], Laurent Stainier[1] [(1)], Jean Philippe Ponthot [(1)], Jérôme Bonini [(2)]

[(1)] Aerospace Laboratories (LTAS-MCT), University of Liège,

Chemin des Chevreuils 1, 4000 Liège, Belgium

[(2)] SNECMA-Moteurs, Engineering Division, Centre de Villaroche, 77550 Moissy-Cramayel, France

ABSTRACT

When an implicit integration scheme is used, variable step strategies are especially well suited to deal with problems characterised by high non-linearities. Constant step size strategies generally lead to divergence or extremely costly computations. An automatic time stepping algorithm is proposed that is based on estimators of the integration error of the differential dynamic balance equations. Additionally, the proposed algorithm automatically takes decisions regarding the necessity of updating the tangent matrix or stopping the iterations, further reducing the computational cost. As an illustration of the capabilities of this algorithm, several numerical simulations of both academic and industrial problems are presented.

KEYWORDS

Computer simulations, plasticity, automatic time-stepping, non-linear dynamics, contact-impact

[1] Corresponding author : l.stainier@ulg.ac.be (e-mail), +32-(0)4-366-9141 (fax)

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. 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, if the time step size is chosen too small, the calculation is very expensive (in term of computation time), while if it is chosen too large, the integration is not accurate enough or the iterations diverge (when solving the balance equations). Therefore, the 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.

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 with 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 (a priori) 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 computation 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.

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 when there is no negative Jacobian, convergence is not garanteed since the residual is not ensured 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. 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

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 [1-5]:

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

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. F^{ext} 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). Both vectors are non-linear 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:

$$x_0 = x(t_0) \text{ and } \dot{x}_0 = \dot{x}(t_0) \tag{2}$$

2.2 Implicit schemes: The generalized-α trapezoidal scheme

The most general scheme for implicit integration of (1) is a generalized trapezoidal scheme [1, 2, 6] 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

$$\dot{x}_{n+1} = \dot{x}_n + (1-\gamma)\,\Delta t\,\ddot{x}_n + \gamma\,\Delta t\,\ddot{x}_{n+1} \tag{3}$$

$$x_{n+1} = x_n + \Delta t\,\dot{x}_n + \left(\frac{1}{2} - \beta\right)\Delta t^2\,\ddot{x}_n + \beta\,\Delta t^2\,\ddot{x}_{n+1} \tag{4}$$

The discretized equations of motion (1) can be rewritten under the from proposed by Chung and Hulbert [6]:

$$R_{n,n+1} = \frac{1-\alpha_M}{1-\alpha_F} M\,\ddot{x}_{n+1} + \frac{\alpha_M}{1-\alpha_F} M\,\ddot{x}_n + \left(F^{int}_{n+1} - F^{ext}_{n+1}\right) + \frac{\alpha_F}{1-\alpha_F}\left(F^{int}_n - F^{ext}_n\right) = 0 \tag{5}$$

Unconditional stability and second order accuracy of the scheme, for linear problems [6], require that the parameters verify the following conditions:

$$\begin{aligned} &\gamma \geq 1/2 - \alpha_M + \alpha_F \\ &\alpha_M \leq \alpha_F \leq 1/2 \\ &\beta \geq 1/4\left(1 + \alpha_F - \alpha_M\right)^2 \end{aligned} \tag{6}$$

Iterative solution of the nonlinear system (5) first requires the elimination of accelerations and velocities at time t_{n+1} with the help of (3) and (4), as well as the writing of the Hessian matrix of the system, i.e.

$$S = \left[\frac{1}{\beta \Delta t^2} \left(\frac{1-\alpha_M}{1-\alpha_F} \right) M + \frac{\gamma}{\beta \Delta t} C_T + K_T \right] \tag{7}$$

where K_T, C_T are respectively the tangent stiffness and damping matrices. The residual for iteration number $i+1$ is defined by:

$$R = \frac{1-\alpha_M}{1-\alpha_F} M \ddot{x}_{n+1}^i + \frac{\alpha_M}{1-\alpha_F} M \ddot{x}_n + \left[F^{int}\left(x_{n+1}^i, \dot{x}_{n+1}^i \right) - F^{ext}\left(x_{n+1}^i, \dot{x}_{n+1}^i \right) \right] + \frac{\alpha_F}{1-\alpha_F} \left(F_n^{int} - F_n^{ext} \right) \tag{8}$$

Then, the iterative solution of system (5, 6, 7) can be written as:

$$S \cdot \Delta x = -R \tag{9}$$

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

$$r = \frac{\|R\|}{\|F^{ext}\| + \|F^{int}\|} < \delta \tag{10}$$

2.3 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.

$$R_\theta = M\, \ddot{x}_{n+\theta} + F^{int}\left(x_{n+\theta}, \dot{x}_{n+\theta}\right) - F^{ext}\left(x_{n+\theta}, \dot{x}_{n+\theta}\right) = 0 \tag{11}$$

where

$$\ddot{x}_{n+\theta} = \frac{2}{(\theta\,\Delta t)^2}\left[x_{n+\theta} - x_n - \theta\,\Delta t\,\dot{x}_n\right] \tag{12}$$

$$\dot{x}_{n+\theta} = \frac{2}{\theta\,\Delta t}\left[x_{n+\theta} - x_n - \frac{\theta\,\Delta t}{2}\dot{x}_n\right] \tag{13}$$

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

$$S = \left[\frac{2}{(\theta\,\Delta t)^2} M + \frac{2}{\theta\,\Delta t} C_T + K_T\right] \tag{14}$$

The present scheme is $\ddot{x}_n$-independent, thus yielding the final acceleration as a post-treatment result :

$$\ddot{x}_{n+1} = \frac{\dot{x}_{n+1} - \dot{x}_n}{\Delta t} = \ddot{x}_{n+\theta} \tag{15}$$

3 Automatic time step size control

3.1 Introduction

A relatively simple method proposed by Ponthot [3] aims at an optimal number of iterations. If the number of iterations exceeds this optimal number, the next time step size is reduced, while, if the

number of iterations is lower than the optimal number, the time step size is augmented. Givoli and Henisberg [7] propose to modify the time step size to keep the displacements difference between two successive times lower than a given limit. Géradin [8] (Figure 1) estimates the integration error from the accelerations and the inertial forces difference between two successive times multiplied by the square of the time step size. This error is divided by a constant depending on the initial positions and by another constant that is the average error for a one-degree-of-freedom linear system (defined as in section 3.2), the result being the non-dimensional integration error *e*. This error must be lower than a given tolerance (*PRCU*). If this 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 half the tolerance to the power one third. If the error is lower than the tolerance divided by sixteen, the time step size is multiplied by two.

For Cassano and Cardona [9], 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 [10] (Figure 2) estimate the error from the accelerations difference multiplied by the square of time step size. This error is then divided by a term that depends on the positions difference. Their time step control algorithm is characterised 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 the error is lower than *TOL2* and higher than *TOL1*, the time step is accepted and its size is kept constant. If the error is lower than *TOL1* then the time step is accepted. If it occurs successively *LCOUNT* times, then the time step size is increased. The counter is introduced to avoid undesirable change in time step size due to the periodic nature of the local error.

Dutta and Ramakrishnan [11] 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, and 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.

The automatic control scheme presented in this paper is based on the algorithm proposed by Géradin [8], [12]. Nevertheless, due to the non-linear characteristics of the problems we are interested in, we make sure that the time step size reacts only on evolution in physical modes and not on numerical modes. Changes in time step size will also occur only if the new time step size can be kept constant for several steps. On the other hand, the error estimator based on the inertial forces difference (proposed by Géradin [8] 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.

3.2 Error estimator

The integration error is estimated from the truncation error of equations (3, 4) or equations (12, 13). Indeed, the truncation error is of the third order: $e_t = O\left(\frac{1}{6}\Delta t^3 \dddot{x}\right) \approx O\left(\frac{1}{6}\Delta t^2 \Delta\ddot{x}\right)$. Therefore we can write:

$$e_t = \frac{\Delta t^2}{6}\left\|\Delta\ddot{x}\right\| \tag{16}$$

First, this expression must be available for any problem. Then a non-dimensional error e_{nd} is defined (x_0 is the vector of the initial positions):

$$e_{nd} = \frac{\Delta t^2}{6\left\|x_0\right\|}\left\|\Delta\ddot{x}\right\| \tag{17}$$

To ensure that the error estimator can be used for each implicit scheme (the generalised-α

trapezoidal scheme or the generalised-θ mid-point scheme), and for each set of parameters (α_F, α_M, β, γ or θ) without modifying the tolerance on the error (see section 3.3), expression (17) is divided by a reference. This reference is the average error (on a period) for a one-degree-of-freedom linear oscillator. Assuming a constant time step size, and a pulsation ω, we can define the non-dimensional pulsation as $\Omega = \omega \Delta t$. For such a problem, equations (3) to (5) can be rewritten:

$$\begin{cases} x_{n+1} = x_n + \Delta t \ \ \dot{x}_n + (\frac{1}{2} - \beta)\Delta t^2 \ddot{x}_n + \beta \Delta t^2 \ddot{x}_{n+1} \\ \dot{x}_{n+1} = \dot{x}_n + (1-\gamma)\Delta t \ \ \ddot{x}_n + \gamma \Delta t \ \ \ddot{x}_{n+1} \\ (1-\alpha_M)\ddot{x}_{n+1} + \alpha_M \ddot{x}_n + (1-\alpha_F)\omega^2 x_{n+1} + \omega^2 \alpha_F x_n = 0 \end{cases} \tag{18}$$

or

$$\begin{pmatrix} x_{n+1} \\ \Delta t \dot{x}_{n+1} \\ \Delta t^2 \ddot{x}_{n+1} \end{pmatrix} = A(\Omega) \begin{pmatrix} x_n \\ \Delta t \dot{x}_n \\ \Delta t^2 \ddot{x}_n \end{pmatrix} \tag{19}$$

with

$$A(\Omega) = \frac{1}{D(\Omega)} \begin{pmatrix} 1-\alpha_M - \alpha_F \Omega^2 \beta & 1-\alpha_M & \frac{1-\alpha_M - 2\beta}{2} \\ -\gamma \Omega^2 & (1-\alpha_F)(\beta-\gamma)\Omega^2 + 1 - \alpha_M & 1-\gamma-\alpha_M + \Omega^2(\beta - \frac{\gamma}{2})(1-\alpha_F) \\ -\Omega^2 & (\alpha_F - 1)\Omega^2 & -\alpha_M + (1-\alpha_F)(\beta - \frac{1}{2})\Omega^2 \end{pmatrix}$$

where

$$D(\Omega) = 1 - \alpha_M + (1-\alpha_F)\Omega^2 \beta$$

And finally, assuming that for a one-degree-of-freedom linear oscillator we have $x_n = x_0 \cos(\omega t)$, and that equation (19) can be rewritten as:

$$\begin{pmatrix} \Delta x \\ \Delta t\,\Delta \dot{x} \\ \Delta t^2 \Delta \ddot{x} \end{pmatrix} = \begin{pmatrix} x_{n+1} \\ \Delta t\,\dot{x}_{n+1} \\ \Delta t^2 \ddot{x}_{n+1} \end{pmatrix} - \begin{pmatrix} x_n \\ \Delta t\,\dot{x}_n \\ \Delta t^2 \ddot{x}_n \end{pmatrix} = [A(\Omega) - I] \begin{pmatrix} x_n \\ \Delta t\,\dot{x}_n \\ \Delta t^2 \ddot{x}_n \end{pmatrix} \tag{20}$$

it finally comes:

$$\begin{pmatrix} \Delta x \\ \Delta t\ \Delta \dot{x} \\ \Delta t^2 \Delta \ddot{x} \end{pmatrix} = [A(\Omega) - I] \begin{pmatrix} x_0 \cos(\omega t\ ) \\ -\Omega x_0 \sin(\omega t\ ) \\ -\Omega^2 x_0 \cos(\omega t\ ) \end{pmatrix} \tag{21}$$

Therefore $\left\| \Delta t^2 \Delta \ddot{x} \right\|$ for the one-degree of freedom linear oscillator is deduced from relation (21):

$$\left\| \Delta t^2 \Delta \ddot{x} \right\| = \frac{x_0}{D(\Omega)} \left\| -\Omega^2 \cos(\omega\ t\ ) - \Omega^3 \left(\alpha_F - 1\right) \sin(\omega\ t\ ) - \Omega^2 \left(-\alpha_M + (1-\alpha_F)(\beta - 1/2)\Omega^2 - D(\Omega) \right) \cos(\omega\ t\ ) \right\| \tag{22}$$

and expression (17)becomes:

$$e_{nd} = \frac{\left\| \Delta t^2 \Delta \ddot{x} \right\|}{6\ \left\| x_0 \right\|} = \frac{(1-\alpha_F)\Omega^3 \left\| \sin(\omega t\ ) + \frac{\Omega}{2} \cos(\omega t\ ) \right\|}{6\ \left(1 - \alpha_M + (1-\alpha_F)\Omega^2 \beta\right)} \tag{23}$$

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

$$\varepsilon = \frac{\omega}{2\pi} \int_{t=0}^{t=\frac{2\pi}{\omega}} e_{nd}\, dt \tag{24}$$

Using (23), expression (24) yields:

$$\varepsilon(\Omega) = \frac{\left(1-\alpha_F\right)\Omega^3\sqrt{1+\frac{\Omega^2}{4}}}{3\pi \quad \left[1-\alpha_M+\left(1-\alpha_F\right)\Omega^2\beta\right]} \tag{25}$$

If $\alpha_M = 0$, then one gets back the expression calculated by Géradin [8], [12] for the HHT implicit scheme. Expression (25) is established for the generalised-α trapezoidal scheme, while for the generalised-θ mid-point scheme, the system (18) is replaced by:

$$\begin{cases} x_{n+1} = x_n + \Delta t \;\; \dot{x}_n + \frac{\Delta t^2}{2}\ddot{x}_{n+\theta} \\ \Delta t \;\; \dot{x}_{n+1} = \Delta t \;\; \dot{x}_n + \Delta t^2 \ddot{x}_{n+\theta} \\ \Delta t^2 \ddot{x}_{n+1} = \Delta t^2 \ddot{x}_{n+\theta} \end{cases} \tag{26}$$

with:

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

Therefore the matrix $A(\Omega)$ in expressions (19, 20, 21) becomes:

$$A(\Omega) = \frac{2}{2+\theta^2\Omega^2}\begin{pmatrix} 1+\frac{\Omega^2}{2}(\theta^2-1) & 1+\frac{\Omega^2}{2}(\theta^2-\theta) & 0 \\ -\gamma\Omega^2 & 1+\frac{\Omega^2}{2}(\theta^2-2\theta) & 0 \\ -\Omega^2\theta^2 & -\Omega^2\theta & 0 \end{pmatrix} \tag{27}$$

Finally, the reference error (25) is rewritten:

$$\varepsilon(\Omega) = \frac{\Omega^2 \sqrt{\left[\theta^2\Omega^2 + 2(1-\theta^2)\right]^2 + 4\theta^2\Omega^2}}{3\pi \quad \left[2 + \theta^2\Omega^2\right]} \tag{28}$$

The non-dimensional error (17) is then divided by ε (expression 25 or 28) to have an expression independent of the particular scheme used. 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 corresponding to a *0.1 Hz* frequency, given by $\Omega_k = 0.6$, we define, using (17):

$$e_1 = \frac{\Delta t^2}{6\,\varepsilon(\Omega_k)\,\|x_0\|}\|\Delta\ddot{x}\| \tag{29}$$

For linear systems, Géradin demonstrated ([8], [12]) that the error can be evaluated as expression (30). This error filters high frequency modes (as numerical modes). However, for non-linear systems, no advantage is gained (see academic examples 1 and 2) by replacing the acceleration difference by a term depending on the accelerations and the inertial forces difference as in (30), yielding:

$$e_2 = \frac{\Delta t^2}{6\,\varepsilon(\Omega_k)\,\left[x_0^T\, M\, x_0\right]^{1/2}}\left[\Delta\ddot{x}^T\,\Delta(M\ddot{x})\right]^{1/2} \tag{30}$$

Another possibility (Cassano and Cardona [9]) to evaluate the error is to keep the maximum acceleration difference ($L\infty$-norm) instead of the vector L2-norm. We define e_3, with *ndof* the number of degrees of freedom:

$$e_3 = \frac{\Delta t^2}{6\,\varepsilon(\Omega_k)\ \max\limits_{j=1,ndof}(x_0)_j}\ \max_{i=1,ndof}(\Delta\ddot{x})_i \tag{31}$$

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

3.3 Time step size control

In the next paragraphs, the symbol *e* is used to represent any one errors of e_1 (29), e_2 (30) or e_3 (31). The computed error must be of the order of a user-defined tolerance that is noted *PRCU*. A value of this tolerance that lead to a good accuracy to price ratio is typically 10^{-3}. 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 reduces *PRCU* (box 1, Figure 4). Moreover, the time step size is divided by *RDOWN*, that is initialised at 3 by default. After some steps without convergence problems, the tolerance *PRCU* can be augmented. This number of time steps is large enough to avoid oscillation in *PRCU* value. It could depend on divergence occurrences.

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

- The error is larger than *PRCU*/2, and the algorithm goes to box 3 (Figure 6): the error is considered to be too high, and to ensure a good accuracy, next time step size must be smaller.
- The error is in the interval [*TRHLD*, *PRCU*/2], and the algorithm goes to box 4 (Figure 7): the time step size ensures a good accuracy with a relatively low computation cost, and it is kept constant
- The error is smaller than a limit *TRHLD*, and the algorithm goes to box 5 (Figure 8): the error is considered to be too small, and to ensure a reduced computation cost, next time step

size must be larger.

Let us first examine the problem of too high an error (box 3, 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 the 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 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:

$$RAT = \left[\frac{PRCU}{2e} \right]^{1/\eta}, \quad \eta \in [2, 3] \tag{32}$$

For non-linear systems η can be out of this interval. To ensure that the time step size is sufficiently reduced, η is taken smaller than two. The factor that finally multiplies the time step size is $RAT=[0.5\ PRCU/ERRO]^{2/3}$. 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 $RAT=[0.5\ PRCU/e]^{2/3}$. If the error *e* is larger than a limit *REJL*, the time step is rejected and its new value is size is multiplied by $RAT=[0.5\ PRCU/e]^{2/3}$. *REJL* can be taken equal to *1.5 PRCU*.

If error is smaller than *PRCU/2* and higher than *TRHLD*, the time step is kept constant (box 4, Figure 7). Typical values for *TRHLD* are discussed in next paragraph.

Let us now examine the problem of too small an error (box 5, 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 that the time step size is not augmented too much, η from equation (32) is taken larger than 3. The factor multiplying the time step size is finally $RAT=[0.5\ PRCU/ERRT]^{1/5}$. 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. *PRCU/16*) and *CT* relatively large (e.g. 5) to ensure a good accuracy. In these conditions, the 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 *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*.*PRCU* /10.

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

4 Resolution of the Newton-Raphson iterations

4.1 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.

The evolution of the non-dimensional residual r (10) 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 much it could be interesting not to recalculate the Hessian matrix is the ratio *VALRF* between the time needed for an iteration with recalculation and an iteration without recalculation. This ratio indicates how much an iteration without recalculation could advantageously replace an iteration with recalculation.

The proposed algorithm is the following:

- The Hessian matrix is recalculated at the first iteration if the time step size has changed. Indeed, S significantly depends on Δt (7).
- If the number of the iterations is greater than *VALRF* (rounded to an integer), the next iteration is made with recalculation of the Hessian matrix. Then, iterations occur without recalculation only if it is less expensive.
- If the number of the iterations is lower than *VALRF* (rounded to an integer) in an integer, the Hessian matrix is recalculated only if the non-dimensional residual r has not been reduced by a ratio chosen equal to $RAPRES = VALRF/10 \in [0.2, 0.95]$.
- If the non-dimensional residual has not been divided by *RAPRES*, the next iteration then needs recalculation of the Hessian matrix. But ideally, this iteration does not take as initial values for $(x , \dot{x} , \ddot{x})$ the values at the end of the previous iteration, but the value at the end of the last iteration which has converged. Some divergences of the iterations are then avoided. For practical reasons, the implementation of this last remark in MECANO, one

softwares used to validate our algorithms, was not possible. Thus, the following solution has been adopted. If the non-dimensional residual has not been reduced, the next iteration occurs with recalculation and the initial values are the prediction values.

- If an iteration, with a number lower than *VALRF* (rounded to an integer) in an integer and larger than one, with recalculation of the Hessian matrix occurs, all the subsequent iterations of this step will occur with recalculation. Not updating risks to lead to a strategy that diverges or that requires more computation time than with updating.
- If the last iteration of the previous time step has needed recalculation of the Hessian matrix, the first iteration of the present step occurs with recalculation.

This algorithm avoids some needless recalculations and inversions of the Hessian matrix. For strongly non-linear problems with a small number of degrees of freedom, this algorithm is at worst as expensive as an algorithm with recalculation 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 recalculation. In fact, this algorithm allows a lot of iterations without recalculation when possible, and recalculates frequently the Hessian matrix when needed.

4.2 Criterion of divergence

Two problems of divergence can occur. First an element has a negative Jacobian. In this case, detection of divergence is easy to detect by verifying the Jacobian of element. A more difficult problem is to detect divergence when all Jacobian are positive, but when the evolution of the residual in 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 δ, 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. 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 recalculation of the Hessian matrix, the number of iterations is higher than when frequent recalculations occur. A solution consists in considering that divergence occurs if the non-dimensional residual has not been divided by two after 5 successive iterations with recalculation. Several iterations need to be considered, because when divergence occurs, the non-dimensional residual usually presents some oscillations.

5 Numerical examples

In a first part, three academic cases are studied. The problems are solved with the proposed time step control algorithm. The error indicator employed are successively those given by relation (29), relation (30) and relation (31), respectively denoted e_1, e_2 and e_3. A tolerance $PRCU=10^{-3}$ is used for all the problems except for problem 1 that is more difficult to integrate. For this problem, a tolerance $PRCU=10^{-4}$ is taken. Finally, the problems are also solved with Ponthot 's method [3] described in section 3.1. This solution is called "opti". The optimal number of iterations is taken equal to 4 except for problem 1. For the same reason than with new algorithm, the optimal number of iteration is thus taken equal to 2. The problems considered are solved within the formalism of large deformations and displacements. Academic cases were computed in the research code METAFOR [3], in which the automatic time step size control algorithm has been implemented. The criterion of automatic updating and of divergence were also introduced and studied on two others academic cases. Contact is treated with the penalty method [14].

In a second part, industrial problems are studied. The three algorithms (automatic time step size control, selective updating of the inverse Hessian matrix, divergence criterion) have been implemented in the dynamics module MECANO of SAMCEF [13]. In the commercial version of MECANO, time step size is chosen with the scheme proposed by Géradin [8], [12]. The user defines the number of the iterations with recalculation of the Hessian matrix and the maximum number of iterations. With the current algorithm, recalculation occurs at the first iteration of all time steps, according to the fact that time step size can change. Two industrial problems from SNECMA have

been computed with the old and the new algorithm. These problems were three dimensional models with thousands of degrees of freedom. Some elements are non-linear, simulating contacts, rupture… When comparing the new and the old algorithms, precision parameters *(* δ *, PRCU)* are taken identical. For the old algorithm, the iteration numbers with recalculation of the Hessian matrix are chosen to minimise the total computation cost. Some attempts were necessary to define these parameters for having a low cost without divergence of the problems.

5.1 Academic case 1: Contact of an elastic bar

An elastic bar in plane stress (properties in Table 1) with an initial velocity (Figure 9) of -5 m/s (minus sign comes from the orientation of x axis) enters into contact with a rigid matrix initially distant of 0.25 mm. Due to a Poisson coefficient equal to zero, and considering vertical displacement fixed to zero, the problem is one-dimensional. The analytic solution of this problem is knew. In the interval [0 s, 5 10^{-5} s], the bar is in translation at constant velocity towards the wall. Contact occurs at 5 10^{-5}s and the velocity of left edge becomes equal to zero. The velocity of the wave then appearing in the bar is $(E/\rho)^{1/2}$=5120 m/s. Given the length of the bar, the wave needs 10^{-4} s to go from the left edge to the right edge and back. The velocity of left edge is then equal to zero during the interval [5 10^{-5} s, 15 10^{-5} s] and becomes equal (due to conservation properties of an elastic problem) to 5 m/s after 15 10^{-5}s. The problem is solved with the generalized-α trapezoidal scheme *(*α_M = -0.997 and other parameters automatically computed to have a stable scheme, i.e. α_F = 0.05, γ = 1.997 and β =1.558*)*.These parameters are given the most "energetically conservative" values, given the conservative nature of the problem. The numerical dissipation must then be reduced as much as possible to ensure the accuracy of the solution. The evolution of left edge's velocity is illustrated at Figure 10. Oscillations at the end of the computation are a typical numerical problem of implicit schemes. To reduce these, more dissipative parameters should be chosen but they would reduce the accuracy of the solution with the introduction of numerical dissipation at lower frequencies. Relative computation costs are reported in Table 2. Critical observations of these results will be done for the three problems altogether in paragraph 5.3.

5.2 Academic case 2: Taylor's bar

A cylindrical bar (properties in Table 3) with an initial velocity enters into contact with a rigid wall. A reference computation is defined. This reference is a computation of the problem with a small constant time step (Δt = 0.17 μs). The problem is solved with the generalised-θ mid-point scheme (θ = 1.1). The solution obtained after 80 μs is illustrated at Figure 11. Relative computation costs are reported in Table 2. For this problem, a comparison with an explicit central difference scheme is made. The explicit final configuration is nearly identical to the implicit one (Figure 11) but, due to the low number of degree of freedom, the explicit solution is much more expensive (52 s).

5.3 Academic case 3: Dynamic buckling of a cylinder

A hollow cylinder [14, 15, 16] (properties in Table 4) enters into contact with a rigid matrix (Figure 12). The left edge of the cylinder is constrained to move with a constant velocity of 9090 mm/s. 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. α_F = 0.1, γ = 1.48 and β =0.98*)*. The evolution of the geometry every 1.1 ms is showed in Figure 13. The solution obtained after *12* ms is illustrated in Figure 14. Relative computation costs are illustrated in Table 2.

From the three cases studied up to now, we can say that the automatic time step size control algorithm developed is more accurate than the "opti" method. Indeed, a garanty of accuracy is introduced (which depends on the *PRCU* the user has defined). Computation costs (Table 2) can be lower (70 % for academic case 3) than with the "opti" method and are never much more expensive. Slightly better results (same accuracy and lower computational costs) were obtained when the error estimator e_1 (29) was used instead of e_2 (30). The error e_3 is more severe but is also more expensive than e_1. In fact, the error indicator e_2 was developed for linear problems [8], [12] and does not remain appropriate.

5.4 Academic case 4: 3D-Taylor's bar

This problem is similar to the academic case 1 but is treated as a three dimensional problem. The properties of the bar are those given in Table 1. There is no practical interest in a three dimensional model, but it will serve to validate our algorithms. On the basis of the conclusions drown from the previous cases, only error e_1 is used. Nevertheless, the automatic criterion of Hessian matrix updating is introduced. Therefore four resolutions are compared: the "opti" method, the new time stepping algorithms with error e_1 and with systematic re-computation of the Hessian matrix at each iteration, the new time stepping algorithms with error e_1 and with criterion of the Hessian matrix updating and finally a reference computation with a time step Δt=3.2 10^{-7} s and updating of Hessian matrix at each iteration. 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. α_F = 0.1, γ = 1.48 and β =0.98). The solution obtained after 80 μs is illustrated in Figure 15. Relative computation costs are reported in Table 5. Critical observations of these results will be done together with those on next example in paragraph 5.5.

5.5 Academic case 5: Dynamic buckling of a 3D-bar

The problem is the dynamic buckling of a prism of initial height 600 mm (Table 6), and of uniform section (Figure 16). Properties of the bar are given in Table 6. This example models an automobile stringer during a frontal crash. The bar than has initial velocity (25 m/s) parallel to its axis when it enters into contact with a rigid wall. To simulate the vehicle inertia, the opposite edge of the bar is kept moving at constant speed. The methods compared are identical to the previous case, but here, a constant time step strategy lead to a very expensive computation (a few days). Therefore we defined the reference computation, as resolution using error e_3 (that is, the most severe criteria) and a lower *PRCU* (i.e. 10^{-5}). Moreover we impose time step size to be kept lower than 10^{-5} s. The problem is solved with the generalised-θ mid-point scheme (θ = 1.1). The evolution of the geometry every 1.7 ms is showed at Figure 17. The solution obtained after 17 ms is illustrated in Figure 18. Relative computation costs are reported in Table 5.

When the criterion of automatic updating is introduced, the new time stepping algorithm requires a lower expensive (from 5% to 50% in term of CPU) computation cost than with the “opti” method, even if the solution is less accurate (15 % difference in the maximal von Mises stress for problem 5). Let us note that the updating criterion is efficient with the new time stepping because the time step size is kept constant on long periods and that the Hessian matrix must not be updated only because the time step size has changed.

This problem (and previous ones) leads to the next concluding remarks:

- For most problems, an automatic time stepping is necessary. A constant time step strategy is too expensive for practical usage.
- Time stepping algorithms based on an integration error ensure an accuracy that the “opti” method cannot ensure. This accuracy depends on the tolerance *PRCU* chosen by the user. For elaborate problems (buckling , auto-contact and dynamic effects), a better convergence (and a better accuracy) is obtained when *PRCU* is chosen equal to 10^{-4} than when it is choose at 10^{-3} (academic cases 1 and 5) or when “opti” (academic cases 1, 3 and 5) method is used. Indeed, the evolution of time step size is then more appropriate to the evolution of the problem.
- The automatic criterion for Hessian matrix updating allows to reduce computation times (CPU) in some cases (academic case 4). For problems with a lot of modifications of contact (academic example 5), the automatic criterion is reduced to a re-actualisation for most of the iterations. Nevertheless, the problem converges with a good accuracy (in critical academic case 5 there is a lack of 15% accuracy). It allows us to say that, for industrial problems where only a part of the elements are as critical as in academic case 5, the automatic updating criterion will allow to reduce computation costs without loss in accuracy.
- The robustness of the proposed algorithms is established since the algorithm has

always produced accurate results without leading to an exaggerated computational cost for all the highly non linear problems treated.

Let us now confirm these conclusions on industrial cases from SNECMA.

5.6 Industrial case 1

It consist in a three-dimensional model of unbalance in an aircraft engine. The number of degrees of freedom is about ten thousands. Some non-linear element are used (contact between blades and casing, contact between shaft and bearing …). This problem is solved with the old algorithms (commercial version of MECANO) and the new ones (time step control, Hessian matrix recalculation and divergence criterion). In both cases, the tolerance δ on the non-dimensional residual r (10) is taken equal to 10^{-3}. Tolerance *PRCU* on the integration error is taken equal to 10^{-3}. The initial time step size is the same. With the old algorithm, there is recalculation of the Hessian matrix for iterations 1, 3, 6, 7, 8, 9… Figure 19 is showing the time evolution of the displacement of a bearing degree of freedom. New algorithms give a solution nearly identical to the old ones.

Figure 20 shows the energy balance, i.e. the potential energy plus kinetic energy minus the work of external forces. If this balance is positive, energy appears with time and the computation is unstable. If this balance is negative, energy disappears. It could be due to physical dissipation or to numerical dissipation, and if the numerical dissipation is too high, the integration is not accurate. On Figure 20, we see that dissipation with new algorithms is a little lower (0.5%) than with the old algorithms. The new algorithms thus give a good accuracy. Moreover they decrease the computational cost (CPU) to 40% of the old ones.

5.7 Industrial case 2

It consist in a three-dimensional model of a bearing rupture in an aircraft engine. The number of degrees of freedom is about ten thousands. This problem is solved with the old algorithms (commercial version of MECANO) and the new ones (time step control, Hessian matrix

recalculation and divergence criterion). In both cases, tolerance δ on the non-dimensional residual r is taken equal to 10^{-4}. The tolerance *PRCU* on the integration error is taken equal to 10^{-3}. The initial time step size is the same. With the old algorithms, there is recalculation of the Hessian matrix for iterations 1, 3, 5, 6, 7, 8… Figure 21 represents the force evolution for an intact bearing degree on freedom. The new algorithms give the same solutions than the old ones. Figure 22 shows the evolution of time step size. With the new algorithm of time step control, the time step size is constant during longer periods. Costly updating of the Hessian matrix because of time step size changes can thus be avoided. New algorithms reduce the computation time (CPU) to 60% of the old ones.

6 Conclusions

A new time step size control algorithm has been presented. This algorithm is based on the measure of an integration error. By introducing counters, the time 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 limit under which the time step could be augmented, if the problem becomes smoother, time step size can increase rapidly. This algorithm thus gives a good accuracy with a low computation time and a constant time step for long period. If problems of convergence occur, tolerance on the integration error is reduced to adapt the time step size. Costly time step, nevertheless rejected, are thus avoided. This algorithm has been applied to academic problems with contacts and large deformations. Associated to an estimator of the integration error based on the average acceleration jump (relation 29), the algorithm has been shown to ensure accuracy at a relatively low cost.

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 computation time is reduced. Finally, a criterion of divergence was implemented. It considers that the problem does not converge if the

non-dimensional residual does not decrease when iterating. A lot of needless iterations are thus avoided.

These algorithms were also validated on academic cases. Moreover they were implemented in SAMCEF's module MECANO [13], with the proposed time stepping algorithm, and they were validated on industrial cases from SNECMA. Solutions obtained with these algorithms are similar to the old ones but computational cost have been reduced to about 50%.

References

[1] Belytschko T, Hughes TJR (editor). Computational Methods for Transient Analysis. North Holland, 1983.

[2] Hughes TJR. The Finite Element Method. Prentice-Hall, 1987.

[3] Ponthot JP. Traîtement unifié de la Mécanique des Milieux Continus solides en grandes transformations par la méthode des éléments finis. PhD thesis (in French). Liège, Belgium: Université de Liège, 1995.

[4] Ponthot JP, Hogge M. On relative merits of implicit / explicit algorithms for transient problems in metal forming simulation. International Conference on Numerical Methods for Metal Forming in Industry. Baden-Baden, Germany, Sept 1994; (2):128-148.

[5] Hogge M, Ponthot, JP. Efficient implicit schemes for transient problems in metal forming simulation. NUPHYMAT'96, Numerical and Physical Study of Material Forming Processes. CEMEF - Ecole Nationale supérieure des Mines de Paris Sophia-Antipolis: France, June 5-7 1996.

[6] Chung J, Hulbert, GM. A time integration algorithm for structural dynamics with improved numerical dissipations: the generalized-α method. Journal of Applied Mechanics. 1993; 60: 371-375.

[7] Givoli D, Henisberg, I. A simple time-step control scheme. Communication in Numerical Methods in Engineering. 1993; 9: 873-881.

[8] Géradin M. Analyse, simulation et conception de systèmes polyarticulés et structures déployables. Cours IPSI. Paris, 11-13 mars 1997.

[9] Cassano A, Cardona A. A comparison between three variable-step algorithms for the integration of the equations of motion in structural dynamics. Latin American Research. 1991; 21: 187-197.

[10] Hulbert GM, Jang I. Automatic time step control algorithms for structural dynamics. Computer Methods in Applied Mechanics and Engineering. 1995; 126: 155-178.

[11] Dutta A, Ramakrishnan CV. Accurate computation of design sensitivities for structures under transient dynamic loads using time marching scheme. International Journal for Numerical Methods In Engineering. 1998; 41: 977-999.

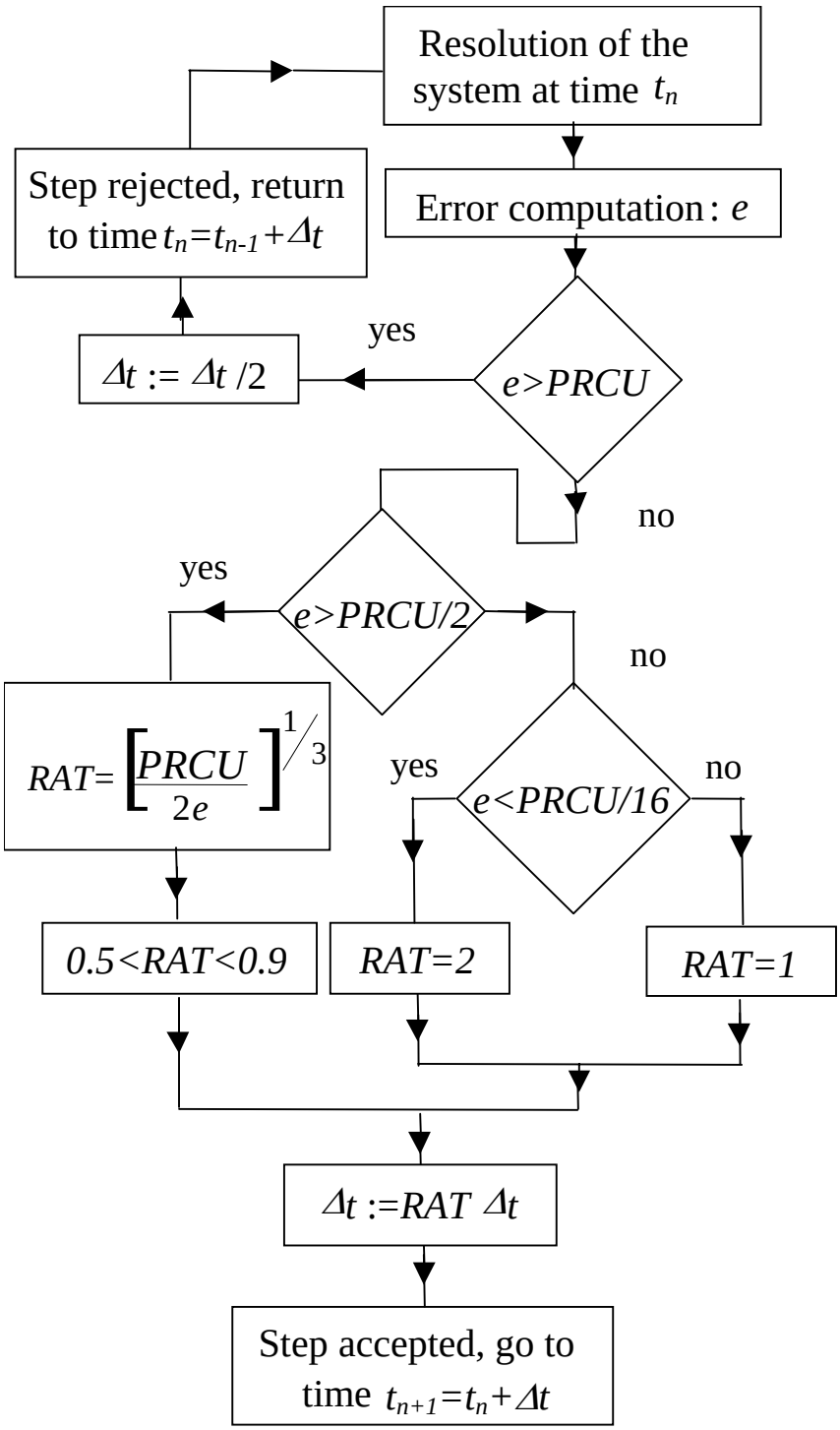
[12] Géradin M. Flexible multibody dynamics (A finite element approach). John Wiley and Sons Inc, 2000.

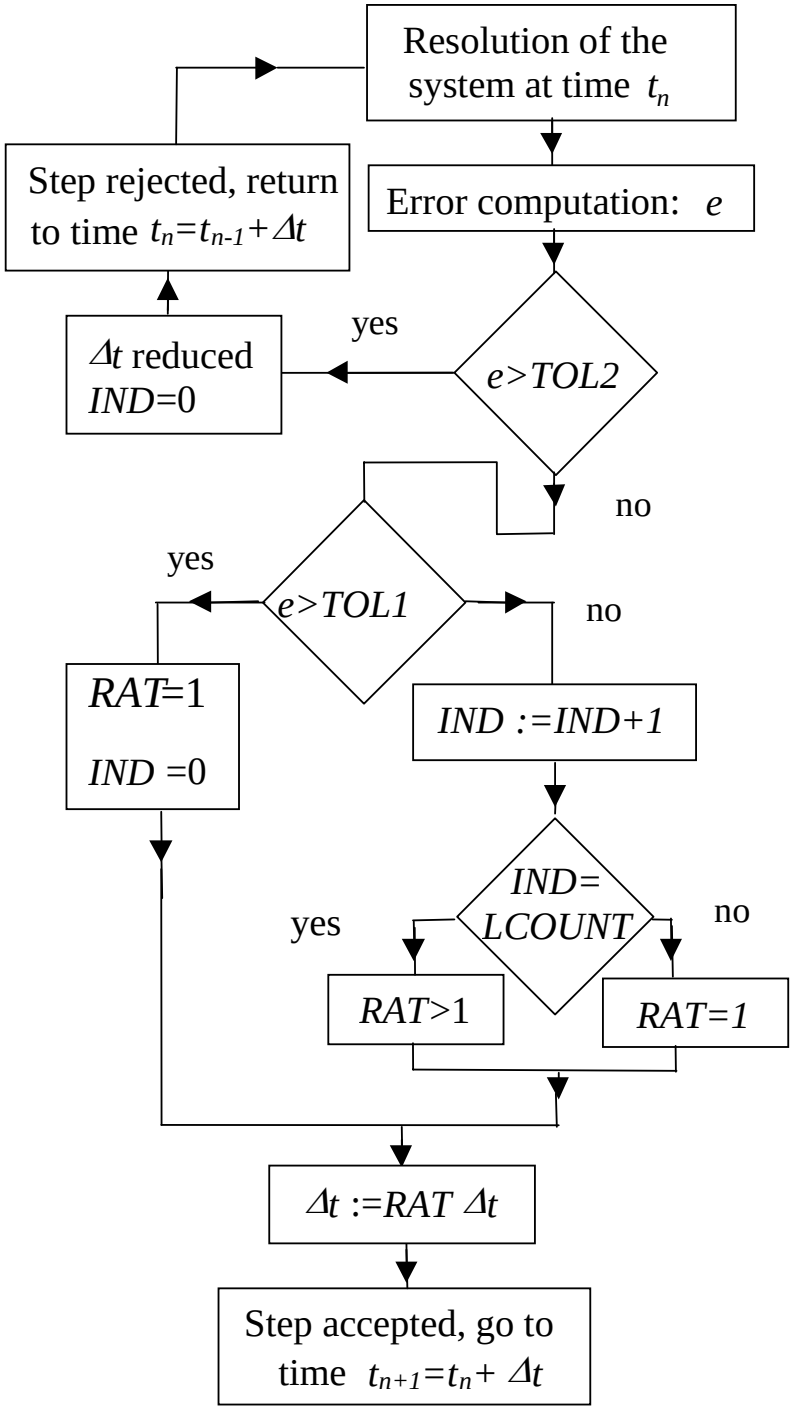
[13] SAMTECH. User Manuel of Samcef, v8.0. Liège, 1999.

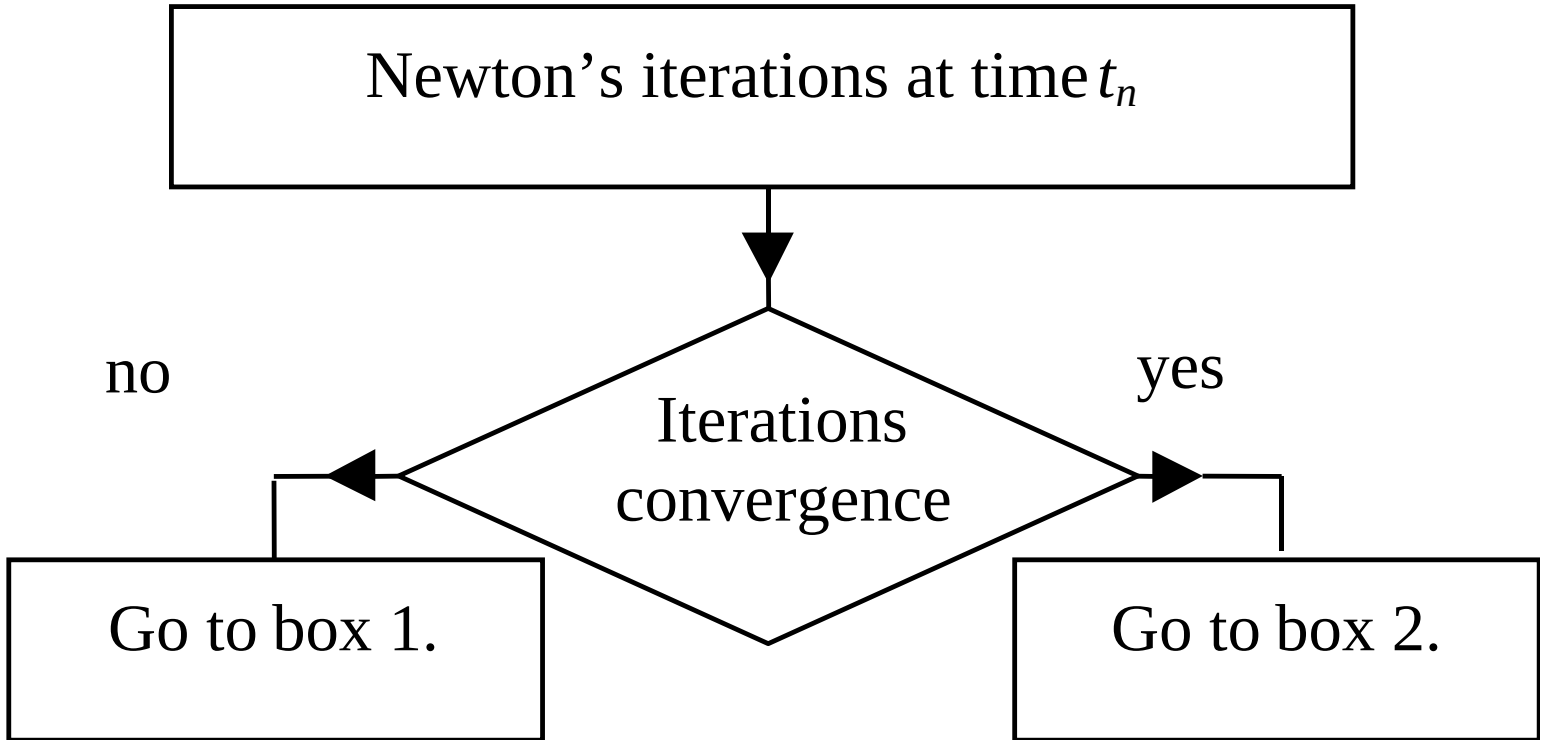
[14] Graillet D. Modélisation tridimensionnelle du contact entre structures à parois minces dans les phénomènes d'impacts et de mise à forme. PhD Thesis (in French). Liège,Belgium: Université de Liège, To appear.

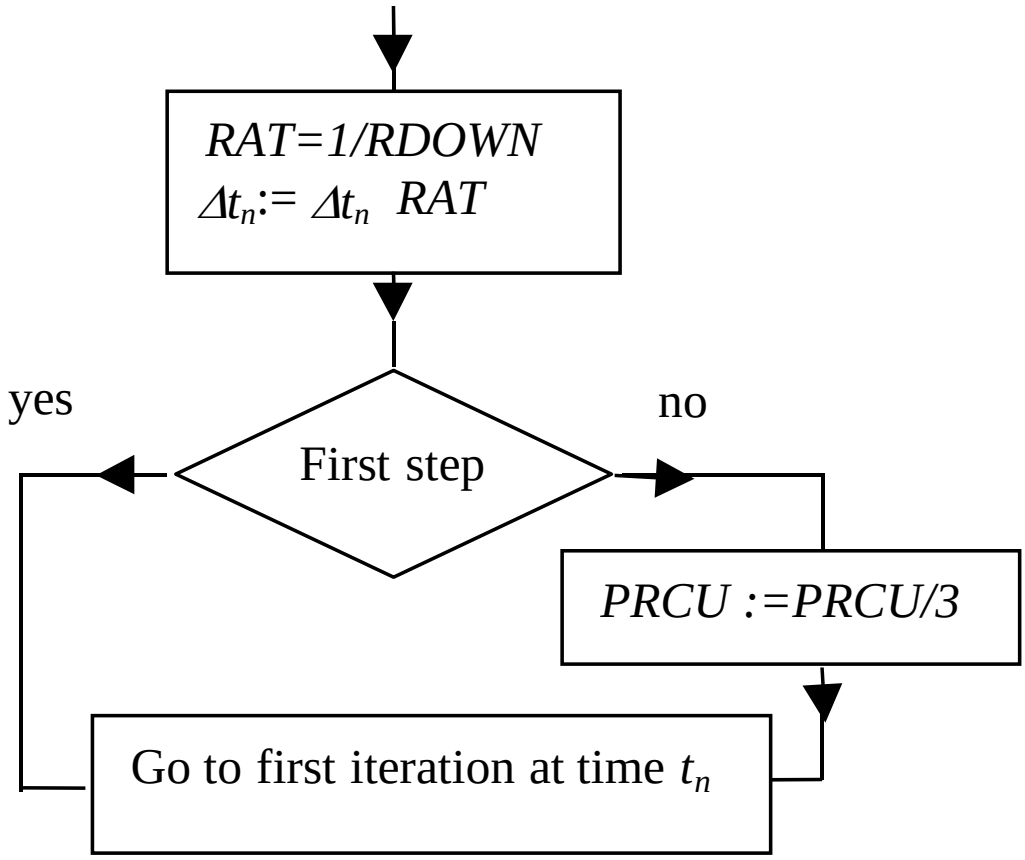
[15] Graillet D, Ponthot, JP. Efficient implicit schemes for the treatment of the contact between deformable bodies: Application to schock-absorber devices. IJCrash . 1999; 4 (3): 273-286.

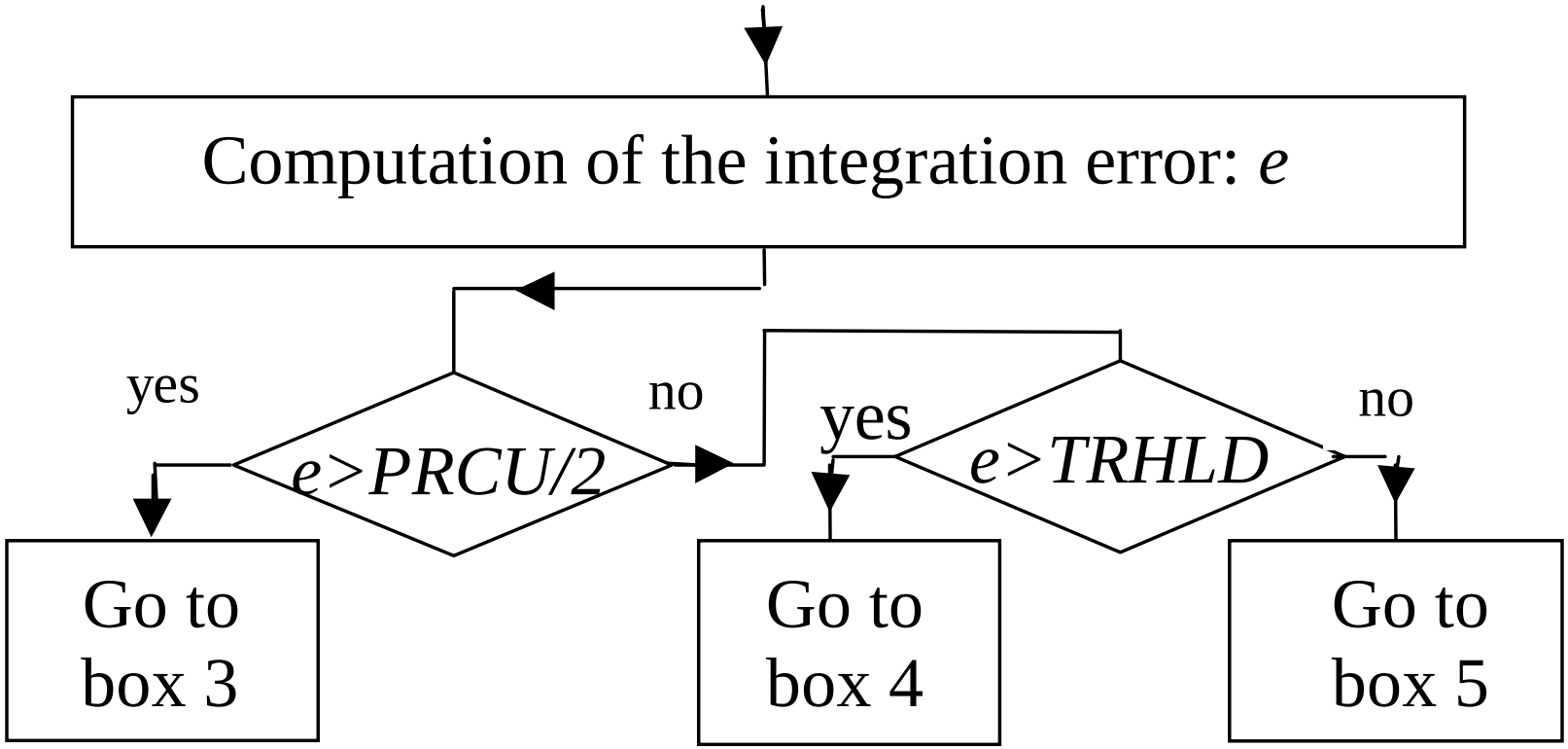
[16] Laursen TA. Formulation and treatment of frictional contact problems using finite elements. PhD Thesis. USA: Departement of mechanical engeneering, Stanford University. 1992.

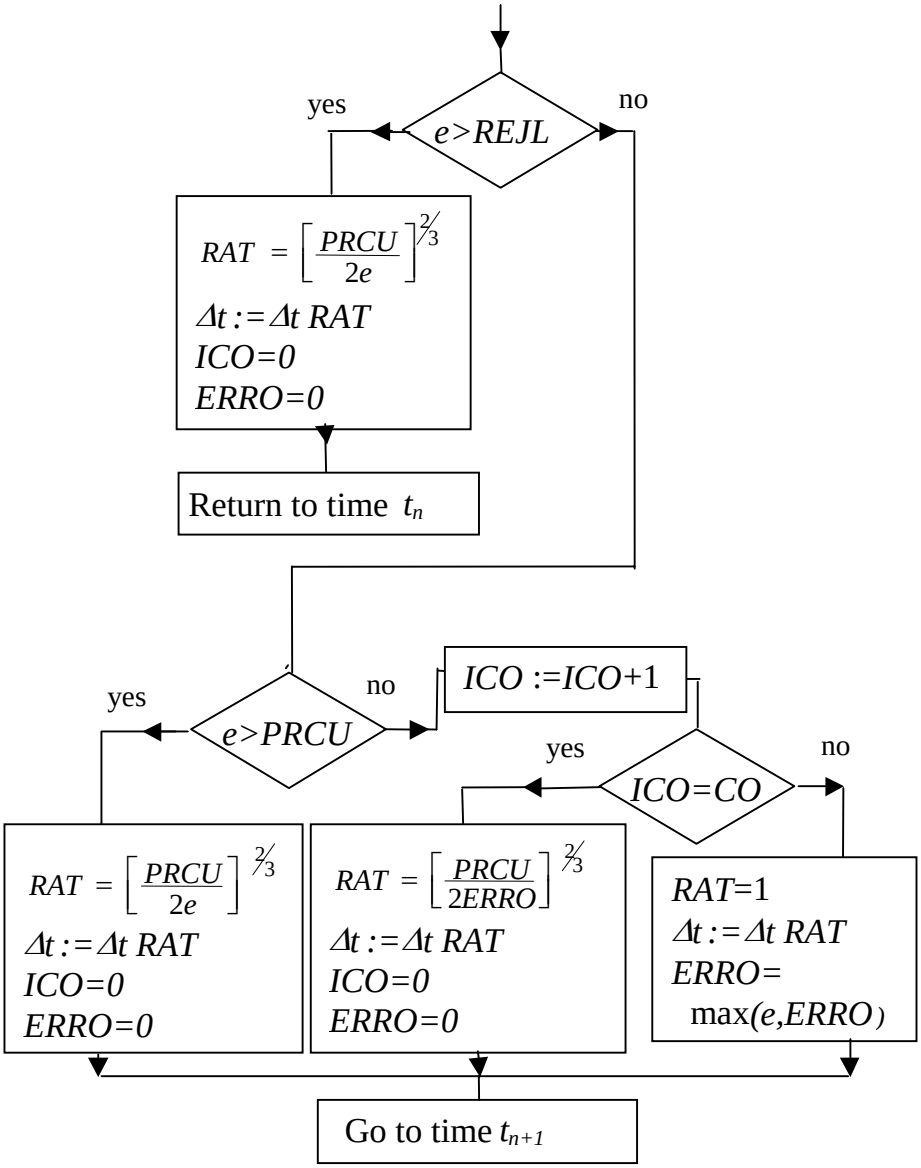

Resolution of the system at time t_n
Error computation : e
$e>PRCU$
yes
$\Delta t := \Delta t /2$
Step rejected, return to time $t_n=t_{n-1}+\Delta t$
no
$e>PRCU/2$
yes
no
$RAT=\left[\frac{PRCU}{2e}\right]^{1/3}$
$e<PRCU/16$
yes
no
$0.5<RAT<0.9$
$RAT=2$
$RAT=1$
$\Delta t :=RAT\ \Delta t$
Step accepted, go to time $t_{n+1}=t_n+\Delta t$

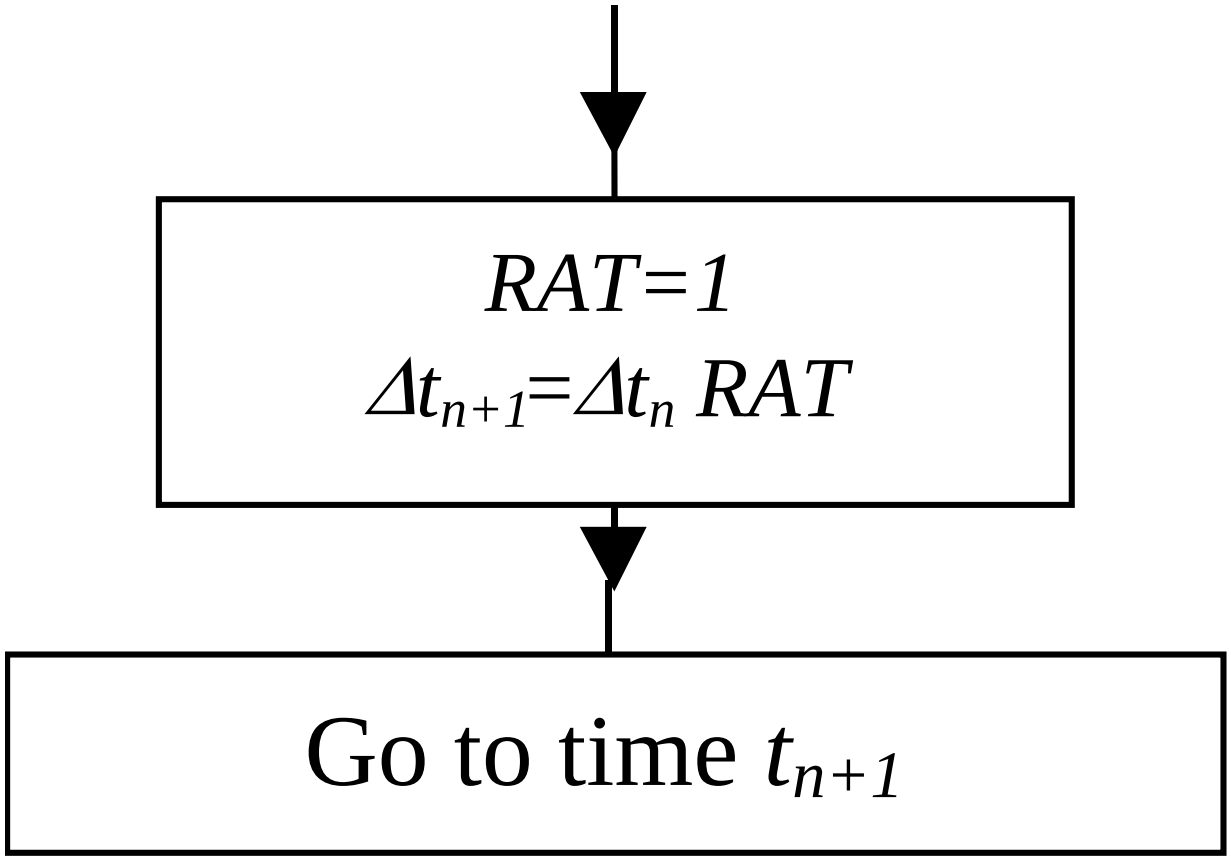

Resolution of the system at time t_n
Error computation: e
$e>TOL2$
yes
Δt reduced
IND=0
Step rejected, return to time $t_n=t_{n-1}+\Delta t$
no
$e>TOL1$
yes
no
RAT=1
IND =0
IND :=IND+1
IND= $LCOUNT$
yes
no
RAT>1
RAT=1
Δt :=RAT Δt
Step accepted, go to time $t_{n+1}=t_n+\Delta t$

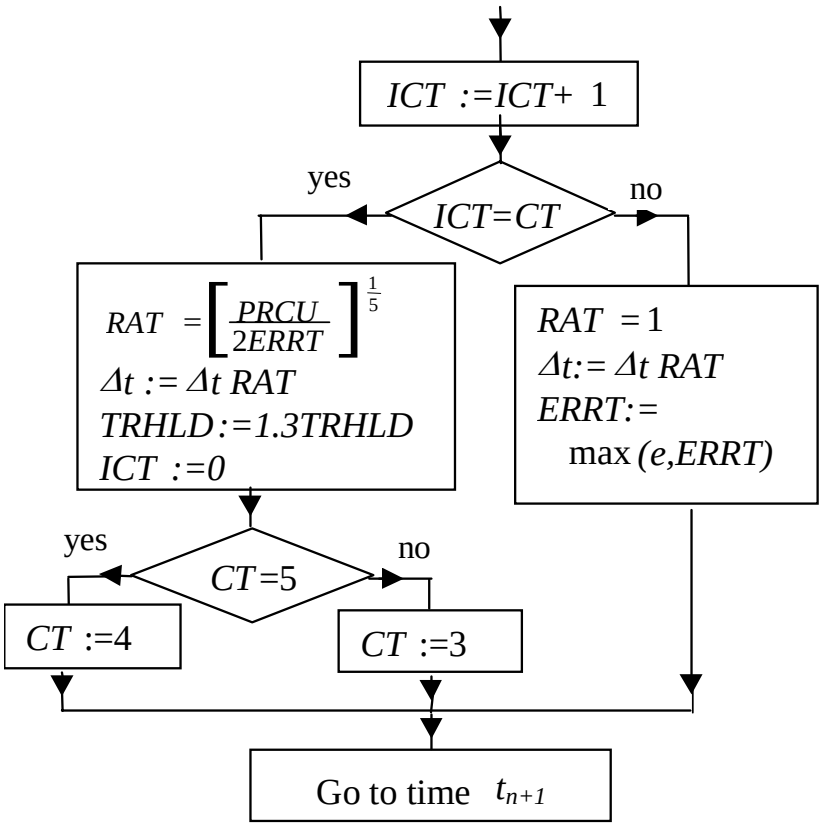

Newton’s iterations at time t_n
Iterations convergence
no
yes
Go to box 1.
Go to box 2.

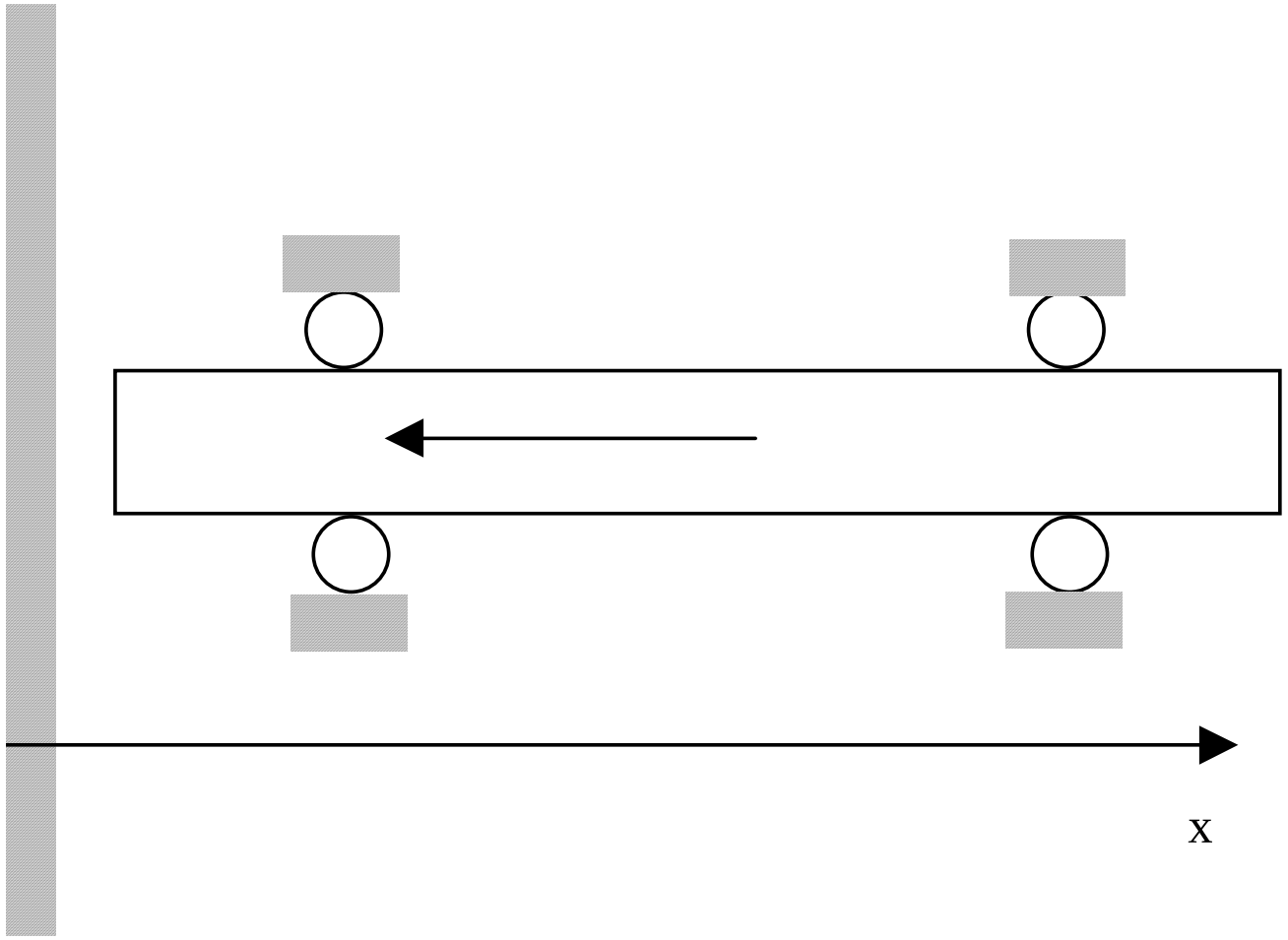

RAT=1/RDOWN
Δt_n:= Δt_n RAT
yes
First step
no
PRCU :=PRCU/3
Go to first iteration at time t_n

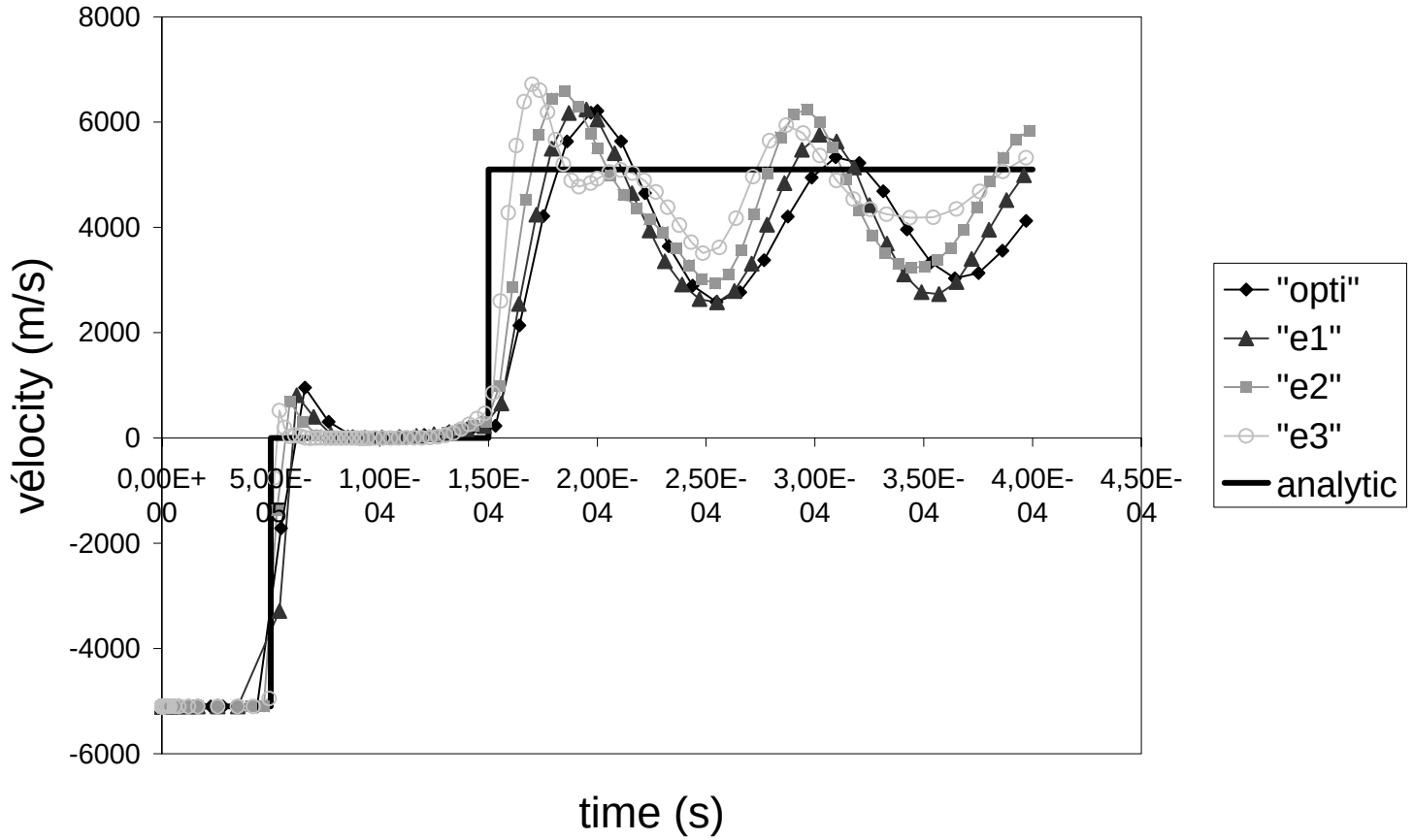

Computation of the integration error: e
yes
e>PRCU/2
no
yes
e>TRHLD
no
Go to box 3
Go to box 4
Go to box 5


yes
e>REJL
no
RAT = [PRCU / 2e]^(2/3)
Δt := Δt RAT
ICO=0
ERRO=0
Return to time t_n
yes
e>PRCU
no
ICO :=ICO+1
yes
ICO=CO
no
RAT = [PRCU / 2e]^(2/3)
Δt := Δt RAT
ICO=0
ERRO=0
RAT = [PRCU / 2ERRO]^(2/3)
Δt := Δt RAT
ICO=0
ERRO=0
RAT=1
Δt := Δt RAT
ERRO=
max(e,ERRO)
Go to time t_{n+1}

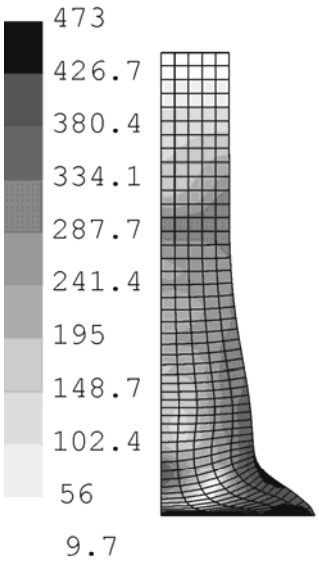

RAT=1
Δt_{n+1}=Δt_n RAT
Go to time t_{n+1}


ICT :=ICT+ 1
yes
ICT=CT
no
RAT = [PRCU / 2ERRT]^1/5
Δt := Δt RAT
TRHLD:=1.3TRHLD
ICT :=0
RAT =1
Δt:= Δt RAT
ERRT:=
max (e,ERRT)
yes
CT=5
no
CT :=4
CT :=3
Go to time t_{n+1}

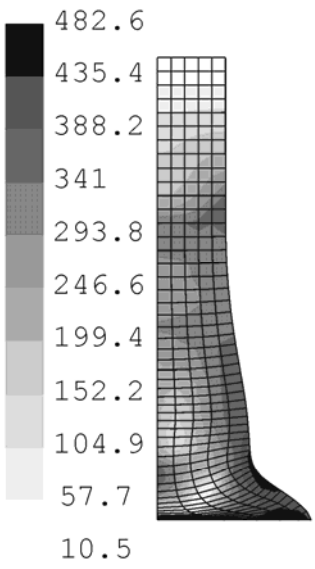

x


8000
6000
4000
2000
0
-2000
-4000
-6000
vélocity (m/s)
0,00E+00
5,00E-05
1,00E-04
1,50E-04
2,00E-04
2,50E-04
3,00E-04
3,50E-04
4,00E-04
4,50E-04
time (s)
"opti"
"e1"
"e2"
"e3"
analytic

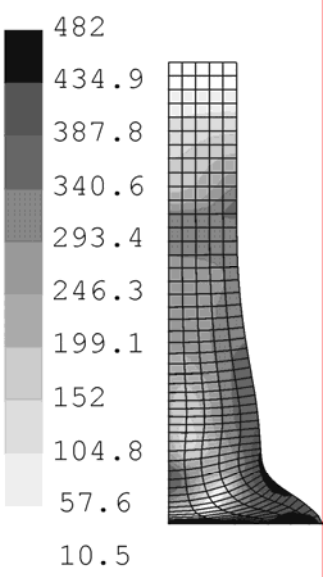
“opti”



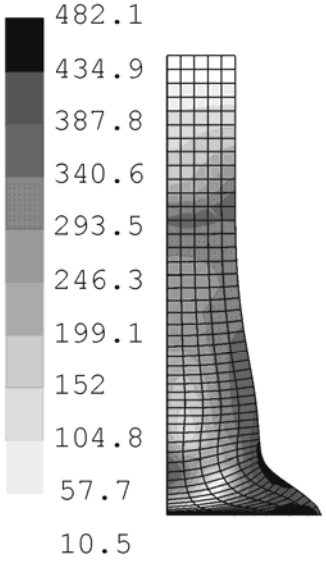
“e_1”



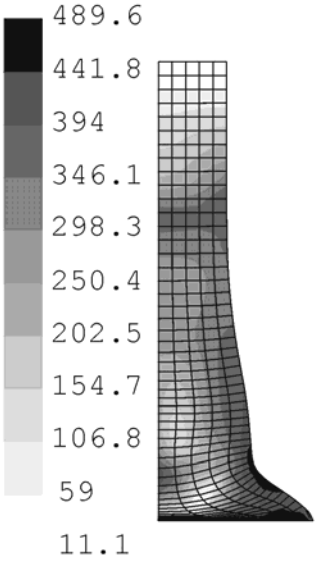
“e_2”



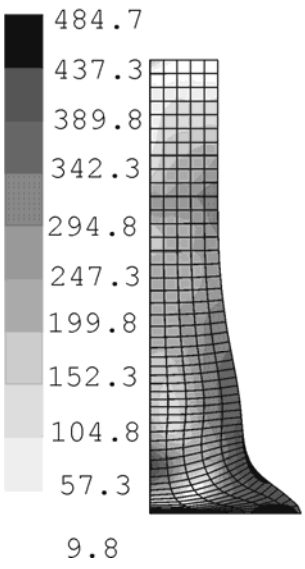
“e_3”

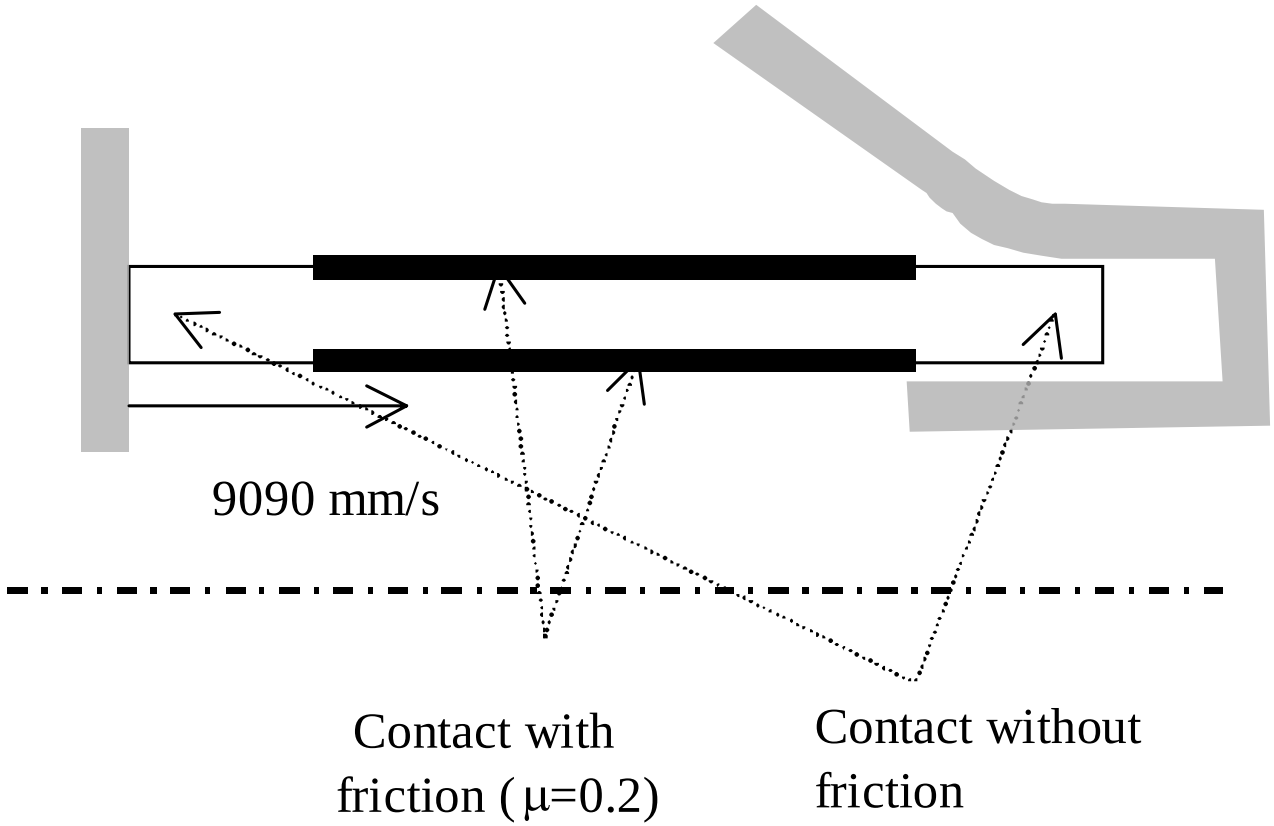


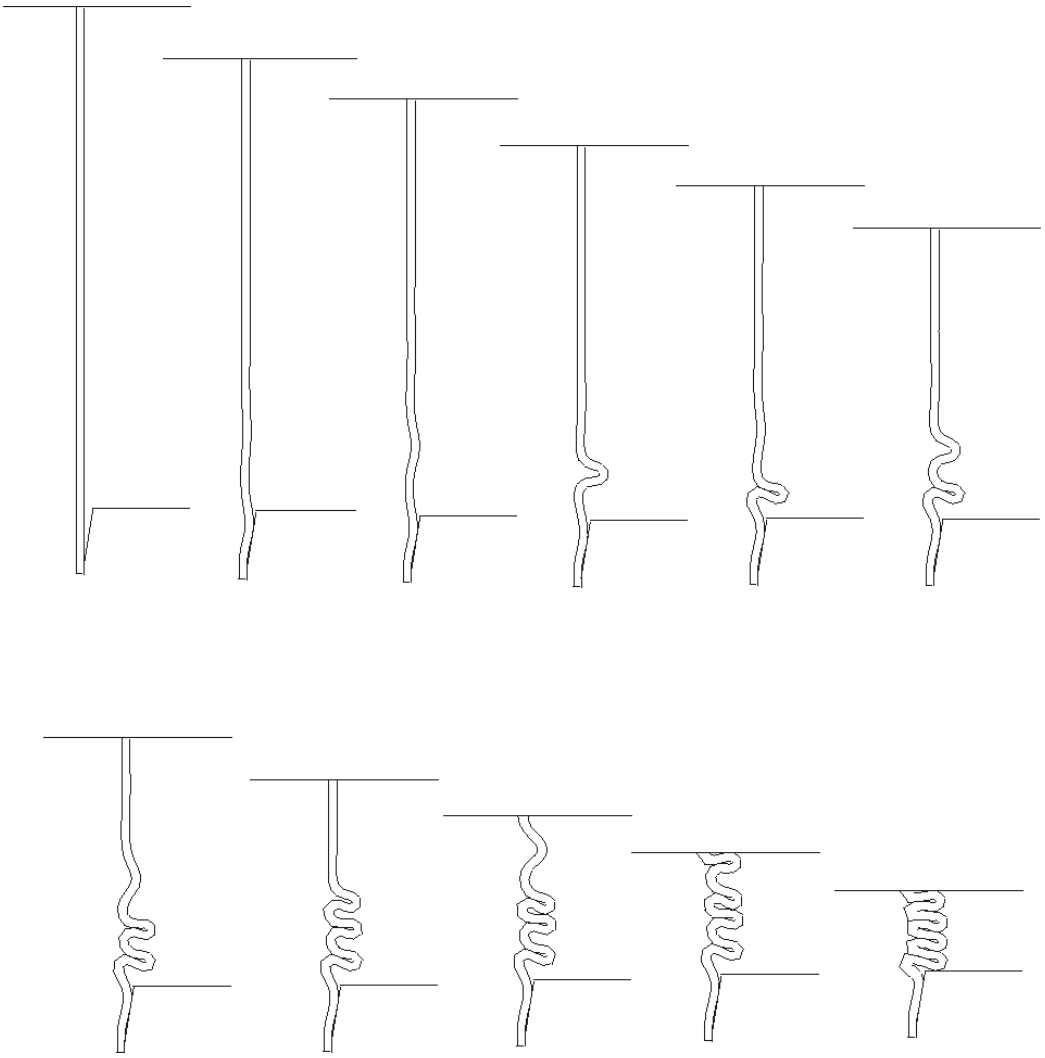
“reference”

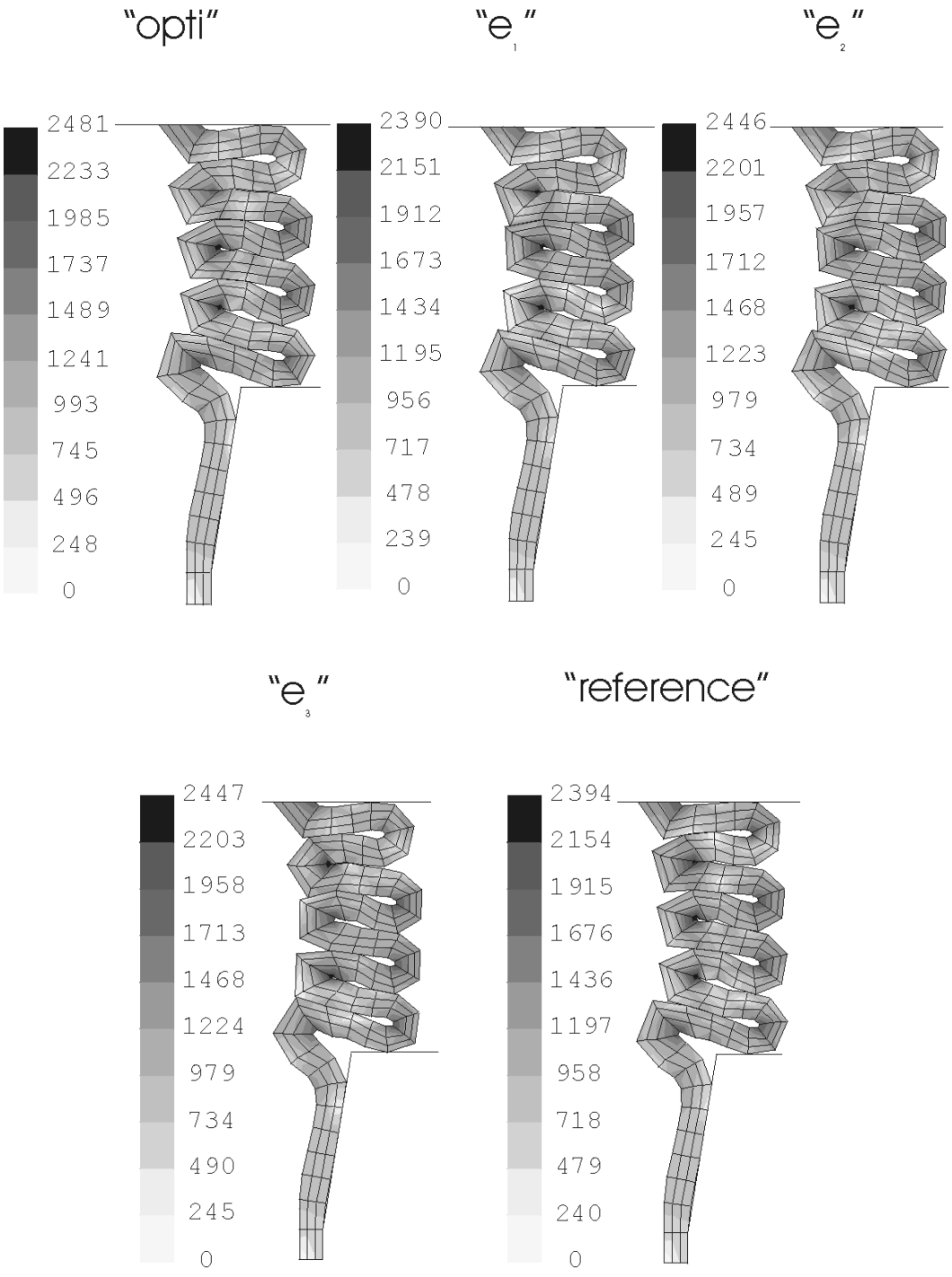


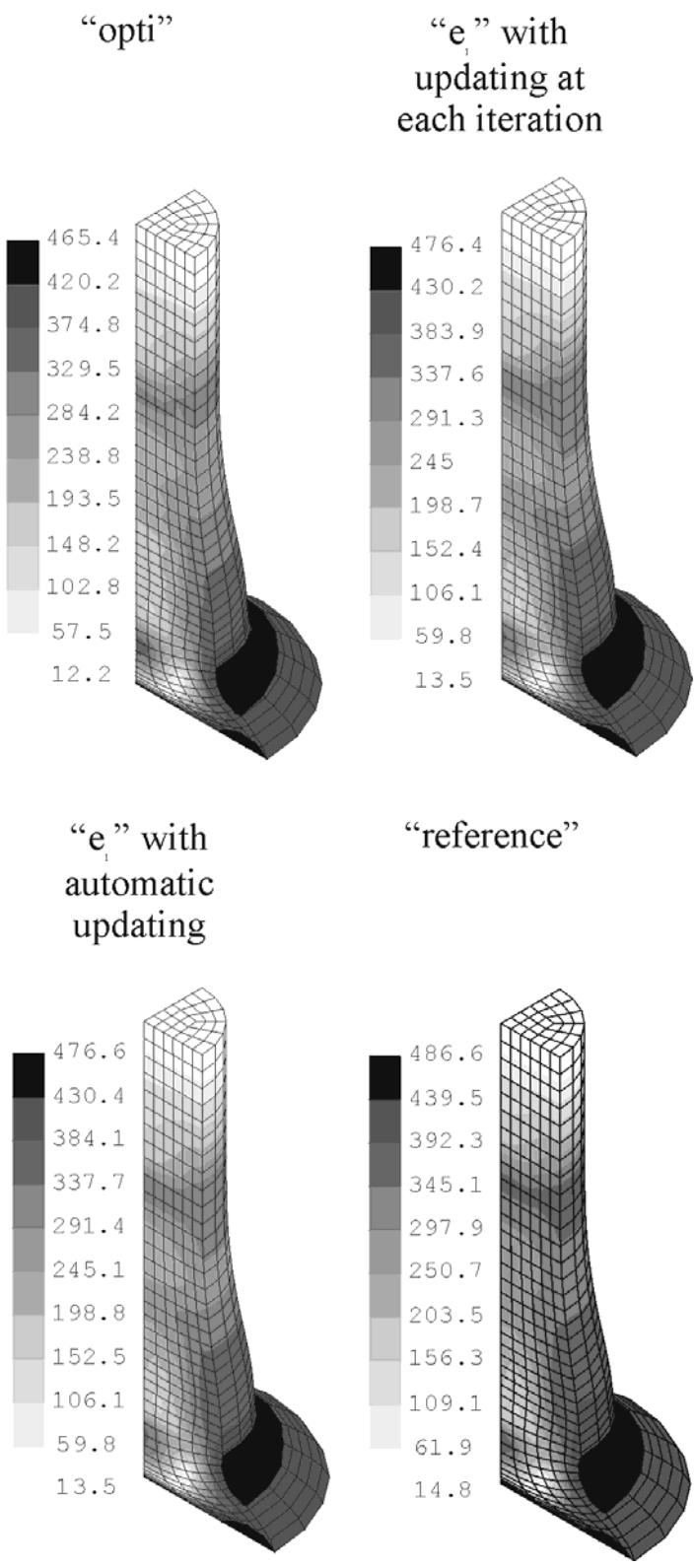
“explicit”

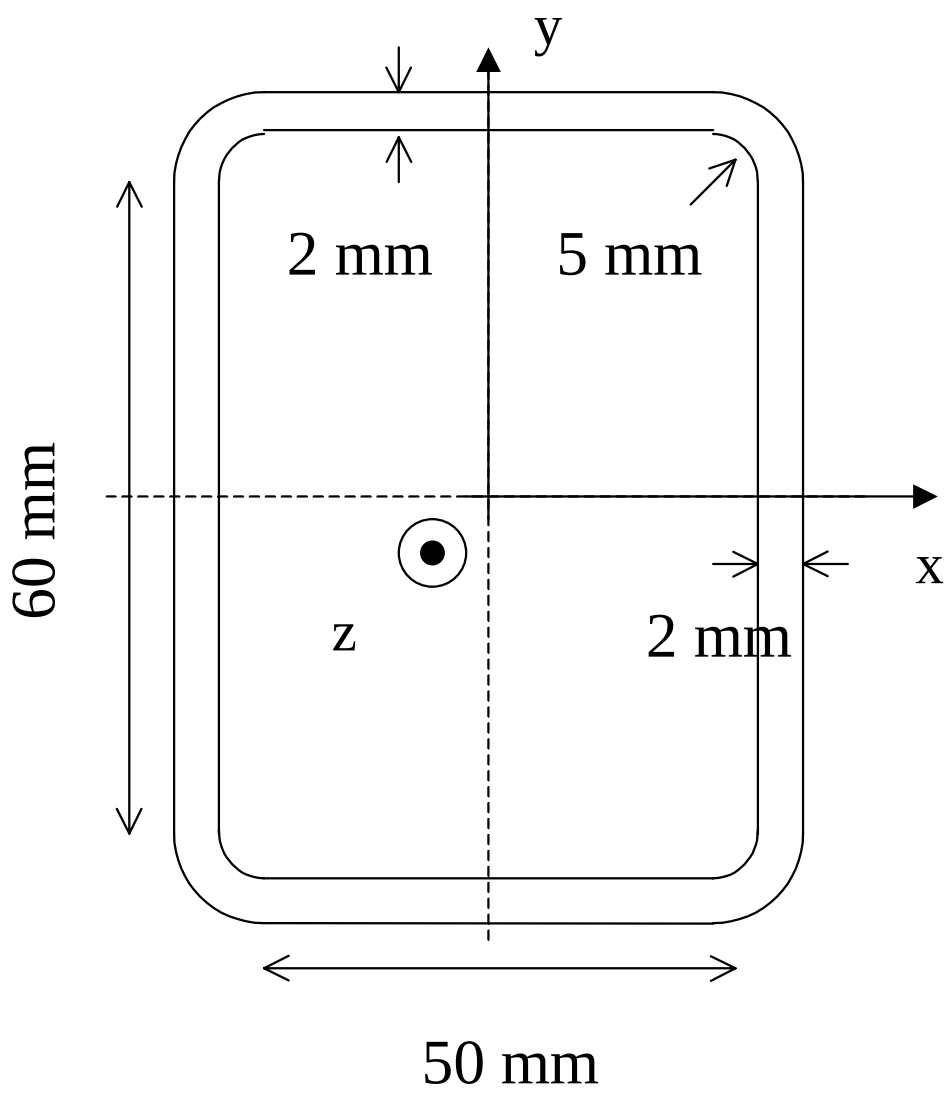


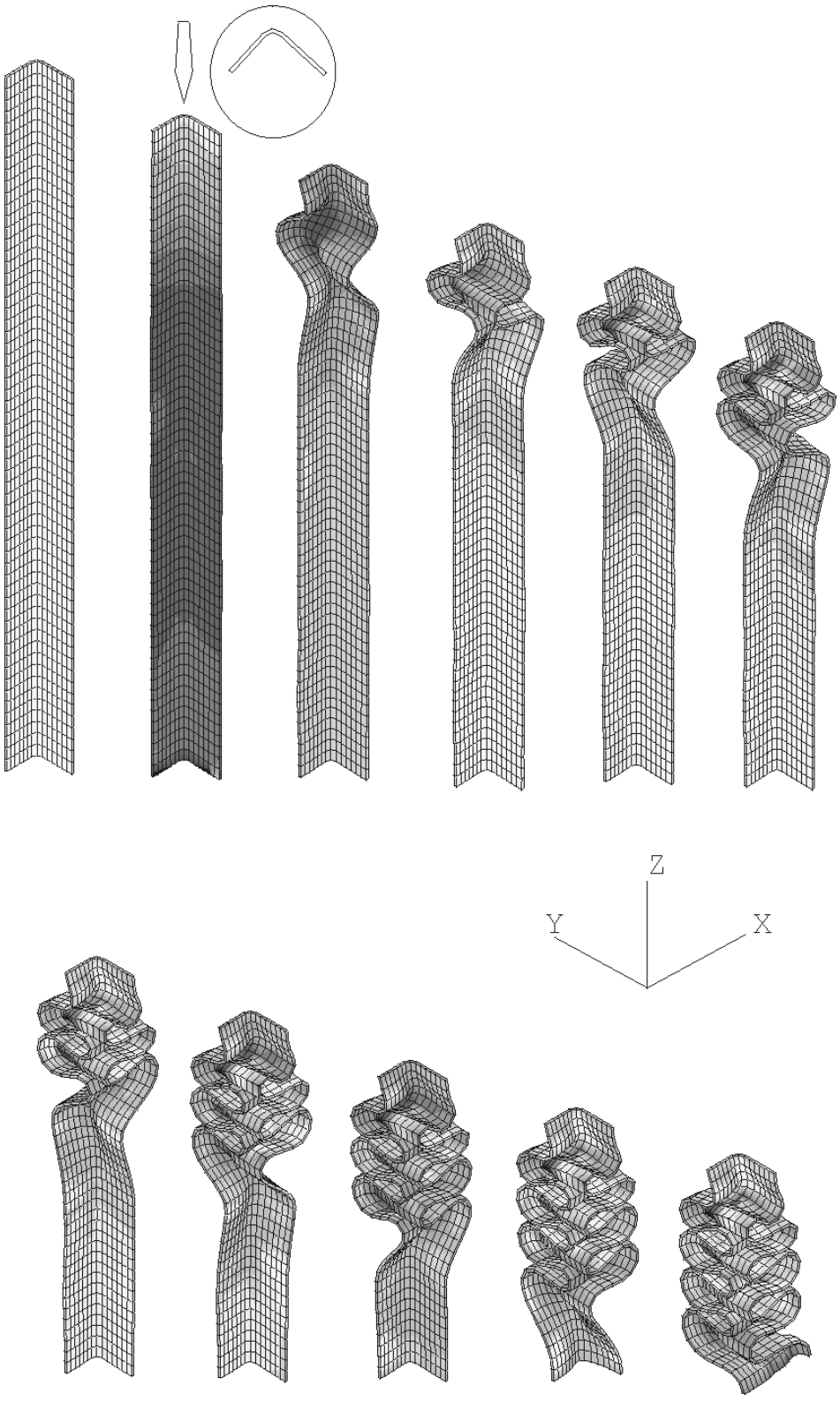

9090 mm/s
Contact with
friction (μ=0.2)
Contact without
friction



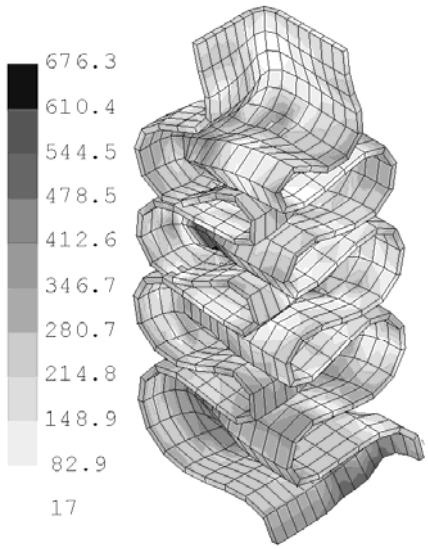

"opti"
2481
2233
1985
1737
1489
1241
993
745
496
248
0
"e1"
2390
2151
1912
1673
1434
1195
956
717
478
239
0
"e2"
2446
2201
1957
1712
1468
1223
979
734
489
245
0
"e3"
2447
2203
1958
1713
1468
1224
979
734
490
245
0
"reference"
2394
2154
1915
1676
1436
1197
958
718
479
240
0


"opti"
465.4
420.2
374.8
329.5
284.2
238.8
193.5
148.2
102.8
57.5
12.2
"e1" with updating at each iteration
476.4
430.2
383.9
337.6
291.3
245
198.7
152.4
106.1
59.8
13.5
"e1" with automatic updating
476.6
430.4
384.1
337.7
291.4
245.1
198.8
152.5
106.1
59.8
13.5
"reference"
486.6
439.5
392.3
345.1
297.9
250.7
203.5
156.3
109.1
61.9
14.8

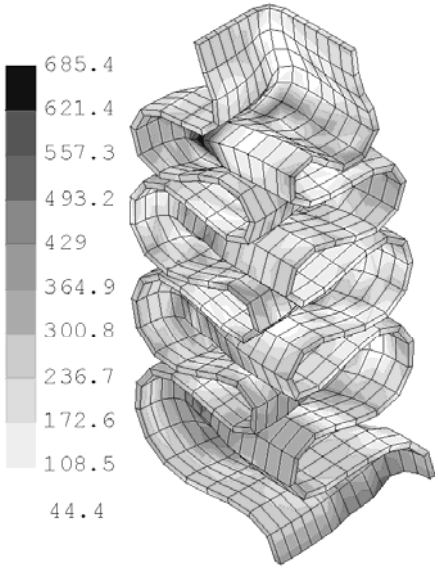

y
2 mm
5 mm
60 mm
x
z
2 mm
50 mm


Z
Y
X

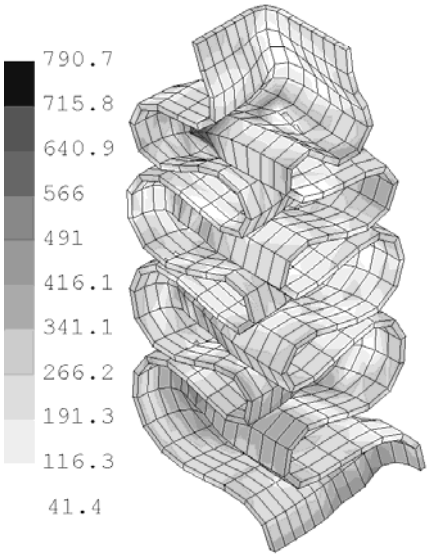
"opti"



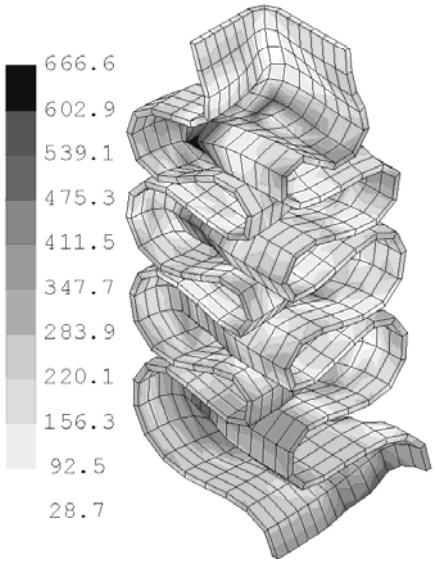
"e_1" with updating at each iteration

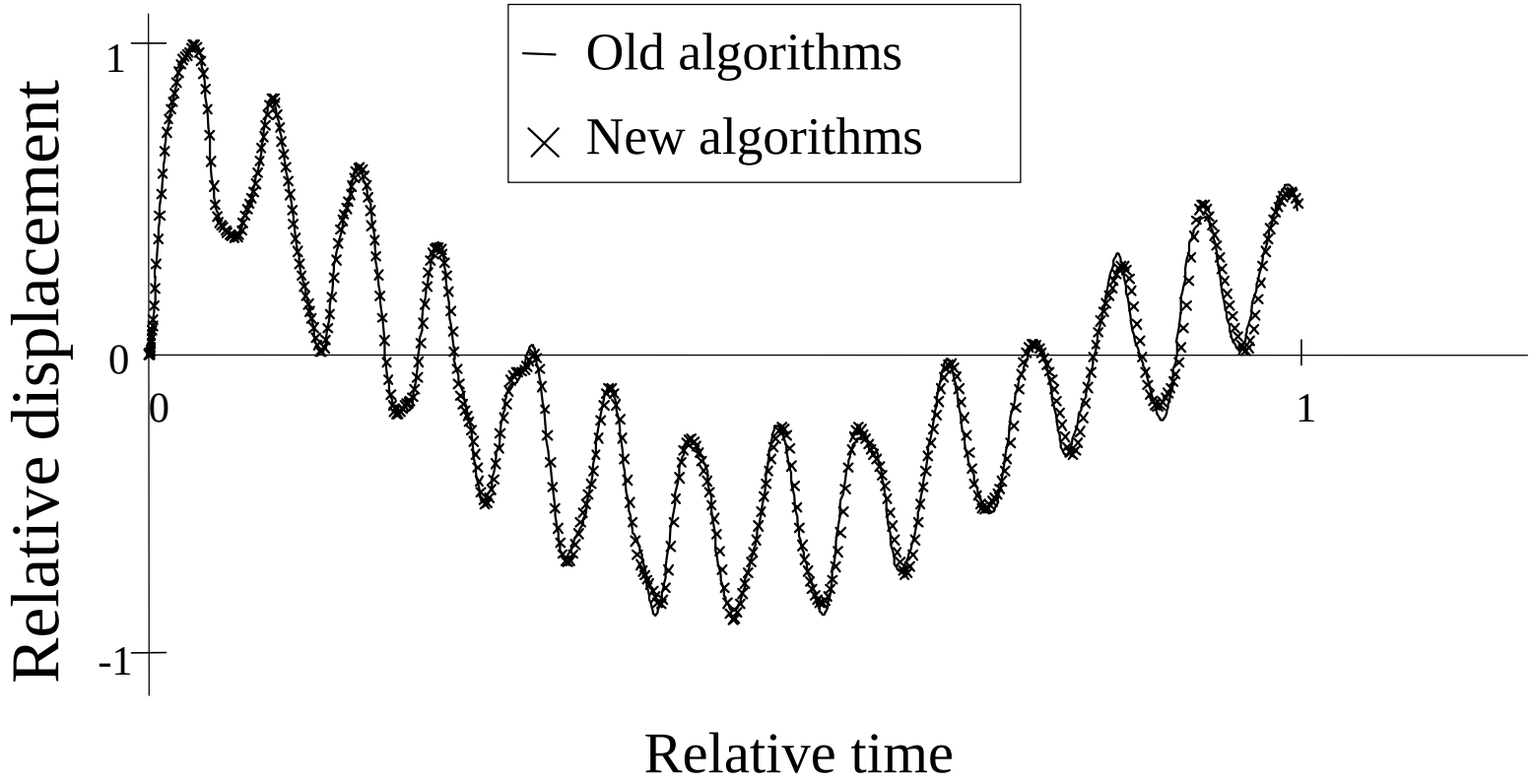


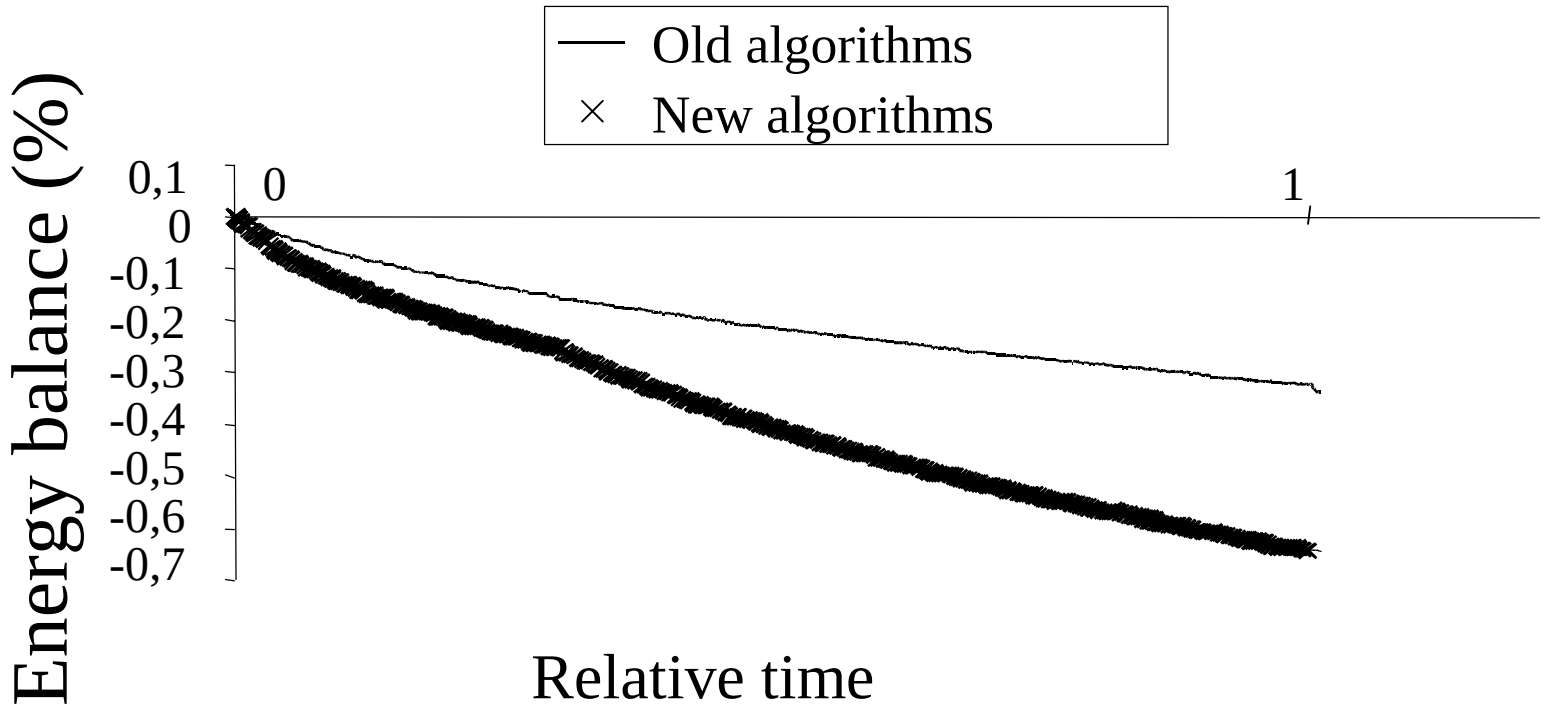
"e_1" with automatic updating

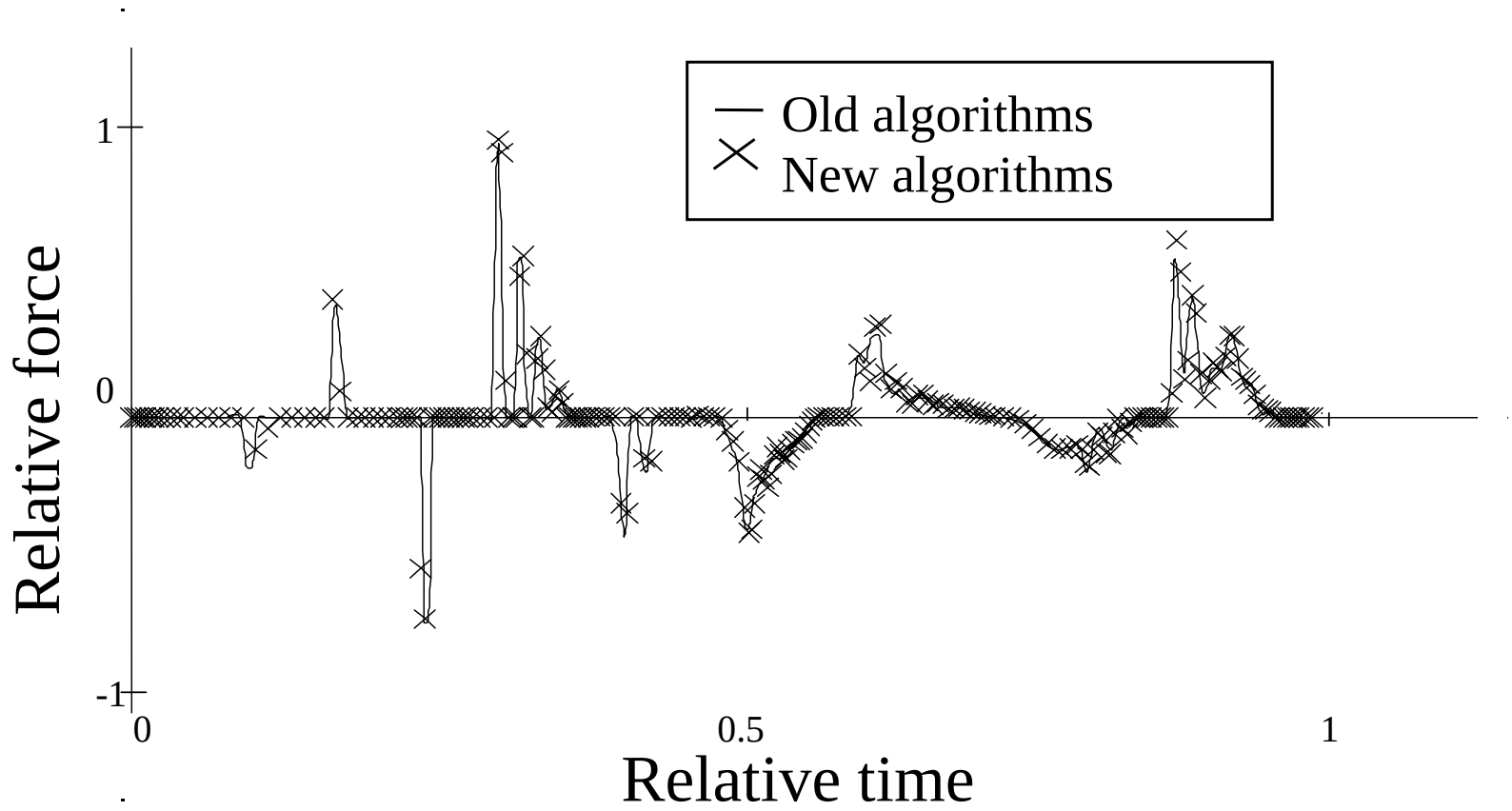


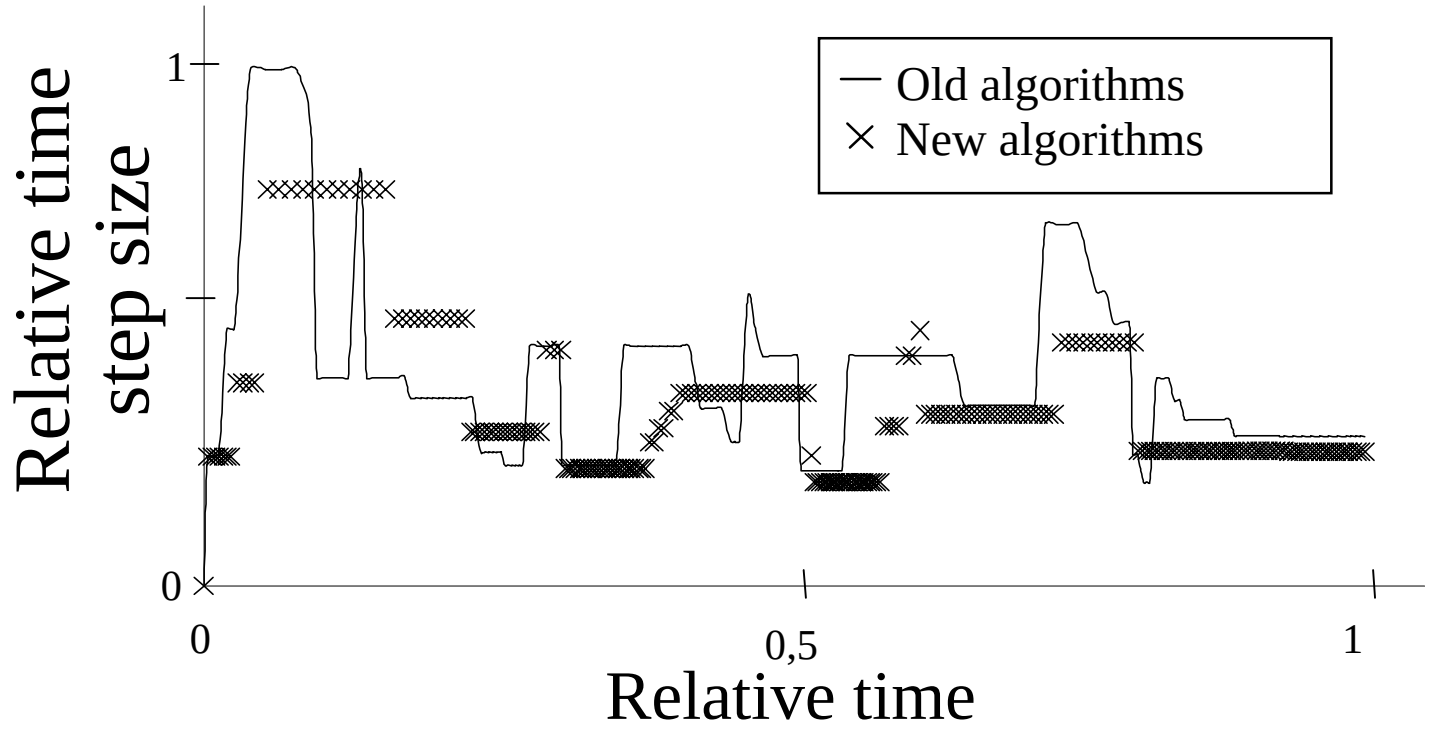
"reference"




Old algorithms
New algorithms
Relative displacement
1
0
-1
0
1
Relative time


Old algorithms
New algorithms
Energy balance (%)
0,1
0
-0,1
-0,2
-0,3
-0,4
-0,5
-0,6
-0,7
0
1
Relative time


Old algorithms
New algorithms
Relative force
1
0
-1
0
0.5
1
Relative time


Relative time step size
1
0
0
0,5
1
Relative time
Old algorithms
New algorithms

Height	Length	Initial distance to matrix	Density	Young modulus	Poisson coefficient	Initial velocity
d=40 mm	l=247.65 mm	d_i=0.25 mm	ρ = 7895 kg/m^3	E=206.84 10^9 kg/m²	ν=0.0	$\dot{x}_0$ =5 m/s

Table 1: Properties of elastic bar

Problem	e_1	e_2	e_3	"opti"	"reference"
Problem 1	330	380	430	28	(analytic)
Problem 2	467	528	627	418	2292
Problem 3	7745	6538	12670	25500	27680

Table 2: Computation cost (ms) comparison for the first three problem

Diameter	Length	Density	Young modulus	Poisson coefficient	Yield stress	Hardening parameter	Initial velocity
d=6.4 mm	l=32.4 mm	ρ = 8930 kg/m^3	E=117 10^9 kg/m²	ν=0.35	σ_0=4 10^8 N/m²	h=1 10^8 N/m²	$\dot{x}_0$ =227 m/s

Table 3: Properties of Taylor's bar

Internal diameter	External diameter	Length	Density	Young modulus	Poisson coefficient	Yield stress	Hardening parameter	Matrix velocity
d_i=27 mm	d_e=31.17 mm	l=180 mm	ρ = 7850 kg/m^3	E=2.1 10^{11} kg/m^2	ν=0.3	σ_0=700 N/mm^2	h=808 N/mm^2	$\dot{x}_0$ =9.09 m/s

Table 4: Properties of buckling cylinder

Problem	e_1 with updating at each iteration	e_1 with automatic updating	"opti"	"reference"
Problem 4	6.35	4.43	4.7	12.6
Problem 5	264	254	395	917

Table 5: Computation cost (min) comparison for problems 4 and 5

Length	Density	Young modulus	Poisson coefficient	Yield stress	Hardening parameter	Matrix velocity
l=600 mm	ρ = 8900 kg/m^3	E=2 10^{11} kg/m^2	ν=0.3	σ_0=200 N/mm^2	h=630 N/mm^2	$\dot{z}_0$ =25 m/s

Table 6: Properties of buckling 3D-bar

Table 5: Computation cost (min) comparison for problems 4 and 5

Table 6: Properties of buckling 3D-bar