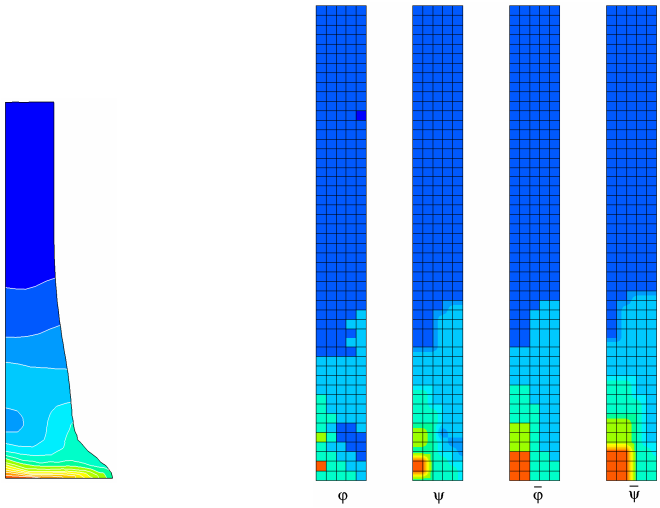
JRC Scientific and Technical Reports

Spatial Time Step Partitioning in Explicit Fast Transient Dynamics

F. Casadei, J.P. Halleux



EUR 23062 EN - 2008

JRC
EUROPEAN COMMISSION

ipsc
Institute for the Protection and Security of the Citizen

The Institute for the Protection and Security of the Citizen provides research-based, systems-oriented support to EU policies so as to protect the citizen against economic and technological risk. The Institute maintains and develops its expertise and networks in information, communication, space and engineering technologies in support of its mission. The strong cross-fertilisation between its nuclear and non-nuclear activities strengthens the expertise it can bring to the benefit of customers in both domains.

European Commission
Joint Research Centre
Institute for the Protection and Security of the Citizen

Contact information
Address: Folco Casadei
E-mail: folco.casadei@jrc.it
Tel.: +39 033278 9563
Fax: +39 033278 9049

http://ipsc.jrc.ec.europa.eu/
http://www.jrc.ec.europa.eu/

JRC 42581

EUR 23062 EN

ISSN 1018-5593

Luxembourg: Office for Official Publications of the European Communities

Printed in Italy

Foreword

This report is a completely revised and re-organized edition of the research results that were preliminarily presented in reference [1]. The spatial partitioning algorithms are given here in full detail and with extreme precision, but in a much more condensed form than in [1], since all unnecessary implementation details and preliminary tests have been removed.

Readers interested in an even more compact presentation should consult the papers [2] and [3], which have been submitted for publication in a leading scientific journal in the field, or the conference paper [4].

[1] J.P. Halleux, F. Casadei: "*Spatial Time Step Partitioning in EUROPLEXUS*", JRC EUR Report N. 22464 EN, 2006.

[2] J.P. Halleux, F. Casadei: "*An algorithm for Spatial Partitioning in Explicit Time Integration — Part I - Basics*", submitted for publication.

[3] F. Casadei, J.P. Halleux: "*An algorithm for Spatial Partitioning in Explicit Time Integration — Part II - Treatment of Boundary Conditions and Extension to ALE*", submitted for publication.

[4] F. Casadei, J.P. Halleux: "*Explicit Time Step Spatial Partitioning in Nonlinear Transient Dynamics*", ECCOMAS Thematic Conference on Computational Methods in Structural Dynamics and Earthquake Engineering, COMPDYN 2007, Rethymno, Crete, June 13-16, 2007.

Spatial Time Step Partitioning in Explicit Fast Transient Dynamics

F. Casadei and J.P. Halleux

European Commission,
Institute for the Protection and Security of the Citizen
Joint Research Centre, 21020 Ispra, Italy

Abstract

This report presents a technique for spatial partitioning of the time increment in the explicit central-difference time integration scheme commonly used for finite-element modeling of fast transient dynamic phenomena. The time increment varies not only in time, as is usual to account for mesh distortion and evolution of material properties, but also in space—at the finite element level—following local stability limitations rather than global ones.

This may lead to substantial savings of computer time whenever the material properties that govern wave propagation speed and/or the mesh size are largely non-uniform in the numerical model, as is typical of many large industrial applications, especially in 3D, and even more so in the presence of fluid-structure interactions.

The proposed partitioning algorithm, which is completely automatic and does not require any specific input data, may be applied in principle to all types of elements and material models. As shown by several numerical examples, it preserves the outstanding numerical properties—i.e. the renowned accuracy and robustness—of the classical uniform-step explicit time integration scheme, of which it may be considered a powerful generalization.

Once fully implemented and validated in a general explicit computer code, the present technique has the potential for freeing the engineer from the main limitation of explicit analysis, usually related through stability requirements to the size of the smallest element. In fact, the computational mesh may be locally refined virtually at will without the usual prohibitive effects on computational costs. This might open the way to applications which are simply out of reach with the classical algorithms.

The present document is subdivided in 2 Parts. Part I introduces the basic spatial partitioning technique within a Lagrangian formulation, with some simple academic examples. Part II presents the treatment of boundary conditions, the extension to fluids via an Arbitrary Lagrangian Eulerian formulation and some more realistic applications.

Spatial Time Step Partitioning in Explicit Fast Transient Dynamics — Part I - Basics

1. Introduction

Explicit finite element codes have been developed, validated and used over more than three decades to model fast transient dynamic phenomena, ranging from explosions to impacts and to crashes. They are now routinely applied for simulations in such broad areas as nuclear safety, transportation, pipelines, marine and offshore, or building vulnerability under terrorist attacks, to name just a few.

Typically the numerical models employed are strongly nonlinear both in the geometry (large displacements, large rotations, large strains, contacts) and in material behavior (plasticity, viscoplasticity, damage, failure, etc). Explicit time integration algorithms such as the well-known central difference scheme have been traditionally preferred by fast dynamics code developers with respect to implicit methods, because they are comparatively much simpler to implement. They do not require numerical iterations—with the associated convergence problems—and at the same time they exhibit good accuracy and outstanding robustness.

The main—and perhaps the only—drawback of these methods is their conditional stability. Very small time increments Δt must be used to obtain stable solutions, and the number of time steps for a typical simulation is often huge (numbers up to the order of millions are not unusual) despite the short duration of the fast transient phenomena of interest, a few milliseconds to a fraction of a second.

Common practice in explicit codes is to use a time increment $\Delta t(t)$ variable in time—this is in fact quite simple to implement—but uniform in space. In other words, all elements in the numerical model use the same time increment, which is the global minimum value over the entire mesh, i.e. the value dictated by the so-called “critical” element.

In large industrial applications the numerical models tend to contain a broad spectrum of finite element sizes, since the mesh is typically refined locally in zones of special interest to the engineer, or where sharp non-linear phenomena are expected. As a consequence, the critical stability step which drives the explicit time integration scheme used in the code may vary by orders of magnitude over the model, when passing from the finer-meshed to the coarser-meshed zones.

Using the same time increment—although variable in time—for the whole mesh may therefore be relatively inefficient, since the smallest element drives the entire computation. Elements which are much larger than the critical one, and which often represent the majority of the model, are in fact integrated in time with an increment which is sometimes orders of magnitude smaller than the value that would be allowed by local stability. This results in a waste of computer time for a given simulation.

A remedy to this situation consists therefore in building up a so-called spatial time step partitioning algorithm, i.e. a generalization of the basic time integration scheme using a time step $\Delta t(\underline{x}, t)$ variable not only in time (t), but also in space ($\underline{x}$). This means that a different time increment is used in general when dealing with each finite element—and with each node—of the discrete numerical model. Clearly the interest of such a technique is exclusively related to optimizing the computational efficiency of the numerical simulations.

Techniques of this kind have been proposed in the literature since FE pioneering times, see e.g. [2-3], under the name of mixed methods for time integration, of (spatial) partitioning, or of sub-cycling techniques. The latter term originates from the fact that, roughly speaking, these techniques compute several “cycles” of time integration over the smaller elements for each integration step over the larger ones. One could say that smaller (or more critical) elements are time-integrated “more often” than larger (or less critical) ones.

More recently a variety of computational strategies have appeared in the literature under the name of domain decomposition methods, see e.g. the work of Combescure *et al.* [4-6]. The numerical model is subdivided into a (usually small) number of sub-domains in order to optimize the calculation speed and/or accuracy. A different time step $\Delta t(i, t)$ is then typically used in each sub-domain i—although the increment is usually uniform in space for the whole sub-domain—and therefore these techniques might perhaps be viewed as a sort of (coarse-grain) partitioning. However, there are some important differences:

- On one hand, domain decomposition methods are more general than spatial partitioning, e.g. because they allow non-conforming meshes along the interfaces between sub-domains, the use of different time integration schemes as appropriate for each sub-domain (e.g. explicit/implicit coupling), and even specialized treatments of some sub-domains, for example by local modal reduction techniques where appropriate in order to further improve efficiency, see e.g. Faucher and Combescure, references [7-8].
- On the other hand, domain decomposition methods are more complicated to use than spatial partitioning. The subdivision into sub-domains is typically left to the code user, who must also define the interfaces between sub-domains. The spatial partitioning process, instead, may be completely automatized, as will be explained in the description of the algorithm proposed in this paper. Also from the informatics viewpoint, the implementation of partitioning in a standard explicit code architecture requires far less modifications than implementing domain decomposition.

From the historical viewpoint, it is noteworthy that the basic spatial partitioning technique to be described below was formulated and implemented as a prototype in an explicit finite element code

(PLEXIS-3C, see reference [9]) by the first author of this paper as early as in 1985. The prototype proved to be quite performing on some academic problems, but the absence at that time of an adequate and full treatment of what could be called “accessories”, notably the large variety of boundary conditions (links) which may occur in fast transient applications made that speed ups which could effectively be gained in realistic industrial simulations were often very modest if not irrelevant.

In fact, the original formulation and prototype implementation dealt with boundary conditions in a straightforward but potentially inefficient way. Simply, any nodes subjected to boundary conditions were associated with the lowest partition level—see below for a precise definition of these terms. Such a simplified treatment may indeed be sufficient for academic-like examples such as the ones presented at the end of this paper, but in applications involving a lot of specific boundary conditions the benefits of partitioning were lost.

However, recent work on the treatment of the boundary condition models in the explicit code EUROPLEXUS (a direct descendent of PLEXIS-3C) and the introduction of a more modern informatics data structure has opened the way to an improved version of the partitioning technique, to be described below, which takes into account boundary conditions in a less conservative manner than previously, and which almost avoids the associated efficiency degradation.

EUROPLEXUS is a general finite-element computer program for the fast transient analysis of fluid-structure systems subjected to transient dynamic loading, which is being jointly developed by the French Commissariat à l’Energie Atomique (CEA Saclay) and by the Joint Research Centre of the European Commission (JRC Ispra).

This paper is organized as follows. Section 2 recalls the standard, non-partitioned explicit time integration scheme which is at the base of the proposed method. Section 3 describes the partitioning technique, considering for simplicity a Lagrangian description suitable for the treatment of purely structural applications and using a simplified, direct treatment of boundary conditions of trivial (uncoupled) types, namely those involving just one node at a time, such as blockages. Finally, section 4 presents some preliminary test examples of academic nature that illustrate the potential of the proposed method in terms of CPU efficiency gains.

Part II of this report describes the full treatment of (coupled) boundary conditions by a Lagrange multipliers method in the algorithm with partitioning, including both permanent and non-permanent conditions, such as contacts. Then it shows the extension of the partition to an Arbitrary Lagrangian Eulerian (ALE) formulation, suitable for the treatment of fluid and fluid-structure interaction problems. Part II also presents some more realistic applications.

All the calculations presented in this work were performed by the EUROPLEXUS code on a Pentium 4 PC with a 3.0 GHz processor and 1GB of RAM.

2. Explicit time integration

We briefly recall the governing equations which are at the base of the transient explicit formulation by assuming for simplicity a Lagrangian description, which is suitable for the treatment of purely structural applications. The interested reader may find further details in references [10-11].

2.1 Governing equation

For the structural domain, the governing equation is the conservation of momentum. By expressing equilibrium in the current configuration and by introducing a spatial semi-discretization based on Finite Elements, the following set of discrete differential equations in time may be obtained:

$$\underline{\underline{M}}\,\underline{a} \;=\; \underline{F}^{\text{ext}} - \underline{F}^{\text{int}}\,, \tag{1}$$

where $\underline{\underline{M}}$ is the mass matrix, $\underline{a}$ is the vector of nodal accelerations, $\underline{F}^{\text{ext}}$ are the external forces and $\underline{F}^{\text{int}}$ are the internal forces, which may be evaluated by spatial integration over the elements as:

$$\underline{F}^{\text{int}} \;=\; \sum\nolimits_e \int_{V_e} \underline{\underline{B}}^{T}\,\underline{\underline{\sigma}}\,dV\,. \tag{2}$$

In (2) the summation symbol represents the ordinary assembly operator over all the Finite Elements e of the mesh, V_e is the volume of the element in the current configuration, $\underline{\underline{B}}$ is the matrix of shape function derivatives, of which a superposed T indicates the transpose, and $\underline{\underline{\sigma}}$ is the Cauchy or "true" stress tensor.

The nodal accelerations are formally obtained from (1) as:

$$\underline{a} \;=\; \underline{\underline{M}}^{-1}(\underline{F}^{\text{ext}} - \underline{F}^{\text{int}})\;. \tag{3}$$

However, since the mass matrix $\underline{\underline{M}}$ may be lumped (i.e., reduced to diagonal form), see e.g. [12], no matrix inversion or system solution is actually required and (3) may be simply treated by considering each degree of freedom j separately:

$$a_j \;=\; (F_j^{\text{ext}} - F_j^{\text{int}})/m_j\;. \tag{4}$$

2.2 Central difference scheme

Time integration of (1) under the form (4) is achieved via the so-called central difference scheme, which is usually written as:

$$\begin{cases} \underline{v}^{n+1} = \underline{v}^{n} + \dfrac{\Delta t}{2}(\underline{a}^{n} + \underline{a}^{n+1}) \\ \underline{d}^{n+1} = \underline{d}^{n} + \Delta t\left(\underline{v}^{n} + \dfrac{\Delta t}{2}\underline{a}^{n}\right) \end{cases}, \tag{5}$$

where $\underline{v}$ are the nodal velocities, $\underline{d}$ the nodal displacements, the upper suffix n denotes a quantity at time t^n and $n+1$ denotes a quantity at time $t^{n+1} = t^n + \Delta t$, Δt being the time increment or time step used in the discretization process.

The time integration scheme (5) is implemented as follows in a typical explicit code. Assume that a complete solution, i.e. all discretized quantities, are known at time t^n. First, an intermediate or mid-step velocity is introduced (sometimes called also “velocity precursor” since it depends only upon quantities at t^n):

$$\underline{v}^{n+1/2} = \underline{v}^{n} + \frac{\Delta t}{2}\underline{a}^{n}. \tag{6}$$

This is the constant velocity that would transform configuration n into $n+1$ over a time interval Δt in the discretization process. In fact, from the second of eqs. (5) the new displacements are given by:

$$\underline{d}^{n+1} = \underline{d}^{n} + \Delta t\underline{v}^{n+1/2}. \tag{7}$$

On the new (i.e., the current) configuration induced by these displacements:

$$\underline{x}^{n+1} = \underline{x}^{n} + \Delta t\underline{v}^{n+1/2} = \underline{x}^{0} + \underline{d}^{n+1}, \tag{8}$$

the internal forces can be evaluated via eq. (2) by applying the material constitutive relations. Then, the new accelerations $\underline{a}^{n+1}$ can be directly computed via the discretized equilibrium equations (1) under the form (4), and finally the new velocities $\underline{v}^{n+1}$ are obtained from the first of eqs. (5).

It is important to note that in the time integration process the new configuration $\underline{x}^{n+1}$, induced by the displacements $\underline{d}^{n+1}$, is obtained first (except at the initial time, when the configuration $\underline{x}^{0}$ is known by definition); then, equilibrium (in a dynamic sense) is solved on the current configuration, resulting in the current accelerations $\underline{a}^{n+1}$; finally, the velocities $\underline{v}^{n+1}$ corresponding to the current configuration are obtained as the last result of the time stepping procedure.

Note that this time integration scheme is explicit in that all quantities in the right-hand-side terms are known when the equations are applied, thus no system solver is needed. This greatly simplifies the

practical implementation of the method and facilitates the treatment of non-linearities such as those arising from geometrical effects (large displacements, large strains) or from material effects (e.g., plasticity).

It is well-known that the above scheme is second-order accurate in time and introduces no numerical damping. Furthermore, spectral analysis (see [12] for details) shows that it tends to produce oscillation frequencies slightly higher than physical ones so that, when combined with the use of a lumped mass matrix —which on the contrary tends to reduce frequency values—it produces a remarkably accurate and fully explicit method. There are thus no matrices to assemble and there is no need for system solvers, except for the treatment of coupled boundary conditions (see Part II).

2.2.1 Stability

As a counterpart to these remarkable advantages, however, the explicit method (6-7) is only conditionally stable. The so-called Courant stability condition must be satisfied:

$$\Delta t \leq \Delta t^{\text{crit}}, \tag{9}$$

where Δt^{crit} indicates a critical value of the time increment, beyond which the scheme becomes numerically unstable.

A value of Δt^{crit} for each finite element i in the mesh may be estimated according to relationships of the form:

$$\Delta t_i^{\text{crit}} \approx \frac{\Delta L_i}{c_i}, \tag{10}$$

where ΔL_i is some characteristic length of the element (e.g. the minimum distance among its nodes) and c_i is the speed of sound in the material. Thus, Δt_i^{crit} corresponds physically to the time interval necessary for a stress wave to traverse the element, and suggests the following qualitative justification for eq. (9).

A direct, explicit time integration method such as the one considered above may only work as long as mechanical waves in the discrete system travel over at most one element during each time step. Otherwise, the numerical scheme has no chance of transmitting the information from an element to the neighbor ones in time to follow the physical propagation of waves. In fact, since the scheme basically considers each element and each node separately (recall eq. 4: no system solving) any “perturbation”—here in the sense of a mechanical stress or pressure wave—occurring at an element or node is only transmitted numerically to its neighbors (i.e., over the distance of one element) during each time step. These considerations, though qualitative in nature, are at the base of the spatial partitioning strategy that will be described in the next Sections.

The standard approach (i.e., without spatial partitioning) to explicit time integration consists in evaluating “the” critical time step for the whole discrete system as:

$$\Delta t^{\text{crit}} = \min_i(\Delta t_i^{\text{crit}}), \tag{11}$$

i.e. in taking the global minimum value among those of all elements i in the computational mesh. Then, eqs. (4-7) are repeatedly applied, until the final time is reached, by using for prudence a somewhat reduced time increment:

$$\Delta t = C_s \Delta t^{\text{crit}}, \tag{12}$$

where C_s is a stability safety factor:

$$0 < C_s \leq 1. \tag{13}$$

Typically, one assumes C_s in the range between 0.5 and 0.8, to account for the fact that relations of the type (10) represent just an estimation of the real stability limit for all but the simplest finite element types.

2.2.2 Variable time increment in time

Thus, normally the same time increment Δt as given by (12) and (11) is used for all elements and for all nodes in the discrete system. Actually, the time step is changed over time, since stability conditions (10) may vary locally in time due to large deformations (L_i) or to changes in the material properties (c_i). The scheme (4-7) is easily generalized to a variable Δt in time (see e.g. [10-11]) and the impact of this generalization on its implementation effort is negligible.

By indicating with Δt^n the time step that has led to the current configuration $\underline{x}^{n+1}$:

$$\Delta t^n \equiv t^{n+1} - t^n, \tag{14}$$

and by Δt^{n+1} the next time step:

$$\Delta t^{n+1} \equiv t^{n+2} - t^{n+1}, \tag{15}$$

the final non-partitioned scheme reads (by letting appear only the mid-step velocities and not the full-step ones):

$$\begin{aligned} \underline{d}^{n+1} &= \underline{d}^n + \Delta t^n \underline{v}^{n+1/2} \\ \underline{a}^{n+1} &= \underline{M}^{-1}(\underline{F}^{\text{ext}} - \underline{F}^{\text{int}})^{n+1} \\ \underline{v}^{n+3/2} &= \underline{v}^{n+1/2} + \frac{\Delta t^n + \Delta t^{n+1}}{2} \underline{a}^{n+1} \end{aligned}. \tag{16}$$

A simplified version (but containing all the essential ingredients) of the actual time integration flowchart is given in Table 1, where the symbol $\leftarrow$ is used to indicate the assignment statement. Time indexes n, $n+1$ etc. appearing e.g. in the form (16) of the equations are intentionally dropped to underline the fact that it is possible to write down the algorithm in a very compact way, by using a single instance of all variables. In other words, if each variable (velocity, stress, etc.) is updated at the right moment, there is no need to "remember" its old value, because it is no longer used.

Included in the chart are also the calculations needed to carry on the so-called energy balance check:

$$W^{\text{ext}} \approx W^{\text{int}} + W^{\text{kin}}, \tag{17}$$

where W^{int}, W^{ext}, W^{kin} are the total internal energy, external work and kinetic energy of the modelled system, respectively. Since the numerical integration scheme is undamped, energy must be conserved. Eq. (17) must therefore hold *a posteriori* with good approximation for an accurate and stable solution. A degradation in this balance usually indicates the onset of a numerical instability, e.g. due to overestimation of the critical step by means of (10).

Typically, when such an instability occurs, the numerical solution blows up within a small number of steps, so the problem is easily detected, unless by chance the instability starts very close to the final time of the simulation.

2.2.3 Drawbacks of the explicit method

Thanks to its simplicity and to its outstanding numerical properties, the explicit approach outlined in the previous Section is characterized by an exceptional robustness with respect to e.g. implicit methods, but it also has some drawbacks.

The first and most important drawback is related to computational efficiency. The calculation is driven by the smallest element, i.e., the most critical element in the sense of eq. (10). There may be applications where high refinement of the mesh is required locally to reach the desired precision, while most of the remaining mesh is made of relatively large elements. In such cases, the large elements are integrated in time tens, hundreds or even thousands of times more frequently than strictly necessary, thus leading to a substantial waste of CPU time.

But there is also a second drawback which comes from the fact that, as is well known (see e.g. [12]), the numerical accuracy of explicit time integration schemes such as the one introduced above in eqs. (5) is best for values near to (or even exactly at) the stability limit. Thus, using too small increments may introduce some (usually very small, but not vanishing) degradation of the numerical precision.

The goal—and the challenge—of the spatial partitioning technique to be introduced in the next Sections is precisely to overcome these two drawbacks, but without at the same introducing any negative

effects upon the exceptionally good properties in terms of robustness and precision of the basic explicit method outlined above, which were demonstrated by decades of applications.

In addition, we shall see that this result may be obtained at the price of only very limited and localized modifications in the implementation of a normal, i.e. non-partitioned, explicit time integration strategy. This, of course, provided the partitioning strategy is accurately chosen and designed, fully understood in all its (sometimes subtle) implications, and carefully implemented.

3. Spatial partitioning

This Section describes in some detail the proposed spatial partitioning technique, by assuming again for simplicity a Lagrangian formulation and by neglecting for the moment the treatment of boundary conditions in their most general (coupled) form (see Part II for such extensions).

In Section 3.1 an attempt is made to provide a rationale for the proposed partitioning strategy, based upon qualitative considerations about the nature of fast transient dynamic equations and about the way of functioning of the chosen numerical time integration scheme. Section 3.2 introduces the resulting data structure. A very simple but hopefully clarifying example of how the partitioning algorithm behaves in practice is offered in Section 3.3. Finally, the complete time integration algorithm with spatial partitioning is formally detailed in Section 3.4.

3.1 Fundamentals of partitioning

The basic idea behind the partitioning mechanism is to integrate in time each finite element by using "its own" stable time step—or nearly so, on the "safe" side of course—instead of a value common to all elements, dictated by the most critical one and therefore far too small for most of the elements in the mesh of a realistic 3D case.

To this end, the elements of the computational mesh regardless of their location in the model are automatically sorted over a number of "levels", according to their intrinsic stability time step, thus forming in practice a spatial partition of the mesh.

As is easily understood, and since elements are generally not isolated but connected to one another, this process will of course have implications also on the treatment of the nodes belonging to each element and connecting the element with its neighbors.

3.1.1 The case of unconnected elements

It is interesting to consider first the trivial case of a mesh composed by a set of unconnected elements, i.e. each element i is completely disjoint from the others, there are no common nodes (each node belongs to one and only one element) and there are no boundary conditions, i.e. no constraints

on the nodal degrees of freedom. An example with four simple 2-noded bar-like elements is sketched in Figure 1.

In this situation, it is clear that each element i (and all its associated nodes) may be integrated in time according to the element's own (or "intrinsic") stability time step Δt_i^{crit}, so that this quantity is sufficient to completely determine the partition. Therefore, in this case the only task for a partition algorithm would be a practical one: simply that of synchronizing from time to time the solutions obtained for the various elements, so that a consistent (synchronized) solution would be computed at some "macro" time instants, e.g. where output is required by the user.

This clarifies the fact that most of the conceptual complexity of partitioning algorithms arises from the connections between elements—be they of topological nature, i.e. the presence of common nodes—or be they dictated by *coupled* boundary conditions (coupled links), i.e. by user-imposed constraints between the degrees of freedom of *different* nodes.

3.1.2 Redundant vs. necessary updates

In setting up a spatial partitioning scheme one must aim, for optimal efficiency, at minimizing the number of updates of elemental and nodal quantities performed, by skipping any redundant or unneeded operations, that is by retaining all and only those which are mandatory to preserve the correctness, the robustness, the accuracy and the stability of the corresponding non-partitioned integration algorithm.

The whole point of the game consists therefore in identifying and separating the redundant from the necessary update operations. To this end, one possible approach is to further elaborate on the inherent way of functioning of the explicit scheme which has been already outlined in Section 2.2.1 when dealing with stability issues.

3.1.3 The role of wave propagation

As is well known, from the mathematical viewpoint the governing equations of fast transient dynamics (1) have a hyperbolic character, which means that the status of the modelled system, and its evolution in time, are mainly ruled by wave propagation phenomena, i.e. by the motion of mechanical stress or pressure waves throughout the system. For example, an external load suddenly applied at one end of a bar is not "felt" at the opposite end until the stress wave which is generated at the loaded extremity has propagated along the whole bar.

Thus, investigating the overall equilibrium of the whole, global system like it would be the case in the equivalent static problem makes no sense and is not necessary in fast transient dynamics. Equilibrium *has*, of course, to be satisfied, but only at a local scale: in the discrete model, each (spatially

discrete) element must, at each chosen time instant (time discretization) be in equilibrium (in a dynamic sense) with its neighbors, and only with them. This may qualitatively justify why eqs. (1) may be fully decoupled into (4) and may be treated explicitly without solving any full algebraic system.

Let us consider a generic element either embedded in a mesh and completely surrounded by neighbors, or located on a free boundary, thus neglecting for the moment the case of elements with nodes subjected to any boundary conditions. The element's internal state, e.g. its stress, may be altered in only one way: by a "perturbation" of at least one of its nodes, in the sense already introduced in Section 2.2.1. That is, basically, by a stress wave traveling throughout the system and reaching the element through its neighbor elements via the common nodes.

More precisely, stress variations are typically caused by changes in the relative distances between nodes of the element. In fact, as long as the element moves like a rigid body—i.e. by a combination of rigid translation and rigid rotation motions, which keep inter-nodal distances constant—its stress does not change and therefore it needs not be updated.

Let us now focus on the way in which such waves are propagated. Physically, they move across the system at a speed which is typically the speed of sound in the material and which may be computed or at least estimated in most situations. In the numerical model, however, the explicit method outlined in Section 2.2 propagates any stress wave at a rate of exactly one element per time step.

This may be simply visualized as follows, see Figure 2. Consider a mesh at rest or moving as a rigid body and imagine that an external action (e.g. an applied load) suddenly perturbs a node P of the mesh by changing its velocity, i.e. by producing a local acceleration (Figure 2a). During the first time step, the only affected elements will be the neighbor ones, i.e. the first layer of elements around the node (Figure 2b). These elements and only these will undergo variations in their stresses.

At the next (second) step, the above stress variations will perturb (i.e., accelerate) all nodes belonging to the neighbor elements (i.e., the neighbor nodes, Figure 2c), and consequently the stresses in the second layer of elements around the initially perturbed node will also be altered, Figure 2d.

At the yet subsequent (third) step, one will then see accelerations in the second layer of nodes and stress changes in the third layer of elements, and so on. Thus, the "numerical" perturbation propagates by exactly one element per time step in all space directions.

3.1.4 Numerical vs. physical propagation

This almost self-evident property of the numerical scheme may be surprising at first sight and one may wonder how can such a scheme at all work, since the numerical propagation apparently occurs

at a a completely arbitrary speed c_n, that has nothing to do with the physical one, being only dictated by the combination of the chosen mesh size ΔL and of the chosen time integration increment Δt:

$$c_n = \Delta L / \Delta t . \tag{18}$$

Indeed, we have already seen in Section 2.2.1 that in reality one of the parameters of eq. (18), the time increment Δt, is not at all arbitrary. An upper limit on Δt is dictated by stability according to eqs. (9-10) and this establishes a lower limit for the acceptable values of c_n, namely (by combining eqs. 18, 10 and 9):

$$c_n \geq c . \tag{19}$$

As already discussed in Section 2.2.1, we see therefore that the numerical propagation of perturbations must occur at a speed (c_n) at least as large as the physical one (c), else the scheme has no chance to work—and in fact it becomes unstable.

But what now about the other end of the scale, i.e. is there an upper limit for c_n, or a lower limit for Δt? The answer seems to be no, at least within a reasonable range of values. As mentioned in Section 2.2.3, a slight degradation of the scheme accuracy is observed as Δt becomes much (i.e., orders of magnitude) smaller than the critical value, but this effect is really very small compared with other sources of inaccuracy that may affect real computations.

What happens in practice when $c_n > c$ is that the perturbation is indeed propagated "too fast" in the numerical model with respect to physical reality, but at the same time the amplitude of that part of the propagated signal which overtakes the physical wave decreases very rapidly. The overall effect is that of generating a so-called "numerical precursor" which has a characteristic negative exponential shape and which is typically damped out in space to completely negligible values over just a few element layers. An illustration of this phenomenon may be observed e.g. in the first numerical examples of Section 4, see Figure 12.

3.1.5 "Rules" for partitioning

From all these considerations, one may tentatively deduce a set of qualitative "rules" for the updating of elemental and nodal quantities, that must be satisfied in order to set up a partitioned version of the explicit time integration algorithm. Note that in the non-partitioned version such rules are automatically satisfied by definition, so there is no need to consider them or—*a fortiori*—to enforce them.

Element-related quantities to be updated are typically the stresses and any constitutive law internal variables, while node-related quantities to be updated are the accelerations, the velocities and the displacements (and hence the configuration). Note that, as concerns nodal updating, the arguments detailed below will lead us to separate acceleration/velocity updating from displacement/configura-

tion updating, since in general the latter is needed more frequently than the former, and never less frequently.

In formulating the rules, use is made of the notion of "neighbor elements" to a given *element*, which are simply defined as all the elements that share at least one node with the element under consideration. Similarly, by "neighbor elements" to a given *node* we will indicate all the elements that contain that node. Furthermore, we will denote as "intrinsic" stability time increment of an element i the quantity Δt_i^{crit} defined by eq. (10), because it depends exclusively upon properties of the element under consideration, and not upon its neighbor elements. This is the basic quantity upon which the partitioning strategy is built up, and from which the actual rules for updating element and nodal quantities are derived.

The observations about stability and perturbations made in the previous Sections may be summarized as follows:

- *Observation 1: to update an element one must respect its intrinsic stability.* Each element's (i) internal state (stress) should be updated by a time increment Δt_i as close as possible (on the safe side) to the element's intrinsic critical value Δt_i^{crit}, see Section 2.2.1. This results in:

$$\Delta t_i \leq \Delta t_i^{\text{crit}}. \tag{20}$$

  To preserve stability the increment must clearly not exceed the intrinsic critical value but at the same time, to enhance efficiency, the increment should be as large as possible, compatibly with correct propagation of perturbations in the discrete model.

- *Observation 2: updating acceleration/velocity of a node immediately perturbs all its neighbor elements*. This is explained in Section 3.1.3 and illustrated in Figure 2b.

- *Observation 3: updating acceleration/velocity of a node must respect the intrinsic stability of all its neighbor elements*. As a consequence of Observations 2 and 1, to ensure that the perturbation resulting from the updating in acceleration/velocity of a node k is correctly transmitted—in a numerical sense—across its neighbor elements, the time increment $\delta t_k^{a,v}$ used to update the acceleration/velocity of the node must respect the intrinsic stability of all its neighbor elements $j(k)$. Thus one may write:

$$\delta t_k^{a,v} \leq \Delta t_{j(k)}^{\text{crit}}. \tag{21}$$

- *Observation 4: updating an element requires prior updating of all its nodal configurations*. Just before an element is updated the positions—i.e. the displacements/configurations—of all its nodes must be updated because stress variations in the element are caused by changes in the rela-

tive distances between the element's nodes, see Section 3.1.3. Therefore, the time increment $\delta t_k^{d,x}$ used to update the acceleration/velocity of a node k must be smaller than or equal to the time increments actually used to update its neighbor elements $j(k)$. One may write then:

$$\delta t_k^{d,x} \leq \Delta t_{j(k)}. \tag{22}$$

From these observations, the following three Rules are derived, which are then actually used for the updating of elemental and nodal quantities:

- ***Rule 1: node updating in acceleration/velocity***. As a direct consequence of Observation 3, each node must be updated in acceleration/velocity by using the smallest intrinsic stability time increment of all its neighbor elements. Symbolically one may write:

$$\delta t_k^{a,v} = \min(\Delta t_{j(k)}^{\text{crit}}). \tag{23}$$

- ***Rule 2: element updating***. As a direct consequence of Observation 2, each element i must be updated by using the smallest time increment among those used to update in acceleration/velocity any of its nodes $m(i)$:

$$\Delta t_i \leq \delta t_{m(i)}^{a,v}. \tag{24}$$

  Note that this condition is stricter than the condition that would result from direct application of Observation 1, i.e. than eq. (20).

- ***Rule 3: node updating in displacement/configuration***. As a direct consequence of Observation 4, each node must be updated in displacement/configuration by using the smallest time increment among those that are actually used to update all its neighbor elements:

$$\delta t_k^{d,x} = \min(\Delta t_{j(k)}) \tag{25}$$

  Note that these time increments are not the elements' intrinsic stability ones ($\Delta t_{j(k)}^{\text{crit}}$), but rather those defined by Rule 2 ($\Delta t_{j(k)}$).

3.2 Data structure for partitioning

We may now proceed to a detailed description of the partitioning strategy and of the associated data structure which, as we will see, is simply a translation into informatics terms of Rules 1, 2 and 3.

For a series of practical reasons that will be presented in the following, it is convenient to adopt a binary partitioning scheme. Thus, the various time increments used to advance the solution are not completely arbitrary, unlike in other methods such as domain decomposition. By taking as reference

one of the elements, e.g. the element with the largest intrinsic stability step, all others will use increments whose ratios to the reference are integer (negative) powers of 2, namely 1, 1/2, 1/4, 1/8 etc.

3.2.1 Frequency of sub-cycling

It seems natural to store this information as integer numbers (1, 2, 4, 8 etc.) and therefore, instead of dealing directly with time increments like in the Rules introduced in Section 3.1.5, we will rather introduce the term “frequency (of sub-cycling)”—not to be confused with frequency of mechanical or numerical oscillations—which is inversely proportional to the time increment. For example, an element or node with an associated frequency of 4 will be integrated (updated) 4 times more often than the reference one and thus it will use a time increment 4 times smaller. As a consequence of this choice, all the minimization processes that appear in the Rules and that deal with time increments are converted into maximization processes in the data structure, which instead uses the notion of frequency.

The terminology adopted in the description of the partition is clarified in Figure 3 and, as concerns the notation, most of the symbols and variable names used in this work are summarized in Table 2 for ease of reference. To help the reader, the same variable is indicated in upper-case when referred to a partition level, in lower-case when it refers to a particular element or node. For example, time will be denoted as T in the first case, t in the second one. Furthermore, variations relative to a level or an element are indicated by Δ, while those relative to a nodal quantity are denoted by δ.

3.2.2 Partition levels

A binary partition of arbitrary depth d ($d \geq 1$) is built up, whereby each element i ($i = 1, \ldots, N_e$), N_e being the total number of elements, is assigned to a level l ($l = 1, \ldots, d$) according to its intrinsic stability time increment Δt_i. The smaller Δt_i, the lower is the level to which element i is assigned, i.e. the larger is l.

Each element thus belongs to one and only one level, while each level contains zero or more elements, except for the first and the last levels ($l = 1$ and $l = d$, respectively), which must contain at least one element.

A time increment ΔT_l is associated with each level l according to a binary rule (and hence we qualify the partition as binary), i.e. such that:

$$\Delta T_l = \frac{1}{2}\Delta T_{l-1} \qquad l = 2, \ldots, d\,. \tag{26}$$

This implies also:

$$\Delta T_l = 2\Delta T_{l+1} \qquad l = 1, \ldots, d-1\,. \tag{27}$$

An element i is associated with level l if and only if its intrinsic stability time increment Δt_i satisfies the following inequalities:

$$\Delta T_{l+1} < \Delta t_i \leq \Delta T_l \,. \tag{28}$$

If we indicate with $\Delta t_{\max}$ (or simply with ΔT) the largest time increment in the partition, i.e. the "macro" step, which is also the time increment associated with the first level:

$$\Delta t_{\max} \equiv \Delta T \equiv \Delta T_1 \,, \tag{29}$$

then in view of (26) or (27) the time increment of the generic level l is given by:

$$\Delta T_l \;=\; \frac{1}{2^{l-1}} \Delta t_{\max} \qquad l \;=\; 1, \ldots, d \,. \tag{30}$$

If we denote as level frequency Φ_l the (integer) quantity appearing at the denominator in eqn. (30):

$$\Phi_l \equiv 2^{l-1} \qquad l \;=\; 1, \ldots, d \,, \tag{31}$$

then this equation becomes:

$$\Delta T_l \;=\; \frac{\Delta t_{\max}}{\Phi_l} \qquad l \;=\; 1, \ldots, d \,. \tag{32}$$

Note that $\Phi_l \;=\; 1, 2, 4, 8, \ldots, 2^{d-1}$ for levels $l \;=\; 1, 2, 3, 4, \ldots, d$, respectively.

The level frequency Φ_l represents the number of "micro" steps (or sub-cycles) that have to be performed on the elements belonging to level l, for each single macro step, i.e. for each step performed on the elements belonging to the first level. In other words, Φ_l is the frequency of sub-cycling of the generic level l with respect to the first level.

From eqn. (31) one may compute the level l from its level frequency Φ_l, by:

$$l \;=\; \log_2(\Phi_l) + 1 \qquad l \;=\; 1, \ldots, d \,. \tag{33}$$

A property of the binary partition which is very important in practice is that the time integration instants associated with elements in a certain level l coincide every second step with those of elements in the previous level ($l-1$) and, more generally, they coincide every $2^{l-l'}$ steps with those of elements in level l' (with $1 \leq l' < l$).

In particular, the integration times of elements in the first level ($l' \;=\; 1$), which we denote as macro times, are guaranteed to coincide with some of the integration times of elements in any subsequent level l (and this every 2^{l-1} steps for the elements in that particular level). Thus, in the chosen partition strategy synchronization of the integration times is guaranteed by construction, unlike in other techniques such as e.g. domain decomposition which in principle allow completely arbitrary time

steps, but then require an explicit synchronization mechanism to obtain a complete solution at a series of chosen macro time instants.

The use of a binary partition is also very convenient from the implementation viewpoint, since the time marching algorithm can be built up by using exclusively integer quantities. In particular, algebraic or comparison operations involving real numbers, which are necessary when using arbitrary time steps like in domain decomposition techniques and require the definition of suitable tolerances (with all the associated potential numerical problems), are completely avoided.

An example of partition with a depth $d = 4$ is illustrated in Figure 3, which defines also the nomenclature associated with the partition algorithm and used throughout this work. The figure shows two consecutive macro time steps $\Delta T'$, $\Delta T''$ for the elements in the first level, and the associated subcycles for the other levels. Note that time increment varies not only in space but also in time, since it is assumed that $\Delta T' \neq \Delta T''$ for full generality.

The above equations (26-33) define the basic ingredients of the element partitioning technique, but a couple of questions of great practical importance remain open at this point:

- The assignment of an element to a certain level and consequently also the partition depth varies in general from a macro step to the following one. But may they also vary *within* a macro step? In the present implementation, the answer to this question is yes: to achieve full generality in large deformation simulations the partition is checked and, if necessary, it is updated at every (sub-)cycle of the time integration process. More on this in Section 3.4.7.
- Eqn. (28) does not indicate how to precisely determine the partition levels. In particular, a certain freedom remains on how to choose the time increment of the first level, all others resulting uniquely from this one. A technique is presented in Section 3.4.8, but alternative strategies could perhaps be more convenient depending on circumstances, as will be shortly mentioned at the end of Section 4.2.2.

The level frequency Φ_l, $l = 1, \ldots, d$ introduced above by eqn. (31) is the primary information for the spatial partitioning process, but it is not sufficient to completely define the partitioned time integration algorithm. Other quantities, derived from it and necessary to enforce the Rules detailed in Section 3.1.5, must be built up according to the following procedures. Their precise meaning and utilization will be detailed in Section 3.4 that presents the complete time integration algorithm.

3.2.3 Intrinsic element frequency

First of all, the "intrinsic element frequency" φ_i for each element i in the mesh is computed, according to eqn. (28):

$$\varphi_i = \Phi_l \qquad \text{such that} \qquad \Delta T_{l+1} < \Delta t_i \leq \Delta T_l \qquad i = 1, \ldots, N_e . \tag{34}$$

The term intrinsic refers to the fact that this is the frequency of sub-cycling of the element *per se*, as resulting from local stability, i.e. by disregarding any connections of the element with its neighbors.

3.2.4 Intrinsic node frequency

Next, we compute the frequency associated with each node in the mesh. We consider first the case of "free" nodes, i.e. nodes not subjected to *any* constraints (denoted "links" in the following). The case of linked nodes will be treated in Section 3.2.5.

The intrinsic node frequency ψ_k for a free node k, ($k = 1, \ldots, N_n$ where N_n represents the total number of nodes in the mesh) is defined as the maximum intrinsic element frequency of its neighbor elements:

$$\psi_k^{\text{free}} = \max(\varphi_j) \qquad j = \text{any element sharing node } k \qquad k = 1, \ldots, N_n . \tag{35}$$

This quantity is related to the enforcement of Rule 1 of Section 3.1.5, i.e. it will be used to drive the updates of nodal accelerations and velocities.

3.2.5 Nodal link indicator

As already mentioned in the Introduction, the original implementation of the partitioning technique dealt with boundary conditions in a simplistic way. In order to avoid the complications arising from the links between different nodes, the intrinsic node frequency ψ_k for any node k subjected to a link condition, *no matter whether coupled or uncoupled* (like e.g. in the case of a simple blockage of one dof), is set to the maximum value of the partition:

$$\psi_k^{\text{link}} = \Phi_d \equiv 2^{d-1} \qquad \forall k \text{ subjected to a link} . \tag{36}$$

To do this efficiently, a nodal link indicator λ_k is built up once (thereby implicitly considering only the permanent links, i.e. those which hold over the whole transient of interest for the calculation) and is then used during the first calculation or during the updating of the partition:

$$\lambda_k = \begin{cases} 0 \text{ if node } k \text{ is not subjected to links} \\ 1 \text{ if node } k \text{ is subjected to links} \end{cases} \qquad k = 1, \ldots, N_n . \tag{37}$$

A node is considered subjected to a link if at least one of its degrees of freedom (dof) appears in any expression imposing a link. Obviously, this treatment is penalizing especially in the case of uncoupled links, for which it is totally unnecessary. However, recognizing whether a given link is effec-

tively coupled or uncoupled is a relatively expensive operation and therefore it is avoided here for maximum simplicity.

3.2.6 Neighboring element frequency

Next, the neighboring element frequency $\bar{\varphi}_i$ is also computed and stored for each element i in the mesh. This quantity is defined as the maximum of the intrinsic node frequencies of all nodes belonging to the element i itself:

$$\bar{\varphi}_i = \max(\psi_m) \qquad m = \text{any node belonging to element } i \qquad i = 1, \ldots, N_e, \tag{38}$$

and is related to the enforcement of Rule 2 of Section 3.1.5, i.e. it will be used to drive the updates of element quantities.

By this definition and by the definitions of intrinsic nodal frequency (35) and (36), one may see that the neighboring element frequency is always greater than or equal to the intrinsic element frequency:

$$\bar{\varphi}_i \geq \varphi_i \qquad i = 1, \ldots, N_e. \tag{39}$$

3.2.7 Neighboring node frequency

Finally, in analogy with (38) we define the neighboring node frequency $\bar{\psi}_k$ for each node k in the mesh, as the maximum neighboring element frequency of its neighbor elements:

$$\bar{\psi}_k = \max(\bar{\varphi}_j) \qquad j = \text{any element sharing node } k \qquad k = 1, \ldots, N_n. \tag{40}$$

This quantity is related to the enforcement of Rule 3 of Section 3.1.5, i.e. it will be used to drive the updates of nodal displacements and configurations.

In analogy with (39), we may note that:

$$\bar{\psi}_k \geq \psi_k \qquad k = 1, \ldots, N_n. \tag{41}$$

3.2.8 Nodal time increment

A final quantity related to mesh nodes is used in the original partitioning algorithm: the nodal time increment δt_k, for each node of the mesh ($k = 1, \ldots, N_n$). This quantity is defined as the time increment that is being currently used to advance in time the acceleration/velocity related to node k. It is stored and continuously updated by the partitioning algorithm as part of the time integration process (see the complete description of the algorithm in Section 3.4 below for further details).

3.3 The partitioning mechanism at work

We now describe by means of a simple example how spatial partitioning may be achieved in practice by making an appropriate use of the quantities introduced in the previous Section. The scope is to give readers with a pragmatic attitude a glimpse into the inner workings of the partitioning mechanism in a very simple yet representative practical case. The theoretically oriented readers may safely

skip this Section since a more formal and fully detailed presentation of the partitioning algorithm will be given in Section 3.4.

3.3.1 The case of unconnected elements

Before considering the practical example, however, it is instructive to see what would be the behavior of the partition in the trivial case of completely unconnected elements already discussed in Section 3.1.1 and sketched in Figure 1.

In that situation, it is clear that each element (and the associated nodes) may be integrated in time according to the element's own stability time step, so that the intrinsic element frequency φ_i as given by (34) is sufficient to completely determine the partition. In fact, from (35), (38) and (40) one sees that all frequencies are equal and depend only upon the element considered: $\psi_k = \varphi_i$, $\bar{\varphi}_i = \varphi_i$ and $\bar{\psi}_k = \varphi_i$, where k are the nodes of element i.

3.3.2 A connected mesh

Let us consider again the same mesh as in the previous example, but now with the elements connected to one another to form a one-dimensional bar, as shown in Figure 4. There are now only 5 nodes instead of 8, because of topological connections. Assume that the stability time increment for element 1 be one half that of element 2, which in turn is one half of those of elements 3 and 4.

From (34) we obtain:

$$\varphi_1 = 4 \qquad \varphi_2 = 2 \qquad \varphi_3 = 1 \qquad \varphi_4 = 1. \tag{42}$$

From (35):

$$\psi_1 = 4 \qquad \psi_2 = 4 \qquad \psi_3 = 2 \qquad \psi_4 = 1 \qquad \psi_5 = 1. \tag{43}$$

From (38):

$$\bar{\varphi}_1 = 4 \qquad \bar{\varphi}_2 = 4 \qquad \bar{\varphi}_3 = 2 \qquad \bar{\varphi}_4 = 1. \tag{44}$$

Finally, from (40):

$$\bar{\psi}_1 = 4 \qquad \bar{\psi}_2 = 4 \qquad \bar{\psi}_3 = 4 \qquad \bar{\psi}_4 = 2 \qquad \bar{\psi}_5 = 1. \tag{45}$$

The arrows in Figure 4 indicate how the most critical (highest) intrinsic element frequencies φ_i are successively "spreading" towards the neighboring nodes and elements during the calculation of the other (derived) frequencies ψ_k, $\bar{\varphi}_i$ and $\bar{\psi}_k$. Solid arrows indicate values that are larger than—and therefore become predominant with respect to—those indicated by dotted arrows.

Note incidentally how, due to the "non-recursive" nature of frequencies definitions (e.g., ψ_k depend only upon φ_i, and not upon ψ_k themselves) given in the previous Section, the value associated with the critical element (e.g. element 1 in the example: $\varphi_1 = 4$) spreads out to the neighbors ($\bar{\varphi}_2 = 4$), i.e. only over one single layer of elements. In fact, if this spreading mechanism would be recursive,

then the most critical value would uniformly affect the whole mesh (assuming this is fully connected), thus rendering the partitioning completely useless: all elements and nodes would simply end up in the lowest partition level.

Criterion for elements updating

During the time integration process, elements are integrated (i.e., as we will see in detail below, their internal state of stress and their contributions to internal forces are updated) with different frequencies depending upon each element's critical step, thanks to the partitioning process.

One might at first sight expect that the quantity used to decide whether or not a certain element i should be integrated at a given cycle would be the intrinsic element frequency φ_i. However, this is not the case and one should use the neighboring element frequency $\bar{\varphi}_i$ instead, in accordance with Rule 2 of Section 3.1.5 and as will be tentatively justified based upon the example under consideration.

In fact, consider element 2 in the example of Figure 4. It should be integrated with a frequency of 4 ($\bar{\varphi}_2$) and not of 2 (φ_2) because it is connected through node 2 to element 1, which has an intrinsic frequency of 4 (φ_1).

A rationale for this could be as follows (see also Section 3.1): imagine the bar is currently at rest or moving as a rigid body and suddenly a perturbation (a stress wave) enters the bar from the left end and propagates right wards, and assume that at the current time it produces an acceleration (change of velocity) of node 2. In the time and space discretization assumed above, this perturbation will "instantaneously" affect both elements sharing the perturbed node, i.e. elements 1 and 2. Consequently, both elements will have to be time integrated at the current time in order to update their status: integrating only element 1 (according to φ_i) would in fact overlook the perturbation in element 2.

However, the same perturbation will not affect (neither physically nor numerically) the following element (element 3) until it has completely traversed element 2, thus producing a perturbation of node 3. But in the assumed numerical scheme this may happen only at the next cycle, as a consequence of the stress change in element 2. Therefore there is no need to update element 3 at the current time, since it belongs to a higher partition level (lower frequency) than element 1: its internal status would not change anyway.

Partitioned time integration

The spatial partition for the mesh of Figure 4 is illustrated in Figure 5. This is similar to the generic partition of Figure 3, except for the following details:

- Only one macro step has been represented for simplicity.
- The neighboring element frequency $\bar{\varphi}_i$ has been depicted instead of the level frequency Φ_l.
- The partition depth is 3 instead of 4.

For each integration in time of element 4 (macro step), element 3 is integrated twice while elements 1 and 2 are integrated four times. Thus there are 4 cycles within the macro step.

Initialization of the macro step

Let us assume that all quantities, i.e. for all elements and all nodes, are known at the beginning of the macro step (t_0): for simplicity let the system be initially stress-free ($\sigma_{i,0} = 0$) and at rest except at node 2, where an arbitrary perturbation is introduced (velocities $v_{k,0} = 0$ for $k \neq 2$, $v_{2,0} \neq 0$).

Note that all variables are indicated here as scalars (σ instead of $\underline{\sigma}$, v instead of $\underline{v}$, etc.), for notational simplicity, because the present example is one-dimensional. The first subscript indicates the element or node, the second one (separated by a comma) indicates the time station according to the normalized time scale shown in Figure 5.

To advance in time the numerical solution for the discrete system, use is made of the central difference time integration scheme which has been shortly recalled in Section 2.2. The internal forces in all elements are computed and assembled onto the corresponding nodes. They will be denoted as F instead of F^{int}, for simplicity, throughout this Section. Since there is no initial stress, these forces are null. Then, the nodal accelerations are computed: they are also zero since we assume no external forces throughout the whole transient of interest:

$$F_{k,0} = 0 \quad , \quad F_k^{\text{ext}} = 0 \quad \Rightarrow \quad a_{k,0} = 0 . \tag{46}$$

Next, we start advancing the numerical solution in time. The nodal accelerations are used to compute the so-called mid-step velocities v'_k at each node k in the mesh ($k = 1, \ldots, N_n$). These are the (constant) velocities that would move each node k to the next (computed) configuration over its own time increment δt_k, and result from (see eq. 6):

$$v'_k = v_k + \frac{\delta t_k}{2} a_k . \tag{47}$$

Note the factor $1/2$, due to the fact that we are at the beginning of a macro step, thus we start from the known full-step velocities v. Note also that, since the nodal time increment δt_k varies from node to node, these velocities v' are in general expressed at different times.

The quantity used to determine the time increment for each node δt_k is the intrinsic nodal frequency ψ_k, according to:

$$\delta t_k = \Delta T / \psi_k , \tag{48}$$

where ΔT is the "macro" time step, i.e. the largest (level 1) time increment in the partition, see also eq. (29). Alternatively, δt_k may be expressed in terms of the smallest (level d) time increment in the partition, ΔT_d (that will be indicated by τ in the rest of this Section for brevity):

$$\delta t_k = \Delta T_d \Phi_d / \psi_k = \tau \Phi_d / \psi_k, \tag{49}$$

since, according to (32):

$$\Delta T = \tau \Phi_d, \tag{50}$$

where Φ_d is the level frequency of the deepest partition level.

In the chosen example, we obtain for the nodal time increments the following values:

$$\delta t_1 = \delta t_2 = \frac{\Delta T}{4} \quad , \quad \delta t_3 = \frac{\Delta T}{2} \quad , \quad \delta t_4 = \delta t_5 = \Delta T. \tag{51}$$

Hence, the mid-step velocities are:

$$v_{1,\,1/8} = 0 \qquad v_{2,\,1/8} = v_{20} \qquad v_{3,\,1/4} = 0 \qquad v_{4,\,1/2} = 0 \qquad v_{5,\,1/2} = 0. \tag{52}$$

This situation is illustrated graphically in Figure 6(a). In this figure and in the following ones within this Section, solid arrows and circled symbols indicate the most recently updated quantities. In the color version of this paper, velocities are indicated in red, configurations in green and accelerations in blue.

Cycle 1

Next, we start the sub-cycling process, whereby time is repeatedly advanced by the smallest time increment in the partition, τ, until the macro step is completed. At each time station, only some of the nodes and elements are "active" and are therefore computed (i.e., their variables are updated), while the others are simply skipped (i.e., the values of the associated variables do not change).

Cycle 1 in the above example corresponds to time $t_{1/4}$ in Figure 6(a). The first quantity to be determined in the central difference time integration process is the configuration at the current time, and this must be done for all active nodes at this time.

Criterion for updating nodal configurations

One might at first sight expect that the quantity used to decide whether or not the displacement/configuration of a certain node k should be updated at a given cycle would be the intrinsic nodal frequency ψ_k, and indeed this is the quantity that has been used above in eqs. (49) and (47) to update the mid-step velocities.

However, similarly to what happens for elements (see the discussion at the beginning of Section 3.3.2) this is not the case and one should use the neighboring node frequency $\bar{\psi}_k$ instead, in accordance with Rule 3 of Section 3.1.5. Again, the justification for this may be deduced by closer inspection of the example under consideration.

As a matter of fact, the purpose of updating the nodal configuration is, ultimately, that of updating the elements by computing the new stresses resulting from the new configuration and the internal forces that these stresses generate.

Now, the elements which are active at the current time $t_{1/4}$ are, as explained above, those having $\bar{\varphi} = 4$, i.e. elements 1 and 2. All nodes belonging to these elements should be updated (in configuration/displacement) before recomputing the elements themselves, and these include not only nodes 1 and 2 (which indeed have $\psi = 4$), but also node 3 (which has $\psi = 2$). Thus, one sees that the quantity to be used here is $\bar{\psi}$, and not ψ: in fact $\bar{\psi} = 4$ for nodes 1, 2 and 3 (and only for these).

Therefore, cycle 1 starts by computing the following displacement increments:

$$\delta u_1 = v_{1,\,1/8}\Delta T/\bar{\psi}_1 \qquad \delta u_2 = v_{2,\,1/8}\Delta T/\bar{\psi}_2 \qquad \delta u_3 = v_{3,\,1/4}\Delta T/\bar{\psi}_3, \tag{53}$$

or (since $\bar{\psi}_1 = \bar{\psi}_2 = \bar{\psi}_3 = 4$ and $\tau = \Delta T/4$):

$$\delta u_1 = v_{1,\,1/8}\tau \qquad \delta u_2 = v_{2,\,1/8}\tau \qquad \delta u_3 = v_{3,\,1/4}\tau. \tag{54}$$

Correspondingly, the new configurations at $t_{1/4}$ are:

$$x_{1,\,1/4} = x_{1,\,0} + \delta u_1 \qquad x_{2,\,1/4} = x_{2,\,0} + \delta u_2 \qquad x_{3,\,1/4} = x_{3,\,0} + \delta u_3. \tag{55}$$

Note that, for nodes 1 and 2 (which actually belong to the lowest partition level, see Figure 6a) these configurations are the actual “end of (sub-)step” ones corresponding to the intrinsic time increment for these nodes, while for node 3 the obtained configuration may be thought of as an intermediate (or interpolated) one, which is obtained with the only purpose of being able to compute element 2 at the current time. The situation is graphically summarized in Figure 6(b).

On the basis of the updated nodal configurations and of the associated displacement increments, the elements 1 and 2 (which are the only ones with $\bar{\varphi} = 4$) may now be updated. One may symbolically write for the stresses:

$$\begin{aligned} \sigma_{1,\,1/4} &= \sigma_{1,\,0} + \Delta\sigma_1(x_{1,\,1/4}, x_{2,\,1/4}, \delta u_1, \delta u_2) \\ \sigma_{2,\,1/4} &= \sigma_{2,\,0} + \Delta\sigma_2(x_{2,\,1/4}, x_{3,\,1/4}, \delta u_2, \delta u_3) \end{aligned}, \tag{56}$$

and for the internal forces:

$$\begin{aligned} F_{1,\,1/4} &= F_{1,\,1/4}(\sigma_{1,\,1/4}) \\ F_{2,\,1/4} &= F_{2,\,1/4}(\sigma_{1,\,1/4}, \sigma_{2,\,1/4}) \\ \tilde{F}_{3,\,1/4} &= \tilde{F}_{3,\,1/4}(\sigma_{2,\,1/4}) \end{aligned}. \tag{57}$$

Contributions from the various elements to internal nodal forces are assembled at nodes by the standard Finite Element process. At the current time, the assembled forces at nodes 1 and 2 as given by eqs. (57) are complete, while the force at node 3 is not, since it includes only the contribution from element 2. For this reason it has been indicated by a superposed $\sim$ in those equations. This is not a

problem, however, since at the current time node 3 is *not* going to be updated, as far as accelerations and velocities are concerned (see hereafter).

Criterion for updating nodal accelerations and velocities

In order to complete the current (first) cycle, the other nodal quantities (accelerations and velocities) must be updated for the currently active nodes. Here the choice is done based upon ψ_k (and not $\bar{\psi}_k$), in accordance with Rule 1 of Section 3.1.5, and results into nodes 1 and 2. For the accelerations we obtain:

$$a_{1,\,1/4} = -F_{1,\,1/4}/m_1 \qquad a_{2,\,1/4} = -F_{2,\,1/4}/m_2 \ . \tag{58}$$

To update the (mid-step) velocities, we use the relationship:

$$v'_k = (v'_k)^{\text{old}} + \delta t_k a_k , \tag{59}$$

which is analogous to eq. (47) except for the lack of the factor $1/2$. This is due to the fact that we now start from the previous mid-step values $(v'_k)^{\text{old}}$ and not from the full-step ones like in eq. (47).

We thus obtain (since $\psi_1 = \psi_2 = 4$):

$$\begin{aligned} v_{1,\,3/8} &= v_{1,\,1/8} + (\Delta T/\psi_1) a_{1,\,1/4} = v_{1,\,1/8} + \tau a_{1,\,1/4} \\ v_{2,\,3/8} &= v_{2,\,1/8} + (\Delta T/\psi_2) a_{2,\,1/4} = v_{2,\,1/8} + \tau a_{2,\,1/4} \end{aligned} , \tag{60}$$

One should stress the fact that these velocities are not expressed at the current time, but are already the mid-step values for the subsequent cycle and for each node at the appropriate time. In fact, recall (see e.g. [10]) that the “end-of (sub-)step” velocities are used only for printout purposes and for the calculation of kinetic energies, since the velocities values used for time integration are the mid-step ones.

This terminates the first cycle ($t_{1/4}$) of the time integration procedure, see Figure 5. The situation at this moment is graphically summarized in Figure 6(c): a complete solution has been obtained at the current time $t_{1/4}$ for the deepest (third) level of the partition, i.e. for nodes 1 and 2 (configurations x, accelerations a) and for elements 1 and 2 (stress σ). The nodal velocities for all nodes already have the mid-step values appropriate, on a node-by-node basis, for the next nodal integration step. Furthermore, the configurations of nodes in the first non-active level (x_3) has also been updated and finds itself in a sort of intermediate status referred to the time scale of that node, as noted before.

Cycle 2

We may now start the second cycle, whereby time is advanced from $t_{1/4}$ to $t_{1/2}$. From Figure 5 it appears that the active levels at the current time $t_{1/2}$ are levels 3 and 2.

Like in the previous cycle, we start by computing the current configurations for all active nodes (based upon $\bar{\psi}_k$). These are now nodes 1, 2, 3 and 4. The configuration of the latter node is updated

only in order to be able to advance element 3, which is active at the current time based upon its $\bar{\varphi}$ ($\bar{\varphi}_3 = 2$).

By noting that $\bar{\psi}_1 = \bar{\psi}_2 = \bar{\psi}_3 = 4$, $\bar{\psi}_4 = 2$, the displacement increments are:

$$\begin{aligned} \delta u_1 &= v_{1,\,3/8}(\Delta T/\bar{\psi}_1) = v_{1,\,3/8}\tau \\ \delta u_2 &= v_{2,\,3/8}(\Delta T/\bar{\psi}_2) = v_{2,\,3/8}\tau \\ \delta u_3 &= v_{3,\,1/4}(\Delta T/\bar{\psi}_3) = v_{3,\,1/4}\tau \\ \delta u_4 &= v_{4,\,1/2}(\Delta T/\bar{\psi}_4) = v_{4,\,1/2}\cdot 2\tau \end{aligned} \quad . \tag{61}$$

The new configurations are therefore:

$$\begin{aligned} x_{1,\,1/2} &= x_{1,\,1/4} + \delta u_1 \qquad x_{2,\,1/2} = x_{2,\,1/4} + \delta u_2 \\ x_{3,\,1/2} &= x_{3,\,1/4} + \delta u_3 \qquad x_{4,\,1/2} = x_{4,\,0} + \delta u_4 \end{aligned} \quad . \tag{62}$$

The situation is graphically summarized in Figure 7(a).

On the basis of the updated nodal configurations and of the associated displacement increments, the elements 1, 2 and 3 (i.e. the only ones with either $\bar{\varphi} = 4$ or $\bar{\varphi} = 2$) may now be updated. For the stresses, we get:

$$\begin{aligned} \sigma_{1,\,1/2} &= \sigma_{1,\,1/4} + \Delta\sigma_1(x_{1,\,1/2}, x_{2,\,1/2}, \delta u_1, \delta u_2) \\ \sigma_{2,\,1/2} &= \sigma_{2,\,1/4} + \Delta\sigma_2(x_{2,\,1/2}, x_{3,\,1/2}, \delta u_2, \delta u_3) \\ \sigma_{3,\,1/2} &= \sigma_{3,\,0} + \Delta\sigma_3(x_{3,\,1/2}, x_{4,\,1/2}, \delta u_3, \delta u_4) \end{aligned} , \tag{63}$$

and for the internal forces:

$$\begin{aligned} F_{1,\,1/2} &= F_{1,\,1/2}(\sigma_{1,\,1/2}) \qquad F_{2,\,1/2} = F_{2,\,1/2}(\sigma_{1,\,1/2}, \sigma_{2,\,1/2}) \\ F_{3,\,1/2} &= F_{3,\,1/2}(\sigma_{2,\,1/2}, \sigma_{3,\,1/2}) \qquad \tilde{F}_{4,\,1/2} = \tilde{F}_{4,\,1/2}(\sigma_{3,\,1/2}) \end{aligned} . \tag{64}$$

Note that the force in node 3 ($F_{3,\,1/2}$) is computed anew as a function of current ($t_{1/2}$) stresses in elements 2 and 3, so that the partial value ($\tilde{F}_{3,\,1/4}$) that had been computed in the previous cycle is discarded. Now it is the internal force of node 4 ($\tilde{F}_{4,\,1/2}$) that is incomplete.

To complete the cycle, the nodal accelerations and velocities must be updated for the currently active nodes (based upon ψ_k), i.e. nodes 1, 2 and 3. The accelerations are:

$$a_{1,\,1/2} = -F_{1,\,1/2}/m_1 \qquad a_{2,\,1/2} = -F_{2,\,1/2}/m_2 \qquad a_{3,\,1/2} = -F_{3,\,1/2}/m_3 \ , \tag{65}$$

and since $\psi_1 = \psi_2 = 4$, $\psi_3 = 2$, the new mid-step velocities are:

$$\begin{aligned} v_{1,\,5/8} &= v_{1,\,3/8} + (\Delta T/\psi_1)a_{1,\,1/2} = v_{1,\,3/8} + \tau a_{1,\,1/2} \\ v_{2,\,5/8} &= v_{2,\,3/8} + (\Delta T/\psi_2)a_{2,\,1/2} = v_{2,\,3/8} + \tau a_{2,\,1/2} \\ v_{3,\,3/4} &= v_{3,\,1/4} + (\Delta T/\psi_3)a_{3,\,1/2} = v_{3,\,1/4} + 2\tau a_{3,\,1/2} \end{aligned} . \tag{66}$$

This terminates the second cycle ($t_{1/2}$) of the time integration procedure, see Figure 5. The situation is depicted in Figure 7(b).

Cycle 3

In the third cycle, time is advanced from $t_{1/2}$ to $t_{3/4}$. From Figure 5 it appears that the only active level at $t_{3/4}$ is level 3 (like in the first cycle). We start by computing the configurations of all active nodes based upon $\bar{\psi}_k$, i.e. nodes 1, 2 and 3. Since $\bar{\psi}_1 = \bar{\psi}_2 = \bar{\psi}_3 = 4$, the displacement increments are:

$$\begin{aligned} \delta u_1 &= v_{1,5/8}(\Delta T/\bar{\psi}_1) = v_{1,5/8}\tau \\ \delta u_2 &= v_{2,5/8}(\Delta T/\bar{\psi}_2) = v_{2,5/8}\tau \\ \delta u_3 &= v_{3,3/4}(\Delta T/\bar{\psi}_3) = v_{3,3/4}\tau \end{aligned} . \tag{67}$$

Correspondingly, the new configurations are:

$$\begin{aligned} x_{1,3/4} &= x_{1,1/2} + \delta u_1 \qquad x_{2,3/4} = x_{2,1/2} + \delta u_2 \\ x_{3,3/4} &= x_{3,1/2} + \delta u_3 \end{aligned} . \tag{68}$$

The situation is depicted in Figure 8(a).

On the basis of the updated nodal configurations and of the associated displacement increments, the elements 1, and 2 (i.e. the only ones with $\bar{\varphi} = 4$) may now be updated. For the stresses, we obtain:

$$\begin{aligned} \sigma_{1,3/4} &= \sigma_{1,1/2} + \Delta\sigma_1(x_{1,3/4}, x_{2,3/4}, \delta u_1, \delta u_2) \\ \sigma_{2,3/4} &= \sigma_{2,1/2} + \Delta\sigma_2(x_{2,3/4}, x_{3,3/4}, \delta u_2, \delta u_3) \end{aligned} , \tag{69}$$

and for the internal forces:

$$\begin{aligned} F_{1,3/4} &= F_{1,3/4}(\sigma_{1,3/4}) \qquad F_{2,3/4} = F_{2,3/4}(\sigma_{1,3/4}, \sigma_{2,3/4}) \\ \tilde{F}_{3,3/4} &= \tilde{F}_{3,3/4}(\sigma_{2,3/4}) \end{aligned} . \tag{70}$$

Again, note that the internal force in node 3 is incomplete.

To complete the cycle, the nodal accelerations and velocities must be updated for the currently active nodes (based upon ψ_k), i.e. nodes 1, and 2. The accelerations are:

$$a_{1,3/4} = -F_{1,3/4}/m_1 \qquad a_{2,3/4} = -F_{2,3/4}/m_2 , \tag{71}$$

and the velocities (since $\psi_1 = \psi_2 = 4$):

$$\begin{aligned} v_{1,7/8} &= v_{1,5/8} + (\Delta T/\psi_1)a_{1,3/4} = v_{1,5/8} + \tau a_{1,3/4} \\ v_{2,7/8} &= v_{2,5/8} + (\Delta T/\psi_2)a_{2,3/4} = v_{2,5/8} + \tau a_{2,3/4} \end{aligned} . \tag{72}$$

This terminates the third cycle ($t_{3/4}$) of the time integration procedure, see Figure 5. The situation is depicted in Figure 8(b).

Cycle 4

In the fourth and last cycle of the macro step, time is advanced from $t_{3/4}$ to t_1. Since this is the last cycle (end of the macro step), all elements and all nodes are active. Since $\bar{\psi}_1 = \bar{\psi}_2 = \bar{\psi}_3 = 4$, $\bar{\psi}_4 = 2$, $\bar{\psi}_5 = 1$, the displacement increments are:

$$\begin{aligned} \delta u_1 &= v_{1,\,7/8}(\Delta T/\bar{\psi}_1) = v_{1,\,7/8}\Delta T_d \\ \delta u_2 &= v_{2,\,7/8}(\Delta T/\bar{\psi}_2) = v_{2,\,7/8}\Delta T_d \\ \delta u_3 &= v_{3,\,3/4}(\Delta T/\bar{\psi}_3) = v_{3,\,3/4}\Delta T_d \\ \delta u_4 &= v_{4,\,1/2}(\Delta T/\bar{\psi}_4) = v_{4,\,1/2}2\Delta T_d \\ \delta u_5 &= v_{5,\,1/2}(\Delta T/\bar{\psi}_5) = v_{5,\,1/2}4\Delta T_d \end{aligned} \quad . \tag{73}$$

Correspondingly, the new configurations at t_1 are:

$$\begin{aligned} x_{1,\,1} &= x_{1,\,3/4} + \delta u_1 \qquad x_{2,\,1} = x_{2,\,3/4} + \delta u_2 \\ x_{3,\,1} &= x_{3,\,3/4} + \delta u_3 \\ x_{4,\,1} &= x_{4,\,1/2} + \delta u_4 \qquad x_{5,\,1} = x_{5,\,0} + \delta u_5 \end{aligned} \quad . \tag{74}$$

The situation is depicted in Figure 9(a).

On the basis of the updated nodal configurations and of the associated displacement increments, all elements may now be updated. For the stresses, we get:

$$\begin{aligned} \sigma_{1,\,1} &= \sigma_{1,\,3/4} + \Delta\sigma_1(x_{1,\,1}, x_{2,\,1}, \delta u_1, \delta u_2) \\ \sigma_{2,\,1} &= \sigma_{2,\,3/4} + \Delta\sigma_2(x_{2,\,1}, x_{3,\,1}, \delta u_2, \delta u_3) \\ \sigma_{3,\,1} &= \sigma_{3,\,1/2} + \Delta\sigma_3(x_{3,\,1}, x_{4,\,1}, \delta u_3, \delta u_4) \\ \sigma_{4,\,1} &= \sigma_{4,\,0} + \Delta\sigma_4(x_{4,\,1}, x_{5,\,1}, \delta u_4, \delta u_5) \end{aligned} , \tag{75}$$

and for the internal forces:

$$\begin{aligned} F_{1,\,1} &= F_{1,\,1}(\sigma_{1,\,1}) \qquad F_{2,\,1} = F_{2,\,1}(\sigma_{1,\,1}, \sigma_{2,\,1}) \\ F_{3,\,1} &= F_{3,\,1}(\sigma_{2,\,1}, \sigma_{3,\,1}) \\ F_{4,\,1} &= F_{4,\,1}(\sigma_{3,\,1}, \sigma_{4,\,1}) \qquad F_{5,\,1} = F_{5,\,1}(\sigma_{4,\,1}) \end{aligned} \quad . \tag{76}$$

To complete the cycle (and the step), the nodal accelerations and velocities must be updated for all nodes. The accelerations are:

$$\begin{aligned} a_{1,\,1} &= -F_{1,\,1}/m_1 \qquad a_{2,\,1} = -F_{2,\,1}/m_2 \\ a_{3,\,1} &= -F_{3,\,1}/m_3 \\ a_{4,\,1} &= -F_{4,\,1}/m_4 \qquad a_{5,\,1} = -F_{5,\,1}/m_5 \end{aligned} \quad . \tag{77}$$

Since we are at the end of the macro step, it is convenient to express the new velocities at the same time as the other quantities, i.e. at t_1. In other words, they are full-step values and not mid-step values (so that e.g. the kinetic energy may be evaluated); they will again be updated to mid-step values at the beginning of the next macro step (see eq. 47). Since $\psi_1 = \psi_2 = 4$, $\psi_3 = 2$, $\psi_4 = \psi_5 = 1$,

we obtain (note again the factors $1/2$ like in eq. 47 for the step initialization, and unlike eq. 59, which is the one used during the intermediate cycles 1, 2 and 3):

$$\begin{aligned}
v_{1,1} &= v_{1,7/8} + \frac{1}{2}(\Delta T/\psi_1)a_{1,1} = v_{1,7/8} + \frac{1}{2}\tau a_{1,1} \\
v_{2,1} &= v_{2,7/8} + \frac{1}{2}(\Delta T/\psi_2)a_{2,1} = v_{2,7/8} + \frac{1}{2}\tau a_{2,1} \\
v_{3,1} &= v_{3,3/4} + \frac{1}{2}(\Delta T/\psi_3)a_{3,1} = v_{3,3/4} + \tau a_{3,1} \ , \\
v_{4,1} &= v_{4,1/2} + \frac{1}{2}(\Delta T/\psi_4)a_{4,1} = v_{4,1/2} + 2\tau a_{4,1} \\
v_{5,1} &= v_{5,1/2} + \frac{1}{2}(\Delta T/\psi_5)a_{5,1} = v_{5,1/2} + 2\tau a_{5,1}
\end{aligned} \tag{78}$$

This terminates the fourth and last cycle (t_1) of the time integration procedure, see Figure 5. The final situation is depicted in Figure 9(b).

As a final remark, it should be noted that the algorithm as detailed above is slightly simplified with respect to the actual procedure implemented in the code. The main simplification lies in the fact that for simplicity the partition has been tacitly considered constant during the macro step (“static” partition), while in reality it should be re-evaluated at each cycle because, due to large deformations, an element may jump from a level to another one (upper or lower) not only at the end of each macro step but *within* the macro step itself, i.e. at the end of each cycle (“dynamic” partition). More details on the full (dynamic) partitioning algorithm are given in the next Section.

3.4 Partitioned time integration algorithm

In this Section a detailed description of the time integration algorithm with spatial partitioning is given.

3.4.1 Nomenclature

To help the reader, the various quantities used in the time integration algorithm are summarized in Table 2. The following conventions have been followed:

- A quantity related to a partition level uses an upper-case symbol (e.g. T) while the same quantity related to an element or a node uses the corresponding lower-case symbol (t).
- Variations are indicated by an upper-case delta (Δ) when referred to a partition level or to an element, while they use a lower-case delta (δ) when referred to a node.
- Subscript i indicates the generic element ($i = 1, \ldots, N_e$).
- Subscript k indicates the generic node ($k = 1, \ldots, N_n$).

- Subscript l indicates the partition level ($l = 1, \ldots, d$).
- Subscript d indicates the deepest partition level.

3.4.2 Activity function

During the time integration process that will be detailed below, use is made of two integer quantities that will be denoted M and m, whose precise meaning is defined hereafter. The M quantity is simply the current maximum level frequency of the partition:

$$M \equiv \Phi_d \equiv 2^{d-1}, \tag{79}$$

where d is the current partition depth. The quantity m will be denoted as activity function and is defined as the minimum level frequency that is currently active. Its meaning and relevance are best explained by an example.

Consider one "macro" step of the time integration procedure with a partition of depth $d = 5$, as shown in Figure 10. Assume for simplicity that the partition depth d is constant over this macro step, so that M is also constant and has the value $\Phi_d = 2^{d-1} = 2^4 = 16$. This means that the elements in the lowest partition level will have to perform 16 "cycles" or micro steps within the macro step, i.e. while the elements in the highest level perform just one step. Correspondingly, the elements in intermediate levels 2 to 4 will perform 2, 4, or 8 cycles, respectively.

Let us introduce a micro-step (or cycle) counter I, which will go from 1 to 16 (i.e. from 1 to M) over the macro step. At each cycle, only some of the partition levels are "active" (i.e. the corresponding elemental or nodal quantities must be updated), while the others are not active. For example, and with reference to Figure 10, when $I = 4$ the levels 3, 4 and 5 are active while the levels 1 and 2 are not. The m variable indicates the level frequency of the first active partition level encountered from the top, i.e. the level with the minimum index among the currently active ones. All levels below and including this one are active, while all levels above this one are not active.

One may easily verify (see also the table contained in Figure 10) that $m(I)$ can be computed as:

$$m(I) = M / \kappa(I) = \Phi_d / \kappa(I), \tag{80}$$

where $\kappa(I)$ is the highest power of 2 'contained' as a factor in I. Consequently, for odd values of I, it is simply $m = M$, i.e. the first and only active level is the deepest one. For even values of I it is $m < M$ and in particular for $I = M$ (and only for this value) it is $m = 1$.

The activity function $m(I)$ exhibits the characteristic pattern shown in Figure 10. The dashed zone highlights the partition levels that are active at any given cycle.

3.4.3 Time loop algorithm

We are now in a position to give a formal outline of the time integration process with spatial partitioning. This is denoted below as Algorithm P. For completeness the algorithm assumes an ALE formulation, of which the Lagrangian case—which is the only one considered in the present paper—may be seen as a subset. The parts relative to an Eulerian or to a full ALE description are discussed in Part II.

Algorithm P

0. Initialization: set the time $t \leftarrow t_0$ (initial time), macro step counter $N_p \leftarrow 0$, $I \leftarrow 0$, $M \leftarrow 0$, $m \leftarrow 1$; $\delta t_k \leftarrow \underline{0}$ for $k = 1, \ldots, N_n$; the initial velocities $\underline{v}_k \leftarrow \underline{v}_{k0}$ for $k = 1, \ldots, N_n$, $\Delta t_{\min} \leftarrow 0$, $\Delta t_{\max} \leftarrow 0$, $M_s \leftarrow 0$, the initial displacements $\underline{u}_k \leftarrow \underline{0}$ for $k = 1, \ldots, N_n$ (for simplicity), and the initial configuration $\underline{x}_k \leftarrow \underline{x}_{k0}$ for $k = 1, \ldots, N_n$.

1. **GO TO point 8** below.

2. Increment the macro-step counter: $N_p \leftarrow N_p + 1$.

3. Update the velocities: $\underline{v}_k \leftarrow \underline{v}_k + \delta\tau_k \cdot \underline{a}_k$, where $\delta\tau_k = \delta t_k / 2$. The operation is performed for all nodes, since we are at the beginning of a macro step. Velocities are now expressed at the mid-step (i.e., at time instants that vary in general from node to node).

4. Increment time by a cycle: $t \leftarrow t + \Delta t_{\min}$; increment the cycle counter: $I \leftarrow I + 1$. Note, however, that $m(I)$ is not updated until point 7 below. Thus, the m used at points 5 and 6 is the one relative to the previous cycle, i.e. the cycle that led to the current time t.

5. If an Arbitrary Lagrangian Eulerian (ALE) formulation has been chosen, then update the grid velocity $\underline{w}_k$ of all nodes k in an active partition level during the previous cycle, i.e. such that $\psi_k \geq m$. In general $\underline{w}$ is a function of the current configuration $\underline{x}$ and of the current velocity $\underline{v}$, see Part II for details.

6. For all nodes k in an active partition level during the previous cycle, i.e. such that $\bar{\psi}_k \geq m$: A) update the Lagrangian displacement increments $\delta\underline{u}_k = \delta\bar{\tau}_k \cdot \underline{v}_k$ where $\delta\bar{\tau}_k = (\Phi_d / \bar{\psi}_k)\Delta t_{\min} = \Delta T / \bar{\psi}_k$; B) in an ALE or Eulerian calculation, update also the relative displacement increments $\delta\underline{u}_k' \leftarrow \delta\bar{\tau}_k \cdot (\underline{v}_k - \underline{w}_k)$ (note that in the Eulerian case it is simply $\underline{w} \leftarrow \underline{0}$); C) compute the displacement increments: in the Lagrangian case it is $\delta\underline{x}_k \leftarrow \delta\bar{\tau}_k \cdot \underline{v}_k = \delta\underline{u}_k$, while in the Eulerian or ALE cases $\delta\underline{x}_k \leftarrow \delta\bar{\tau}_k \cdot \underline{w}_k$; D) update the total displacements: $\underline{u}_k \leftarrow \underline{u}_k + \delta\underline{x}_k$; E) update the current configuration: $\underline{x}_k \leftarrow \underline{x}_k + \delta\underline{x}_k$.

7. Update $m(I)$ by eq. (80), i.e. $m(I) \leftarrow \Phi_d / \kappa(I)$, where $\kappa(I)$ is the highest power of 2 'contained' as a factor in I.

8. Initialize the internal forces and the external forces: $\underline{F}_k^{\text{int}} \leftarrow \underline{0}$, $\underline{F}_k^{\text{ext}} \leftarrow \underline{0}$. Compute and assemble the internal forces at the current time by updating all currently active elements. See details in Section 3.4.4.

9. If we are at the end of a macro step ($I = M$), then: A) check the minimum and maximum elemental time increments $\Delta t_{\min}$ and $\Delta t_{\max}$ that were computed at the end of the elements update process (see point 8. above and Section 3.4.4), against any restraints (on $\Delta t_{\max}$) that might be imposed by the user or that might result from the chosen results printout and storage times; B) check whether it is worthwhile to partition: if $\Delta t_{\max}/\Delta t_{\min} \leq 1.7$ then set $\Delta t_{\max} \leftarrow \Delta t_{\min}$, thus de-activating *de facto* the partition, as may be verified by inspecting the partition updating algorithm, see Section 3.4.8 below (point 3 of Algorithm P.10). Note that, since both I and M are initialized at 0, these checks are performed also at step 0 (i.e. at the initial time of the calculation).

10. Compute the data structure relative to the partition, as detailed below in Section 3.4.8. This includes, among other things, the calculation of level frequencies: φ, ψ, $\overline{\varphi}$ and $\overline{\psi}$. If we are at the end of a macro step ($I = M$), then reset $I \leftarrow 0$ and re-compute M.

11. For all nodes k in a currently active partition level, i.e. such that $\psi_k \geq m(I)$, evaluate the prescribed external forces at the current time.

12. For all nodes k in a currently active partition level, i.e. such that $\psi_k \geq m(I)$, compute the accelerations at the current time: $\underline{a}_k \leftarrow (\underline{F}_k^{\text{ext}} - \underline{F}_k^{\text{int}})/\underline{m}_k$.

13. For all nodes k in a currently active partition level, i.e. such that $\psi_k \geq m(I)$: A) update the velocities: $\underline{v}_k \leftarrow \underline{v}_k + \delta\tau_k \cdot \underline{a}_k$, where $\delta\tau_k = \delta t_k/2$ if we are starting a macro step (i.e. when $I = 0$), or else $\delta\tau_k = (\delta t_k + \delta t_k^{\text{new}})/2$, with $\delta t_k^{\text{new}} = (\Phi_d/\psi_k)\Delta t_{\min} = \Delta T/\psi_k$. Note that at the initial time of the calculation $\delta\tau_k = 0$ because of the initializations performed in step 0 above. This update may be safely applied even at the initial time, since it will not modify the initial velocities; B) keep track of the time increment by which the nodal velocity is being updated, by setting $\delta t_k \leftarrow \delta t_k^{\text{new}}$.

14. If $I > 0$, then **GO TO point 4** above (next cycle).

15. The macro step is complete. Compute the kinetic energy (for the energy balance). Print results if so requested.

16. If the final time has not been reached, then **GO TO point 2** above (next macro step).

17. End of the calculation.

In the next Sections the various points of the algorithm are further detailed.

3.4.4 Calculation of internal forces

We consider in more detail the calculation of element internal forces, corresponding to point 8 of the Algorithm P presented in Section 3.4.3.

Algorithm P.8

0. Initialization: if we are at the end of a macro step ($I = M$), then initialize a local version of the minimum and maximum elemental time increments: $\Delta\tau_{\min} \leftarrow 1.0\times10^{12}$ and $\Delta\tau_{\max} \leftarrow 0$. Note that, since both I and M are initialized at 0, this calculation is performed also at step 0 (i.e. at the initial time of the calculation).

1. Perform a loop over all the (currently active) elements in the mesh, see Section 3.4.5 for details. Within the loop, the relevant subroutine for each element is called, according to its type. This element-specific subroutine computes the element's contribution to internal forces and estimates the element's stability time step (Δt_s). Then it computes the "safe" step for the element, Δt, as $\Delta t \leftarrow \Delta t_s C_s$ (the former being the element's stability time step and the latter being the safety factor). If we are at the end of a macro step ($I = M$), then the value of Δt is stored in a table Δt_i and is used to update the value of $\Delta\tau_{\min}$ and $\Delta\tau_{\max}$. Otherwise (i.e. *within* the macro step), a check is done to see whether the assignment of the element to the current partition level is still valid and, if necessary, it is changed (see details in Section 3.4.7).

2. Set the current value of the stability step: $\Delta t_e \leftarrow \Delta\tau_{\min}$.

3. If we are at the end of a macro step ($I = M$), then update the values of the minimum and maximum elemental time increments: $\Delta t_{\min} \leftarrow \Delta\tau_{\min}$ and $\Delta t_{\max} \leftarrow \Delta\tau_{\max}$, respectively.

4. End of the calculation of internal forces.

3.4.5 Update of an element

We consider now in detail the updating of a generic element, corresponding to point 1 of Algorithm P8 that was presented in Section 3.4.4.

Algorithm P.8.1

0. Initialization: set Δt_s to a very large value ($\Delta t_s \leftarrow 1.\times10^{12}$). Let the index of the current element be i.

1. If we are at the beginning of the calculation ($N_p = 0$), set $\bar{\varphi}_i \leftarrow 1$ (this initialization is necessary for the subsequent filling of local arrays of nodal variables at step 0, see Section 3.4.6 below for details) and continue, i.e. go to the next point; otherwise, check $\bar{\varphi}_i$. If $\bar{\varphi}_i < m$, then the current element is inactive at the present cycle, so we skip its calculation altogether (**GO TO point 7**).

2. Fill in local (element-specific) arrays of nodal variables to be passed to the element subroutine (see Section 3.4.6 below for details).

3. Call the element-specific subroutine, according to the current element's type. This subroutine computes the element's contribution to internal forces and estimates the element's stability step Δt_s.

4. Set $\Delta t \leftarrow \Delta t_s C_s$.

5. If we are at the end of a macro step ($I = M$), then store the element's time increment: $\Delta t_i \leftarrow \Delta t$; and use the value of Δt to update $\Delta\tau_{\min}$ and $\Delta\tau_{\max}$: $\Delta\tau_{\min} \leftarrow \min(\Delta\tau_{\min}, \Delta t)$ and $\Delta\tau_{\max} \leftarrow \max(\Delta\tau_{\max}, \Delta t)$.

6. Otherwise (i.e. *within* a macro step), check whether the current element (with its newly calculated Δt) still "fits" into the partition level to which it belongs and, if necessary, modify the partition (see details in Section 3.4.7 below).

7. End of calculation for the current element.

3.4.6 Filling in local arrays of an element's nodal variables

As pointed out in step 2 of Algorithm P.8.1 above, the code prepares local (element-specific) arrays of the most relevant nodal variables before calling the element-specific subroutine, basically by copying the values from the corresponding global arrays. The concerned variables include fluxes, coordinates, displacements, forces, relative displacements, nodal masses, velocities, relative velocities and total displacements. This preparatory work is done in a dedicated subroutine and the resulting arrays are then passed to the element-specific subroutine, which may thus treat the element's node numbering as if it were local rather than global. This considerably simplifies the programming of the element subroutine.

Hereafter we examine in some detail the variables that are related to partitioning (i.e., only the displacement increments and the relative displacement increments).

Algorithm P.8.1.2

0. Initialization: compute the element's time increment to be used for displacement/configuration update as: $\Delta\bar{t}_i \leftarrow (\Phi_d / \bar{\varphi}_i)\Delta t_{\min} = \Delta T / \bar{\varphi}_i$.

1. Compute the array of local displacement increments as: $\delta\underline{u}_l \leftarrow \Delta\bar{t}_i \delta\underline{u}_k$ where k is the global node index corresponding to local node index l.

2. If the chosen formulation is Eulerian or ALE, then a similar operation is performed also for the relative displacement increments: compute the array of local relative displacement increments as: $\delta \underline{u}_l' \leftarrow \Delta \bar{t}_i \delta \underline{u}_k'$ where k is the global node index corresponding to local node index l.

3. End of filling local arrays.

By examining point 0 of the above algorithm, it becomes clear why one has to set $\bar{\varphi}_i \leftarrow 1$ at point 1 of Algorithm P.8.1 in Section 3.4.5 above. This is only to avoid a division by zero when computing $\Delta \bar{t}_i$ at step 0. The actual value chosen to initialize $\bar{\varphi}_i$ (here 1 is chosen for convenience) does not matter, provided it is not zero. In fact, since at step 0 both Φ_d and $\Delta t_{\min}$ are initialized at zero, the result is $\Delta \bar{t}_i = 0$.

This produces the desired effect in points 1 and 2 of the algorithm, i.e. at step 0 both the displacement increments and the relative displacement increments are zero.

3.4.7 Checking the partition

As pointed out in step 6 of Algorithm P.8.1 above, during the sub-cycling process within each macro step, the stability time increment of elements will in general change. Therefore, it is necessary to continuously check that each element "fits" within the partition level to which it currently belongs and, if necessary, move it to a different level (or even update the whole partition).

This is accomplished by the following algorithm.

Algorithm P.8.1.6

0. If $\varphi_i < m$, then we skip this check altogether (**GO TO point 6**).

1. If $\Delta t \geq (1-\varepsilon)\Delta t_i$ where ε is a small tolerance (typically 1×10^{-6}), then the element's time increment is not decreasing (or at least, not significantly), so we may skip this check (**GO TO point 6**).

2. Here we are *within* a macro step and the current element's time increment has decreased with respect to the value it had previously. We compute the time increment associated with the level to which the element currently belongs: $\Delta T_l \leftarrow \Delta t_{\min} M / \varphi_i$. If $\Delta t \geq \Delta T_l$, then the element still fits in the current partition level, so we may skip this check (**GO TO point 6**).

3. We set a flag $\mu \leftarrow 1$. This means that the partition has been modified and needs being re-computed as detailed below in Section 3.4.8.

4. We set $\varphi_i \leftarrow \varphi_i + \varphi_i$, $\Delta t_i \leftarrow \Delta t_i / 2$ and $\Delta T_l \leftarrow \Delta T_l / 2$; if $\varphi_i > M$ then a new (lowest) level of the partition needs to be created, so the depth of the partition is increased by 1, both I and M are

doubled (by the way, this guarantees that they are *even*) and $\Delta t_{\min}$ is halved: $I \leftarrow I + I$, $M \leftarrow M + M$, $\Delta t_{\min} \leftarrow \Delta t_{\min}/2$.

5. If $\Delta t < \Delta T_l$, then the element still doesn't fit in the level, so **GO TO point 4**.

6. End of the partition check.

3.4.8 Updating the partition

Let us finally see in detail how the partition data structure is updated. This is done in a dedicated subroutine which is called at every cycle of the time integration process (see point P.10 in Section 3.4.3).

Algorithm P.10

0. Initialization: set a small tolerance $\varepsilon \leftarrow 1.\times 10^{-6}$.

1. If $I < M$, then **GO TO point 8**.

2. Increment the counter of the total number of cycles $M_s \leftarrow M_s + M$, set $I \leftarrow 0$, $\Phi_d \equiv M \leftarrow 1$ and $\mu \leftarrow 1$, the latter meaning that the partition needs to be updated.

3. If $\Delta t_{\min} < (1-\varepsilon)\Delta t_{\max}$ then compute the partition depth d and the maximum level frequency Φ_d by expressions (29), (30), (31) and (79): start by setting $d \leftarrow 1$ and $\Delta T_d \leftarrow \Delta t_{\max}$, then repeatedly do $d \leftarrow d + 1$, $\Phi_d \leftarrow \Phi_d + \Phi_d$, $\Delta T_d \leftarrow \Delta T_d/2$ until $(1-\varepsilon)\Delta T_d \le \Delta t_{\min}$, and when this is verified set $\Delta t_{\min} \leftarrow \Delta T_d$.

4. Initialize all intrinsic element frequencies to the maximum value just found: $\varphi_i \leftarrow \Phi_d$, for $i = 1, \ldots, N_e$.

5. Initialize all element frequencies to one: $\bar{\varphi}_i \leftarrow 1$, for $i = 1, \ldots, N_e$.

6. If $M \le 1$, i.e. if there is just one level in the partition, then **GO TO point 8**.

7. Compute the intrinsic element frequencies φ_i and the element-specific time increments Δt_i. For each element i: start by setting $\Delta\tau \leftarrow \Delta t_{\min}$, then repeatedly do $\Delta\tau \leftarrow \Delta\tau + \Delta\tau$ until either the iteration counter reaches the partition depth or $\Delta\tau > (1+\varepsilon)\Delta t_i$. As long as neither of the two above conditions is verified, do $\varphi_i \leftarrow \varphi_i/2$. When one of the conditions is verified, terminate by setting $\Delta t_i \leftarrow \Delta\tau/2$.

8. If $\mu = 0$, then **GO TO point 14**.

9. Complete the updating of the partition: since the φ_i have been recomputed, the other level frequencies must be re-computed as well. Initialize by setting $\mu \leftarrow 0$, $\psi_k \leftarrow 1$, $\bar{\psi}_k \leftarrow 1$ for $k = 1, \ldots, N_n$.

10. If $M \le 1$, i.e. if there is just one level in the partition, then **GO TO point 14**.

11. Take into account the effect of links (see eqn. 37): for each node k subjected to a link, i.e. such that $\lambda_k = 1$, set $\psi_k \leftarrow \Phi_d \equiv M$.

12. Compute the intrinsic nodal frequencies: loop over all the elements $i = 1, \ldots, N_e$, and for each element loop over its nodes $k = 1, \ldots, N_n^e$ by setting $\psi_k \leftarrow \max(\psi_k, \varphi_i)$ according to eqn. (35).

13. Compute the neighboring element frequencies and the neighboring node frequencies: loop over all the elements $i = 1, \ldots, N_e$, and for each element: A) loop over its nodes $k = 1, \ldots, N_n^e$ by setting $\bar{\varphi}_i \leftarrow \max(\bar{\varphi}_i, \psi_k)$ according to eqn. (38); then B) loop again over the element's nodes $k = 1, \ldots, N_n^e$ by setting $\bar{\psi}_k \leftarrow \max(\bar{\psi}_k, \bar{\varphi}_i)$ according to eqn. (40).

14. End of partition update.

This completes the description of the spatial partitioning algorithm. This algorithm is indeed somewhat more complicated (but only conceptually!) than the version with spatially uniform time increment described in Section 2, see Table 1. However, it is important to underline that it may be implemented in an extremely compact way and without compromising in any way the full generality of the basic explicit formulation. In fact, the spatial partitioning does not inherently modify the informatic data structure and data flow as far as a standard finite element explicit code is concerned.

The only potential pitfall lies in the correct estimate of limiting time steps for the various element types, which must typically be improved with respect to the case with uniform time increment. In fact, in the presence of partitioning all elements in the model, and not only a small minority, are updated by time increments which are close to their local stability values, so that a precise estimate of these values becomes critical.

4. Simple numerical examples

The numerical examples presented in this Section are intentionally very simple. They aim primarily at validating the partitioning technique by showing that it preserves the robustness and quality of results obtained with the non-partitioned version of the explicit time integration scheme.

The first example is one-dimensional and involves only small deformations, so that an analytical solution may be obtained for comparison. The second example is the well-known Taylor bar impact and involves very large deformations in a 2-D axisymmetric geometry. More complex and realistic examples will be presented in Part II.

4.1 1-D wave propagation

The first test problem concerns the propagation of elastic longitudinal waves in a one-dimensional bar, see Figure 11. The whole bar is initially moving at a velocity $v_0 = 100$ m/s and, starting at the initial time, its right extremity P2 is blocked so that a compression wave develops and then propagates along the bar towards the left extremity P1.

The initial length of the bar is 1.0 m and the cross-section is assumed to be uniform. The material is linear elastic, with density $\rho = 8000$ kg/m^3, Young's modulus $E = 2.0\times10^{11}$ Pa and Poisson's coefficient $\nu = 0$.

The bar model is discretized by means of two-noded bar-like elements. In order to compare with the purely one-dimensional analytical solution, special options are activated in the code that remove the non-linear effects which would be included by default in the numerical solution:

- a small-strain formulation is adopted, whereby strain increments are referred to the initial—and not to the current—element length;
- element cross-sections are kept constant and the initial element length rather than the current one is used to compute the element's intrinsic time increment, which therefore stays constant.

The analytical solution of this problem in terms of stress consists of a step-like compression wave:

$$\sigma = \frac{v_0}{c}E, \tag{81}$$

which propagates at the speed of sound c in the material:

$$c = \sqrt{\frac{E}{\rho}}. \tag{82}$$

With the assumed values of the parameters, we obtain $\sigma = -4.0\times10^9$ Pa and $c = 5000$ m/s. The step wave therefore reaches the mid-point of the bar P3 at time:

$$t_3 = \frac{L}{2c} = 1.0\times10^{-4} \text{ s}. \tag{83}$$

The wave reaches the left extremity P1 at time $2t_3$, reflects as a traction wave which cancels out the incoming compression wave, reaches again P3 at $3t_3$ and returns to the right extremity at $4t_3$. This is assumed as the final time of the simulation because after this time the bar would rebound as a rigid body at velocity $-v_0$, in the assumption that the rigid obstacle is unilateral, while here for simplicity a bi-lateral constraint (full blockage) is assumed at the right extremity.

To further simplify, in this example the blockage condition is modelled by using a specific, "direct" method, instead of using the much more general boundary condition model (the so-called "links"

formulation) based upon Lagrange multipliers that will be described in Part II. This simplification is possible since the constraint is uncoupled, i.e. it involves degrees of freedom (dofs) of only one node—actually, just one dof in this case. Roughly speaking, it consists in evaluating the internal force F^{int} at the concerned node, and in imposing an external force (reaction) with the same value ($F^{\text{ext}} = F^{\text{int}}$), so that from Eq. (1) the resulting acceleration is null and, if the node is initially at rest, it stays blocked during the whole transient.

When a calculation with partitioning is performed, any nodes subjected to these particular blocking conditions are treated as "free" nodes, i.e. Eq. (35) is applied instead of Eq. (36). This technique, which is only applicable to extremely simple test cases such as the ones treated in the present paper, almost completely avoids any overhead in the treatment of boundary conditions and therefore allows to measure more precisely the efficiency of the "core" partitioning algorithm.

4.1.1 Uniform-mesh solutions without partitioning

The first numerical solution obtained for this test problem uses a uniform discretization with $N_e = 100$ elements of length 0.01 m each and a constant time increment of 1.6×10^{-6} s. This corresponds to a safety factor $C_s = 0.8$ over the stability value $\Delta t^{\text{crit}} = L/(N_e c) = 1/(100 \cdot 5000) = 2.0\times10^{-6}$.

The computed stress and velocity at point P3 (dashed lines) are compared in Figure 12 with the corresponding analytical solutions (solid lines). Two numerical stress curves are shown, corresponding to the elements immediately before and immediately after P3 (and sharing P3 as a common node), while for the velocity the numerical result at node P3 is available and is shown alone. As expected, the agreement with the analytical solution is very good. Characteristic of these explicit solutions are the numerical precursor (as already mentioned in Section 3.1.4) and the oscillations behind the step, which rapidly converge onto the analytical value. The latter phenomenon is related to spatial and time discretization and is typical of linear elastic solutions, as will be discussed below.

The next solution uses a much finer uniform mesh of 800 elements of length 1.25×10^{-3} m each and a constant time increment of 0.2×10^{-6} s, corresponding to the same safety factor $C_s = 0.8$ over the stability value as in the previous case. Results in term of velocity are shown in Figure 13a. It may be observed that, as expected, the wave front is steeper, the oscillations have a much higher frequency and the solution converges more rapidly onto the analytical value than in the coarse-mesh case. These effects may be better appreciated in Figure 13b, which shows a zoom-in of the first step region for the two solutions obtained so far.

4.1.2 Gradual-mesh solution without partitioning

For the third solution, we take a progressively refined mesh consisting (from left to right) of 97 elements of length 0.01 like in the first solution, followed by 2 elements of length 0.005, then by 4 elements of length 0.0025 and finally by 8 elements of length 1.25×10^{-3} like in the second solution. Altogether, there are 111 elements. This case simulates localized mesh refinement in the vicinity of the impacting end, which is typical of many realistic applications.

The solution is obtained by using a constant time increment of 0.2×10^{-6} s, corresponding to the same safety factor $C_s = 0.8$ over the stability value as in the previous case, if referred to the smallest elements in the mesh (i.e. the last 8 elements), but to a safety factor of only 0.1 for the first 97 elements, i.e. the largest ones, which are by far the majority.

The computed velocity is shown in Figure 14 compared with the previous solutions. The result is more similar to the coarse-mesh solution than to the fine-mesh one, as might perhaps be expected since the majority of elements in the gradual mesh are coarse. However, perhaps surprisingly at first sight, the third solution does not lie "between" the first two solutions, but exhibits oscillations with a frequency even slightly lower than that of the coarse mesh solution.

Influence of stability safety factor

This apparent discrepancy may be explained by the fact that, while the first two solutions are obtained by using the same value for the stability safety factor $C_s = 0.8$, the third one in practice uses different safety factors in the same calculation, ranging from 0.8 down to 0.1 depending upon the elements. To study the effect of the safety factor on numerical solutions, a series of calculations are performed, all using the same mesh as the coarse solution (100 elements) and values of C_s ranging from 1.000 to 0.010. Results are presented in Figure 15a and raise several remarks.

First of all, the solution with $C_s = 1.000$, i.e. using the theoretical stability limit, is indeed stable and, quite remarkably, coincides exactly with the analytical solution, i.e. there are no oscillations. The step in the velocity curve appears less steep than the perfectly vertical analytical curve in the drawing, but this is simply due to the time discretization in the numerical solution: only one value at each given time is available. The fact that a "perfect" numerical solution like this one may be obtained by simply using a unit safety factor is well known. However, this result has only an academic interest since in realistic cases meshes are not uniform, so it is impossible to use $C_s = 1.000$ everywhere in non-partitioned solutions. However, one may note incidentally that in solutions with partitioning the effective safety factor is much closer to one on all the elements than in uniform-step calculations, so that the quality of the ideal solution is better approximated.

Returning to the parametric study shown in Figure 15a, one sees that, as the safety factor decreases from the limit value, numerical oscillations of increasing amplitude (the so-called elastic "overshoot") and of decreasing frequency appear. The effect is relatively pronounced for C_s between 1.000 and 0.500, but for values of C_s below 0.250 it almost disappears and all solutions are practically identical.

By considering the curves for $C_s = 0.800$ and for $C_s = 0.100$ in Figure 15a we see exactly the same difference observed in Figure 14b between the coarse-mesh solution and the gradual-mesh solution. This indicates that the above-mentioned apparent discrepancy is simply due to the fact that in the gradual-mesh case the "average" effective stability safety factor is (much) closer to 0.100 than to 0.800. Evidently, the presence of many large elements which are integrated with a rather low effective safety factor is preponderant with respect to the few small elements which use a large effective safety factor.

Influence of mesh size

In order to complete this parametric study, it is also interesting to investigate the effect of mesh size on numerical solutions. To this end, a series of solutions are obtained with uniform meshes of 10, 20, 50, 100, 200 and 1000 elements, all using the same safety factor $C_s = 0.050$. A low value of C_s is assumed so as to be in a zone where its influence is almost irrelevant, as shown from the previous parametric study.

Results are presented in Figure 15b, where it may be seen that, as the mesh gets finer and finer, the numerical velocity step becomes steeper, the frequency of numerical oscillations behind it increases, and the amplitude of the elastic overshoot increases as well. This behavior is consistent with the fact that a spatially discrete model may only represent a finite range of frequencies, while the perfectly vertical, analytical step solution contains infinite frequencies. Only in the limit of a vanishing mesh size (i.e. of an infinite number of elements) can the discrete system approach the analytical solution.

4.1.3 Gradual-mesh solution with partitioning

For the fourth and final solution, we consider again the gradual mesh with 111 elements used in the third solution, but we activate the spatial partitioning algorithm, by using a stability safety factor $C_s = 0.800$, i.e. the same value used in the first two solutions.

There are four levels in the partition and the intrinsic element frequencies φ_i are in this case: 1 for the first 97 elements, 2 for the following two elements, 4 for the following four elements and 8 for the last eight elements. As a consequence of the options activated to linearize the solution, which have been already mentioned at the beginning of Section 4.1, the intrinsic time increments of the ele-

ments do not change and therefore the partition is “static” in the present example: all frequencies are constant in time and elements and nodes always belong to the same level during the whole transient.

The obtained stress and velocity curves are shown in Figure 16. The solution is stable and presents the same overall characteristics already observed in the solutions with spatially uniform time step, thus indicating that the good qualities of the basic explicit time integration algorithm are indeed preserved by the proposed spatial partitioning technique. For a finer comparison, the partitioned and non-partitioned solutions obtained with the same gradual mesh are plotted together in Figure 17a. In the light of the previous parametric studies we may observe that, as it might be expected, the partitioning process shifts the solution towards larger “equivalent” stability safety factors: steeper wave front, smaller overshoot, larger oscillations frequency.

To conclude, it is interesting to compare the above two solutions against the coarse uniform mesh (100 elements) solution with the same stability safety factor $C_s = 0.800$, see Figure 17b. The result with gradual mesh and partitioning is in fact very close to the coarse mesh without partitioning, if one only neglects the fact that the partitioned solution appears somewhat jaggier because results are plotted only at “macro” time instants. This agreement is again very reassuring about the quality of the partitioning technique. In fact, there is no reason why the two numerical solutions should be different since, on one hand, we consider results at a point (P3) relatively far from the refined mesh zone and, on the other hand, the “equivalent” stability safety factor is as already observed almost the same in the two cases.

The reader might be curious about the comparison of non-partitioned and partitioned calculations in terms of CPU time. While the partitioned solution is indeed (much) faster than the non-partitioned one, the total CPU time needed to solve this first test problem is so small that a quantification of speed-up would be at best very inaccurate. This exercise is therefore left for the next numerical example, which is slightly more time-consuming although still of an academic nature, and even better for the examples in Part II.

The four numerical solutions described above are summarized in Table 3. The following additional remarks may be raised:

- As expected, solutions (2) and (3) need 8 times more steps (2000 instead of 250) than solution (1) to reach the final time of the calculation, because in both cases the shortest element is 8 time smaller.
- The solution with partitioning (4) performs roughly the same number of macro steps as the coarse-mesh solution (1), and roughly the same number of (sub-)cycles as the fine-mesh solutions (2) and (3).

- By letting the code record the total number of "elements × cycles", i.e. the total number of element update operations performed (see the fore last column of Table 3), one obtains a first-approximation indicator of the (relative) cost of calculations and thus of the "theoretical" speed-up, defined here as the cost of the uniform-step solution divided by the cost of the partitioned solution. The actual speed-up, however, is usually smaller than the theoretical one because of the overhead necessary to manage the partitioned itself.
- The CPU times, although reported in Table 3 for completeness, are not indicative of relative performance because of the shortness of these calculations.

4.2 Taylor bar impact

The second test problem is the well-known Taylor bar impact, see Figure 18, frequently used (with many variations) in the literature to assess numerical models, see e.g. Hallquist and Benson [13]. Apparently G.I. Taylor was the first to recognize, in 1948, that the final length and shape of a cylinder impacting against a rigid surface (anvil) are quite sensitive to the constitutive behavior of the material [14].

Although no analytical solution exists for such problems—and experimental results may be affected by complex phenomena such as friction between the cylinder and the anvil—an extremely accurate numerical solution for the idealized problem (i.e. without friction) can be obtained, at a high CPU cost indeed, with the Lagrangian formulation and a very fine mesh, and can then be used to assess the quality of other solutions.

The data of the problem considered here are taken from reference [13]: a cylindrical bar of radius 3.2 mm and length 32.4 mm impacts a rigid, frictionless wall, at an initial velocity of 227 m/s. The material is assumed elastoplastic with isotropic hardening, with Young's modulus $E = 117$ GPa, Poisson's ratio $\nu = 0.350$, yield stress $\sigma_y = 400$ MPa, density $\rho = 8930$ Kg/m^3 and plastic modulus $E_P = 100$ MPa. The final time of the simulation is $t_1 = 80$ μs.

In 1987, the authors presented in reference [15] a 2D axisymmetric solution of this problem, obtained by a 9-node parabolic element and the Lagrangian description with a mesh of 3×30 elements. The final height ($H_1 = 21.47$) and radius ($R_1 = 7.113$) of the projectile were in very good agreement with the values found by Hallquist and Benson in [13] using various 2D and 3D, finite-difference and finite-element codes. More recently, the solution was repeated using a slightly more precise implementation of the constitutive law, and resulted in a slightly different final height ($H_1 = 21.41$) and radius ($R_1 = 7.203$).

In order to find an even more precise reference solution, the Lagrangian solution with the quadratic element was repeated by using finer meshes, until a satisfactory convergence (stabilization) of the results was obtained, see [16]. A calculation with 6×60 elements yielded $H_1 = 21.41$ and $R_1 = 7.218$, while a model with an extremely fine mesh of 12×120 elements resulted in $H_1 = 21.41$ and $R_1 = 7.230$. The latter was considered an extremely precise solution and was adopted as a reference for the calibration of Arbitrary Lagrangian Eulerian methods in solid mechanics presented in the same reference [16].

4.2.1 Solution without partitioning

For the scope of the present paper, a relatively coarse initially regular mesh of 5×50 4-noded quadrilateral elements is assumed. Only one half of the geometry (i.e. the region to the right of the symmetry axis in Figure 18) is modelled, as is usual in 2D axisymmetric calculations.

Although it is not particularly relevant for the present paper, one may note incidentally that two versions of the quadrilateral element are actually used: a reduced-integrated version, with just one integration point, and a fully integrated version, with a 2×2 Gauss integration rule. Using only reduced-integrated elements would lead to spurious mechanisms in the solution, while using only fully-integrated elements would produce some numerical locking. A mixture of the two elements (1 fully-integrated element every 11 reduced-integrated ones) seems to avoid both problems and yields a good solution.

First, a solution with spatially uniform time increment is obtained, which is then used as a reference to validate the solution with partitioning. All solutions are summarized in Table 4. They all adopt the same stability safety factor $C_s = 0.500$. For simplicity, the boundary conditions are modelled as follows: all nodes along the base of the projectile, which at the initial time $t_0 = 0$ come into contact with the rigid target, are blocked in the axial direction. Thus, the calculation simulates a friction-less contact without rebound (like in the first numerical example). Furthermore, all nodes on the axis of symmetry are blocked in the radial direction. All blockages are imposed by the same direct (uncoupled) method already used in the first test case and described in Section 4.1.

In this test the elements close to the projectile tip become heavily distorted and the overall stability time step decreases drastically during the transient calculation. This penalizes the solution with spatially uniform time increment, which needs to perform almost 60000 steps in order to arrive at the final physical time, see Table 4. Therefore, despite the fact that the mesh is initially uniform and all the elements have the same initial time increment, this test case lends itself well to partitioning.

4.2.2 Solution with partitioning

The second solution assumes the same data as the first one, but the spatial partitioning mechanism is activated. In this case the code performs only 1162 macro steps (but 85537 cycles) to arrive at the final time. The partition reaches a maximum depth $d = 7$ in the final part of the calculation, corresponding to a maximum level frequency $\Phi_d = 128$, as seen in Table 4. This allows a substantial saving of CPU time. The listed CPU times do not include the CPU needed for post-processing, but contain the time used for producing normal output (listing, log file, results storage files).

A warning is necessary: the speed-up factors listed in the Table are only indicative, firstly because all the test cases considered in this Section are quite short and secondly because the run time for the same calculation may undergo relatively large variations (up to, say, 10%) depending on the machine load at the time of execution.

The results obtained in this calculation are virtually identical to those of the corresponding case without partitioning. See e.g. Figure 19 which represents the final distribution of yield stress in the two cases (for the chosen material law, this quantity is directly related to the equivalent plastic strain), and Figure 20 which compares the displacements and the velocities at points P1 and P2 of the projectile, which are indicated in Figure 18. In Figure 20 the curves for the two solutions are almost perfectly superposed (including the wiggles), so that they may be hardly distinguished in the drawing.

One sees therefore that, as soon as the numerical simulation contains some dissipative phenomena such as the material plasticity of the present example (and also of the vast majority of practical applications), the solutions with and without spatial partitioning should be practically identical. The differences observed in the first test problem—which, recall, affected only the numerical oscillations, and not on the “engineering” result, as discussed in Section 4.1—are due only to the fact that the central difference time integration scheme, both in its non-partitioned and in its partitioned form, introduces no damping in the solution, so that when the material is purely elastic one gets different numerical oscillations.

A comparison of time steps in the two solutions is shown in Figure 21a. As may be seen, in the case with partitioning the “micro” time step (or sub-cycle) is always smaller than or equal to the time step in the solution without partitioning, but the “macro” time step remains practically constant in this case (except at the very beginning of the calculation), since some elements do not deform at all. The same drawing is repeated in Figure 21b by using a logarithmic scale for the ordinates, which allows to better observe the range of very small time increments.

The initial decrease of the macro time step during the first 4 μs of the transient deserves some explanation. The reason is simply that initially all elements have the same size, so that the time increment

is spatially uniform. In these conditions, the partitioning algorithm is automatically disabled (see point 9 of Algorithm P in Section 3.4.3) since it would most probably produce a CPU overhead instead of a benefit. However, as soon as a reasonably large ratio (of the order of 1.7) between the largest and the smallest elemental time increments is reached, the partitioning "wakes up" and starts to operate. In the present case this happens at about 4 μs (see point P in Figure 21), where one observes a sudden bifurcation of the minimum and maximum time increments ($\Delta t_{\min}$ and $\Delta t_{\max}$) in the partition. The "spikes" visible in Figure 21a and more clearly in Figure 21b in the curves relative to the solution without partitioning are due to the adaptation of the automatic time step to precisely fit the chosen output times, which have been set to occur every 10 μs.

To conclude, Figure 22 shows the final distributions of the various frequencies (φ, ψ, $\bar{\varphi}$ and $\bar{\psi}$) which determine the spatial partition. The figure represents the logarithm in base 2 of the frequencies rather than the frequencies themselves, i.e. in practice the corresponding partition level minus 1, according to Eq. (33). Note that these distributions are drawn over the initial, underformed geometry of the model rather than on the final geometry. In fact, the impacting end of the projectile becomes so distorted that the single elements would not be visible otherwise. In the element frequency plots (φ and $\bar{\varphi}$), each element is painted in a different color according to its own frequency value so that transitions between elements are sharp, while in the node frequency plots (ψ, and $\bar{\psi}$) the values vary rather from node to node so transitions appear smoother due to interpolation over the elements in the graphical representation.

In this picture one observes very clearly the "spreading out" mechanism of the most critical frequencies onto the neighbor elements and nodes described in Section 3.3.2. For example, the largest intrinsic element frequency φ (level 8) affects just one element, marked as "hot spot" in Figure 22, whereas the largest $\bar{\varphi}$ affects that element and all its neighbors (6 elements altogether).

Similarly, the smallest intrinsic element frequency φ (level 1) also applies to just one element, marked as "cold spot" in Figure 22. Incidentally, in this example most of the elements belong to level 2 of the partition during practically the whole computation (except during the initial 4 μs when all elements belong to the same level). One might wonder whether in such a case there would be room for some extra benefit by slightly changing the partitioning strategy. In fact, it might be convenient to base the partition on a slightly smaller value than the maximum one (say 90% or so). In the present example, this would have the effect of placing most of the elements in level 1, instead of level 2. Although the macro time increment would be smaller (90%), the overall effect might perhaps be favorable. Such alternative partitioning strategies will be tested in the near future.

4.2.3 Solutions with domain decomposition and ALE

To conclude the analysis of the Taylor bar impact test case, it is perhaps interesting to briefly consider other improved-efficiency solutions obtained with alternative methods to the spatial partitioning presented in this paper, which are available in the EUROPLEXUS code.

Two such solutions are shown, see Table 4 (solutions 3 and 4): the first one uses the domain decomposition technique [4-6] which has already been mentioned in the Introduction, while the second one uses an Arbitrary Lagrangian Eulerian (ALE) formulation for solid materials presented in reference [16].

The calculation with domain decomposition uses 6 sub-domains, as shown in Figure 23, and yields results practically identical to those obtained previously, at a CPU cost comparable with the one required by spatial partitioning. However, it should be noted that, while in the EUROPLEXUS code the spatial partitioning technique is completely automatic—it is activated by a simple option in the input data set—the decomposition into sub-domains must be entirely specified by the user and is therefore more demanding in terms of human time. The particular decomposition presented here, which seems to yield the best speed-up obtainable for this example, was in fact chosen among a series of other decompositions which were tested out and were found to be slightly less efficient.

The final calculation uses an ALE formulation for solid materials which allows to keep the computational mesh almost uniform during the whole transient. By avoiding excessive squeezing of the elements at the impacting bar end, the overall critical time step of the calculation (which is spatially uniform, like in solution 1) is much larger than with the Lagrangian formulation. Consequently, the total number of time steps required is 30 times less, and the total CPU cost 15 times less, than for the Lagrangian solution. This solution is even more efficient than those with partitioning and with domain decomposition, which have a speed-up of “only” about 10. However, using ALE introduces approximations in the solution, due to the treatment of transport terms, so that this solution is probably slightly less accurate than the purely Lagrangian ones. In fact, small differences may be appreciated by comparing the right drawing of Figure 24 with the left drawing of the same Figure and with both drawings in Figure 19 (the latter three being practically identical). However, in engineering terms the ALE solution is practically equivalent to the others, as may be seen by comparing the displacements and velocities of all four solutions, see Figure 25.

5. Conclusions

The numerical examples of Section 4 seem to confirm that the “core” spatial partitioning technique proposed in this paper has the potential for obtaining significative reductions of CPU time in tran-

sient explicit analyses, without at the same time losing any of the good properties of the classical uniform-step time integration algorithm.

Since the partitioning mechanism is fully automatic, explicit code users may activate it without any effort, simply by specifying an option in the input data file. The actual speed-up obtained with respect to a uniform-step solution depends of course upon the degree of non-uniformity of the time step in the chosen numerical model.

Partitioned solutions are by construction as reliable and accurate as uniform-step solutions because, according to Section 3.1, the proposed partitioning technique introduces no (additional) approximations in the time integration algorithm. In special cases of concern, e.g. when using for the first time a specific model never tested before with partitioning, there is always the possibility of obtaining also the corresponding non-partitioned solution for comparison. As shown in the examples, the two solutions should be identical except for irrelevant effects—in particular irrelevant from the engineering viewpoint—such as e.g. higher-frequency numerical oscillations in purely elastic cases.

It is important to underline that, besides allowing solution of existing test cases in less CPU time, the partitioning technique has the potential for opening the way to simulations that are simply out of reach with the uniform-step algorithm. A significative change of mentality becomes possible. Experienced users of explicit codes are traditionally quite concerned by mesh size because they know that the cost of a calculation is, roughly speaking, inversely proportional to the size of the smallest element in the model. With spatial partitioning this constraint almost disappears and, at least potentially, one becomes free to refine the mesh locally almost at will to reach the desired precision, by paying only modest CPU overheads.

Of course, in order to actually obtain all these benefits in realistically complex applications, the core spatial partitioning technique presented above is not sufficient. The method must be “industrialized” and several technical aspects of great practical importance must be dealt with. Part II addresses two such aspects, namely the treatment of general fully coupled boundary conditions by a Lagrange multipliers method, including both permanent and non-permanent conditions such as contacts, and the extension to an Arbitrary Lagrangian Eulerian (ALE) formulation suitable for the treatment of fluid and fluid-structure interaction problems.

Another important extension—the application of partitioning to Finite Volume rather than Finite Element formulations—has not been attempted yet but, at least conceptually, it should not present any overwhelming difficulties.

6. References

[1] F. Casadei, J.P. Halleux: "*An algorithm for Spatial Partitioning in Explicit Time Integration — Part II - Treatment of Boundary Conditions and Extension to ALE*", to appear.

[2] T. Belytschko, H.-J. Yen and R. Mullen: "*Mixed Methods for Time Integration*", Computer Methods in Applied Mechanics and Engineering, Vol. 17/18, pp. 259-275, 1979.

[3] T. Belytschko: "*Partitioned and Adaptive Algorithms for Explicit Time Integration*", in Nonlinear Finite Element Analysis in Structural Mechanics, ed. By W. Wunderlich *et al.*, Springer Verlag, 572-584, 1980.

[4] A. Gravouil and A. Combescure: "*Multi-time-step explicit-implicit method for non-linear structural dynamics*", International Journal for Numerical Methods in Engineering, Vol. 50, pp. 199-225, 2001.

[5] B. Herry, L. Di Valentin, A. Combescure: "*An approach to the connection between subdomains with non-matching meshes for transient mechanical analysis*", International Journal for Numerical Methods in Engineering, Vol. 55, pp. 973-1003, 2002.

[6] A. Combescure, A. Gravouil: "*A numerical scheme to couple subdomains with different time-steps for predominantly linear transient analysis*", Computer Methods in Applied Mechanics and Engineering, Vol. 191, pp. 1129-1157, 2002.

[7] V. Faucher, A. Combescure: "*Local modal reduction in explicit dynamics with domain decomposition. Part 1: extension to subdomains undergoing finite rotations*", International Journal for Numerical Methods in Engineering, Vol. 60, pp. 2531-2560, 2004.

[8] V. Faucher, A. Combescure: "*Local modal reduction in explicit dynamics with domain decomposition. Part 2: specific interface treatment when modal subdomains are involved*", International Journal for Numerical Methods in Engineering, Vol. 61, pp. 69-95, 2004.

[9] H. Bung, F. Casadei, J. P. Halleux, M. Lepareux: "*PLEXIS-3C: a computer code for fast dynamic problems in structures and fluids*", 10th International Conference on Structural Mechanics in Reactor Technology, Anaheim, U.S.A., August 14-18, 1989.

[10] F. Casadei, J. P. Halleux: "*An Algorithm for Permanent Fluid-Structure Interaction in Explicit Transient Dynamics"*, Computer Methods in Applied Mechanics and Engineering, Vol. 128, Nos. 3-4, pp. 231-289, December 1995.

[11] F. Casadei, J. P. Halleux, A. Sala, F. Chillè: "*Transient Fluid-Structure Interaction Algorithms for Large Industrial Applications"*, Computer Methods in Applied Mechanics and Engineering, Vol. 190, Nos. 24-25, pp. 3081-3110, March 2001.

[12] S.W. Key: "*Transient Response by Time Integration: Review of Implicit and Explicit Operators*", in: J. Donea (Ed.), Advanced Structural Dynamics, Applied Science, London, pp. 71-95, 1980.

[13] J.O. Hallquist, D.J. Benson: ''*DYNA-3D: User's Manual (Nonlinear Dynamic Analysis of Solids in Three Dimensions). Revision 2*'', University of California, Lawrence Livermore National Laboratory, Report UCID-19592, 1986.

[14] G.I. Taylor: ''*The Use of Flat-Ended Projectiles for Determining Dynamic Yield Stress*'', Part I, Proceedings of the Royal Society, London, Ser. A, 194, pp. 289-299, 1948.

[15] J.P. Halleux, F. Casadei: "*Transient large-strain finite element analysis of solids*", in Computational Methods for Non-Linear Problems, Eds. C. Taylor, D.R.J. Owen and E. Hinton, Pineridge, Swansea, 1987.

[16] A. Huerta, F. Casadei: "*New ALE Applications in Non-linear Fast-transient Solid Dynamics*", Engineering Computations, Vol. 11, pp. 317-345, 1994.

Table 1 - Non-partitioned explicit time integration scheme for a Lagrangian description

0. Set initial conditions: $n \leftarrow 0$, $t \leftarrow t^0$, $\underline{x} \leftarrow \underline{x}^0$, $\underline{\sigma} \leftarrow \underline{\sigma}^0$, $\underline{v} \leftarrow \underline{v}^0$,
$W^{\text{int}} \leftarrow 0$, $W^{\text{ext}} \leftarrow W^{\text{kin}}$; ($n$ is the step counter);

1. `GO TO 4`;

2. Update velocities to half-step value: $\underline{v} \leftarrow \underline{v} + \frac{\Delta t^{\text{new}}}{2}\underline{a}$ (loop over nodes);
Set: $\boxed{n \leftarrow n+1}$, $\Delta t \leftarrow \Delta t^{\text{new}}$ and $t \leftarrow t + \Delta t$;

3. Update displacements and configuration:
$\underline{d} \leftarrow \underline{d} + \Delta t \underline{v}$ (loop over nodes),
$\underline{x} \leftarrow \underline{x} + \Delta t \underline{v}$ (loop over nodes);

4. Compute internal forces: $\underline{F}^{\text{int}} \leftarrow \sum_e \int_{V^e} \underline{B}^T \underline{\sigma} dV$ (loop over elements);
V^e and $\underline{B}^T$ are evaluated on the current configuration;
If $n > 0$, then $\underline{\sigma} \leftarrow \underline{\sigma} + \Delta\underline{\sigma}$ (constitutive law);
While computing element internal forces, evaluate Δt^{new}, the time increment for the next step, and add the internal energy increment to W^{int};

5. Assemble internal forces; evaluate and assemble external forces $\underline{F}^{\text{ext}}$;
these include both:
5.1 applied loads;
5.2 reaction forces due to essential boundary conditions (see Part II);
Add external work increment to W^{ext};

6. Compute accelerations and update velocities to full-step value (for post-processing only):
$\underline{a} \leftarrow \underline{M}^{-1}(\underline{F}^{\text{ext}} - \underline{F}^{\text{int}})$ (loop over nodes);
If $n > 0$, then $\underline{v} \leftarrow \underline{v} + \frac{\Delta t}{2}\underline{a}$ (loop over nodes);

7. If no output is required at this time, then `GO TO 2`;

8. Compute kinetic energy and check energy balance: $W^{\text{ext}} \approx W^{\text{int}} + W^{\text{kin}}$;
Print requested output;

9. If final time not reached, then `GO TO 2`.

Table 2 - List of quantities used in the time integration algorithm with partitioning

Symbol	Description	Symbol	Description
$\underline{a}_k$	Nodal accelerations	$\bar{\psi}_k$	Neighbouring node frequency
C_s	Stability safety factor	T, t	Time
d	Depth of the partition	Δt	Time increment (for the current element)
$\underline{F}_k^{\text{int}}$	Internal forces (assembled)	Δt_e	Time increment dictated by all elements
$\underline{F}_k^{\text{ext}}$	External forces (assembled)	Δt_i	Elemental time increment
φ_i	Intrinsic element frequency	$\Delta \bar{t}_i$	Elemental time increment for configurations
$\bar{\varphi}_i$	Neighbouring element frequency	Δt_s	Stability time step of current element
Φ_l	Level frequency associated with level l	δt_k	Nodal time increment
Φ_d	Maximum level frequency	Δt_{min}	Minimum time step in the partition
i	Index of current element	Δt_{max}	Maximum time step in the partition
I	Cycle counter within a macro step	ΔT	Macro time step
k	Index of current node	ΔT_d	Time increment of deepest level
κ	Highest power of 2 contained in I	ΔT_l	Level's time increment
l	Index of current level	$\Delta \tau$	Time increment (for the current element)
m	Minimum active level frequency	$\Delta \tau_{\text{min}}$	Minimum time step (in element update)
$\underline{m}_k$	Nodal masses (assembled)	$\Delta \tau_{\text{max}}$	Maximum time step (in element update)
M	Maximum level frequency	$\underline{u}_k$	Total displacements
M_s	Total number of cycles	$\delta \underline{u}_k$	Lagrangian displacement increments
μ	Flag for partition update	$\delta \underline{u}_k'$	Relative displacement increments (ALE)
N_e	Total number of elements	$\underline{v}_k$	Nodal velocities
N_n	Total number of nodes	$\underline{w}_k$	Nodal grid velocities (ALE)
N_p	Macro step counter	$\underline{x}_k$	Current configuration
ψ_k	Intrinsic node frequency	$\delta \underline{x}_k$	Displacement increments

Solution	Mesh	Time step	Steps (N_p)	Cycles (M_s)	Max level frequency (Φ_d)	Elements × cycles	CPU (s)
1 (bipcp2)	Uniform, 100 elements	Uniform, $C_s = 0.8$	250	—	—	25100	0.22
2 (bipcp3)	Uniform, 800 elements	Uniform, $C_s = 0.8$	2000	—	—	1600800	3.37
3 (bipcp0)	Gradual, 111 elements	Uniform, $C_s = 0.8$	2000	—	—	222111	1.80
4 (bipcp1)	Gradual, 111 elements	Partition, $C_s = 0.8$	252	2016	8	47487	0.28

Table 3 - Numerical solutions for the first numerical example

Solution	Description	Steps (N_p)	Cycles (M_s)	Max level frequency (Φ_d)	Elements × cycles	CPU (s)	Speed-up (theor)/ actual
1 (bamd01)	Lagrangian, uniform step	59886	—	—	14971750	61.89	—/—
2 (bamd03)	Lagrangian, spatial partitioning	1162	85537	128	1296171	6.75	11.6/9.2
3 (bamd05)	Lagrangian, domain decomposition	1152	—	—	—	6.05	—/10.2
4 (bamdal)	ALE, uniform step	1950	—	—	487750	4.34	30.7/14.3

Table 4 - Numerical solutions for the second numerical example

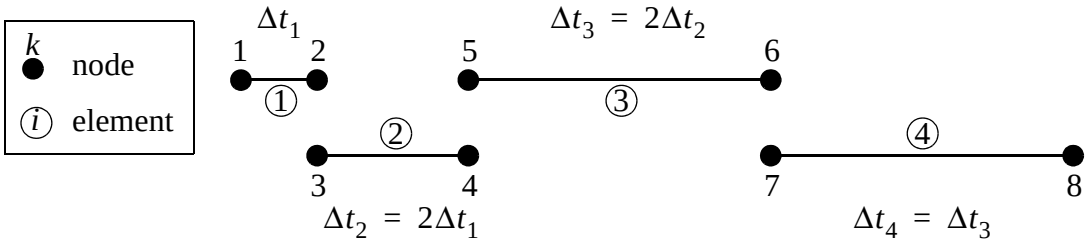
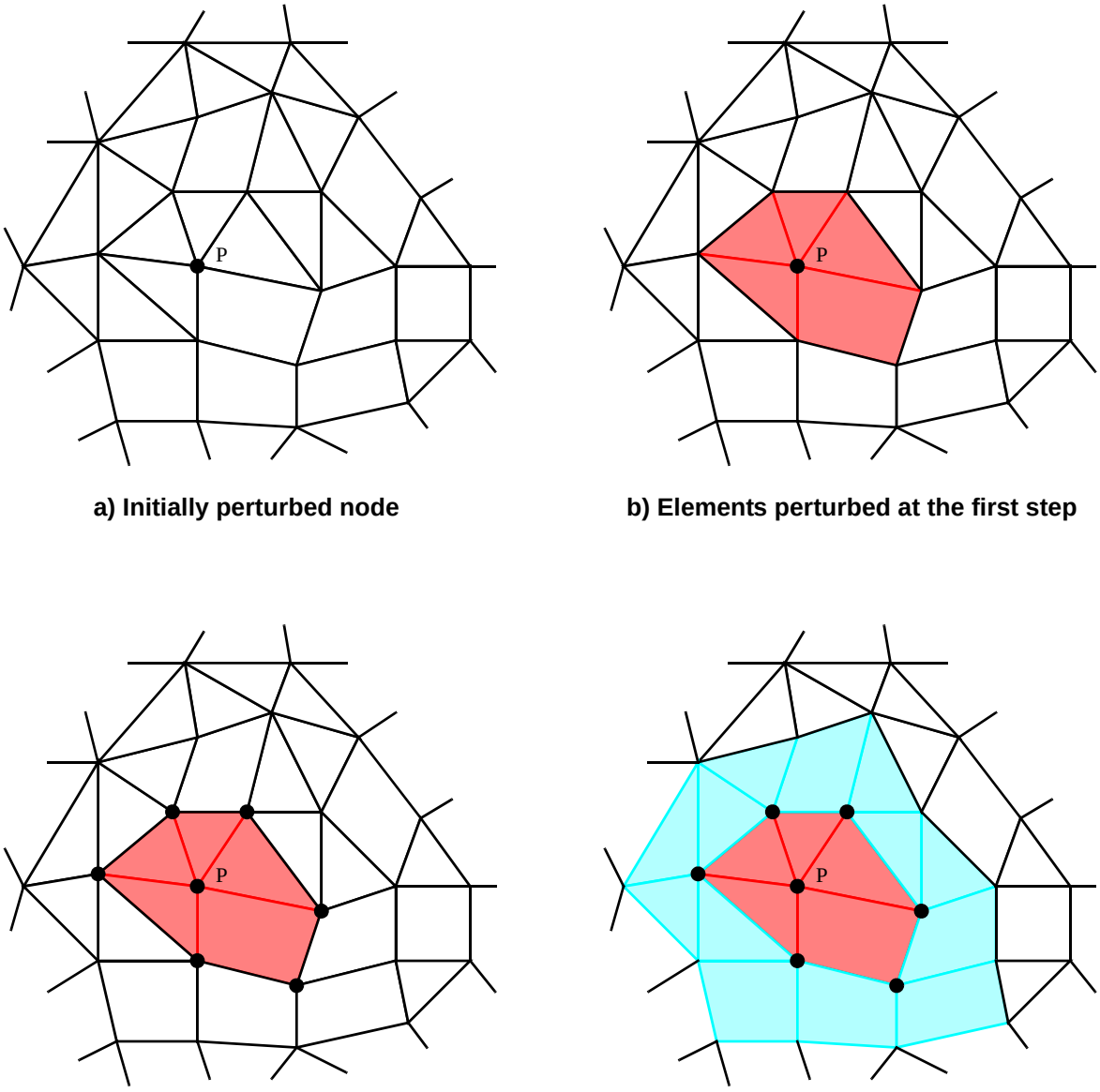


Figure 1 - A simple mesh made of unconnected elements



a) Initially perturbed node

b) Elements perturbed at the first step

c) Nodes perturbed at the second step

d) Elements perturbed at the second step

Figure 2 - Propagation of perturbations

.

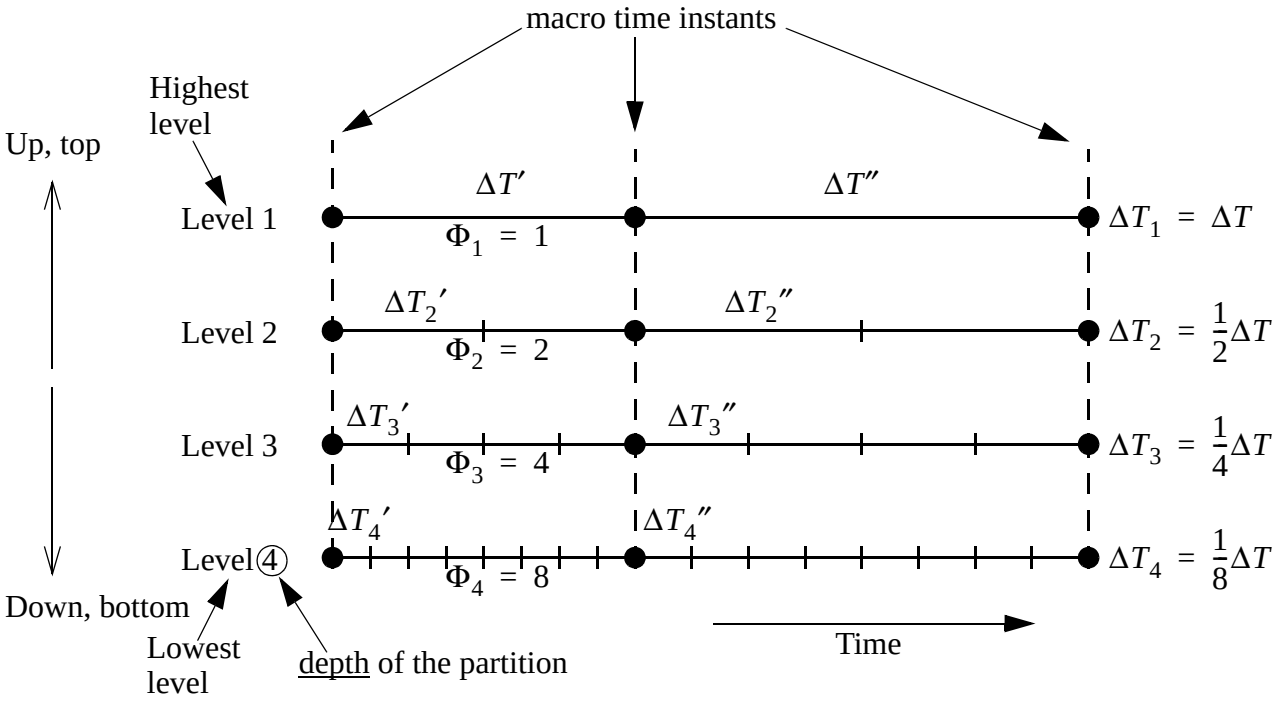


Figure 3 - Example of two consecutive macro steps in a binary partition with 4 levels

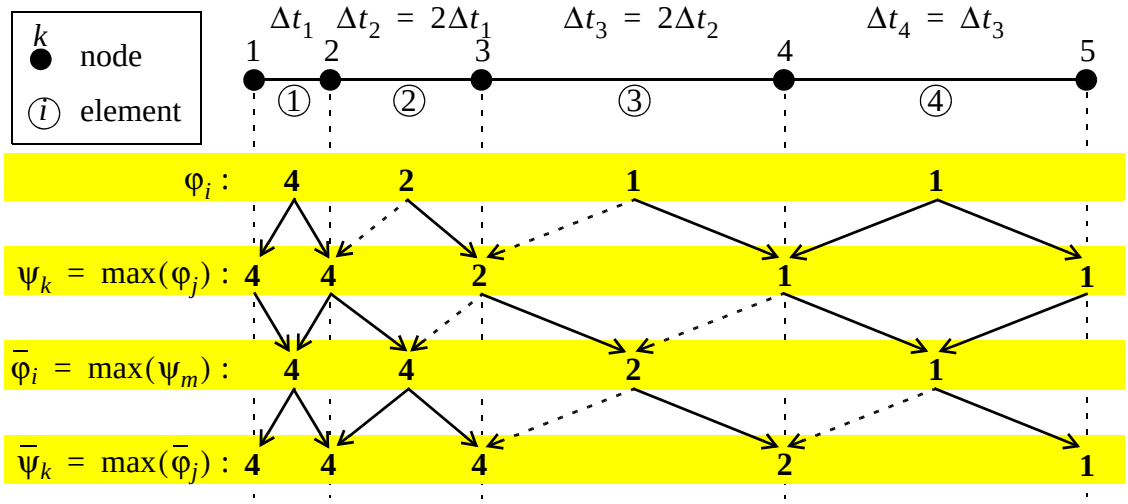


Figure 4 - A simple mesh of connected elements

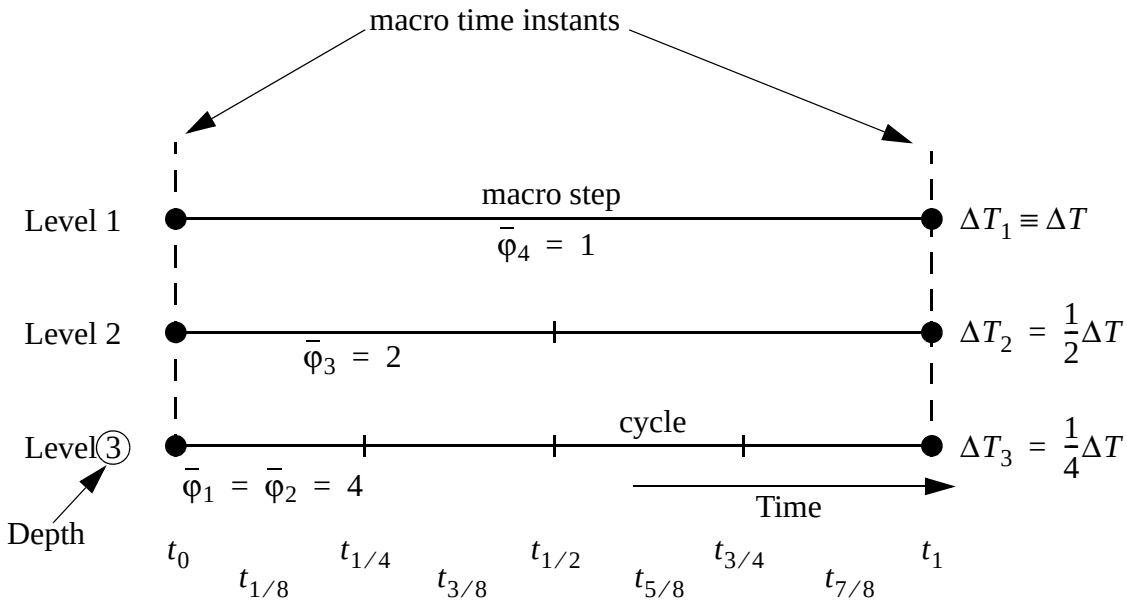
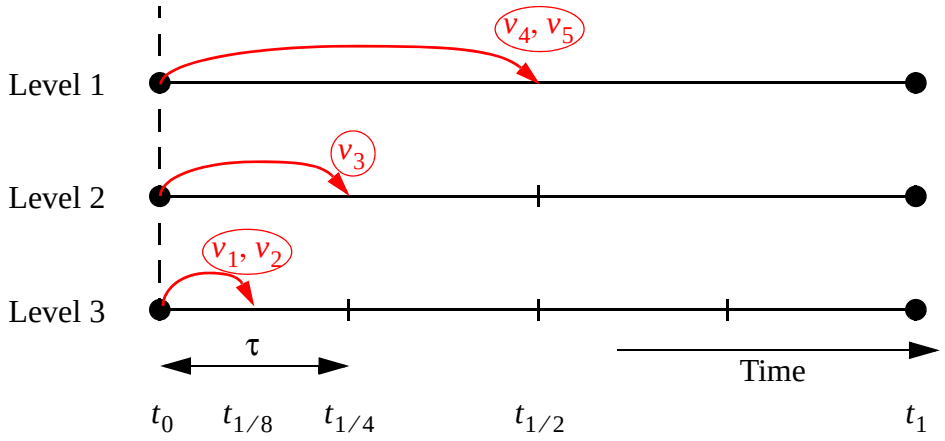
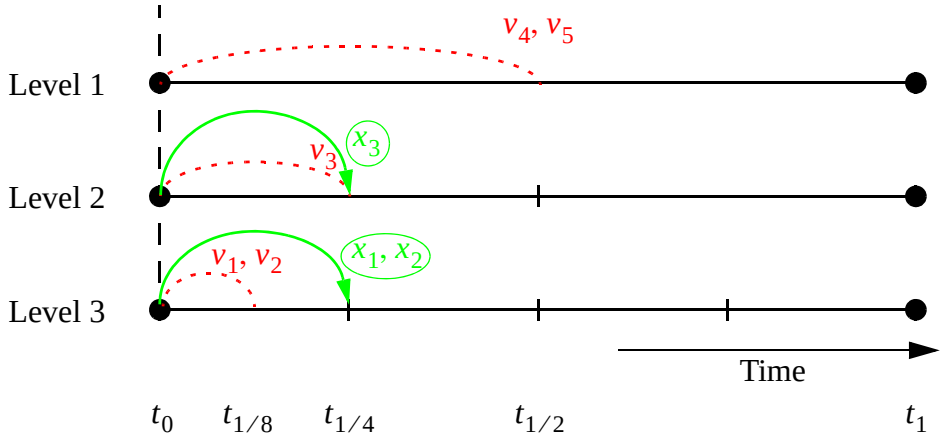


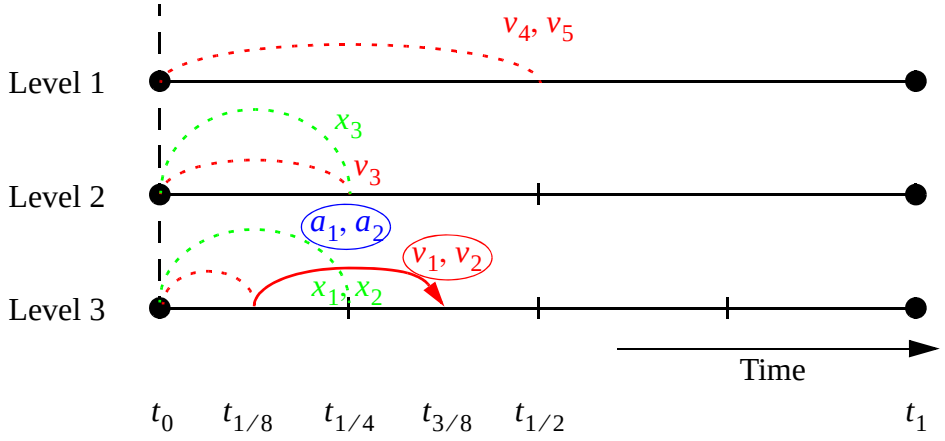
Figure 5 - Partition for the mesh shown in the previous Figure



a) First calculation of mid-step velocities in the macro step

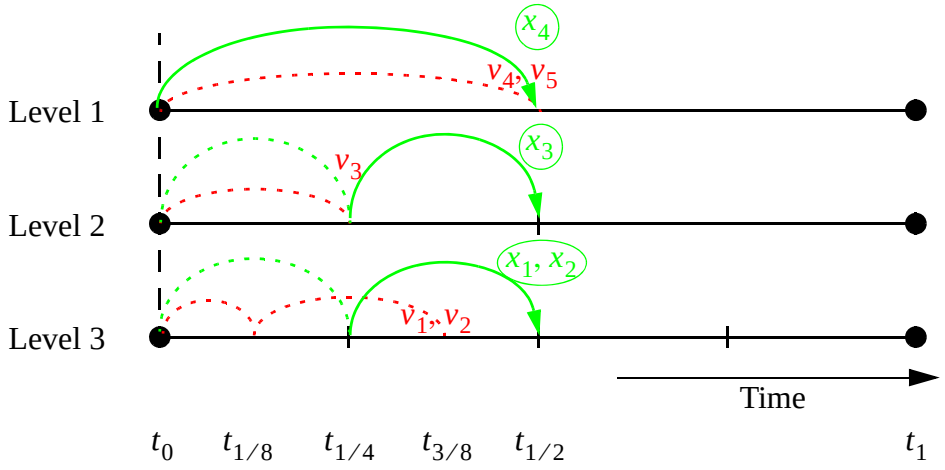


b) First update of configurations in the macro step

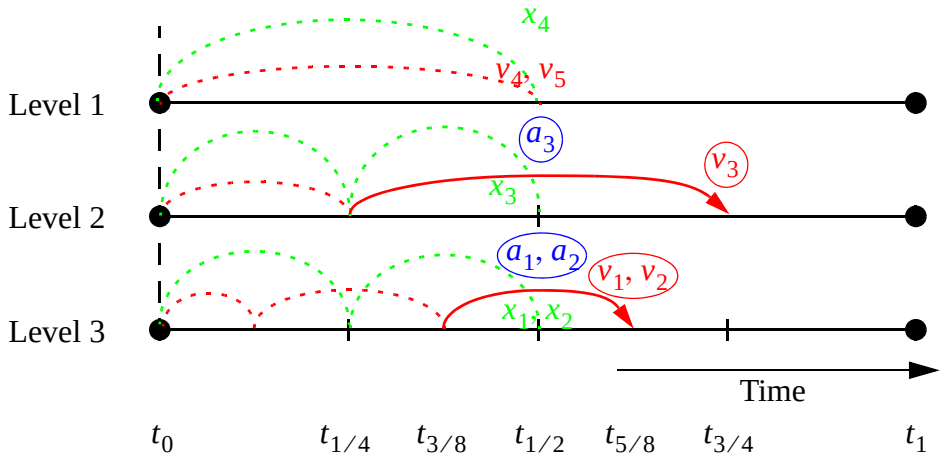


c) The situation at the end of the first cycle

Figure 6 - First cycle

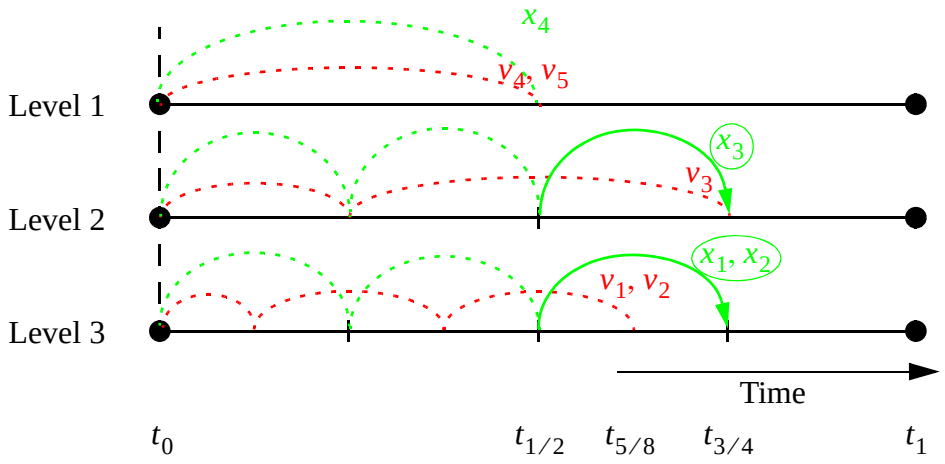


a) Second update of configurations in the macro step

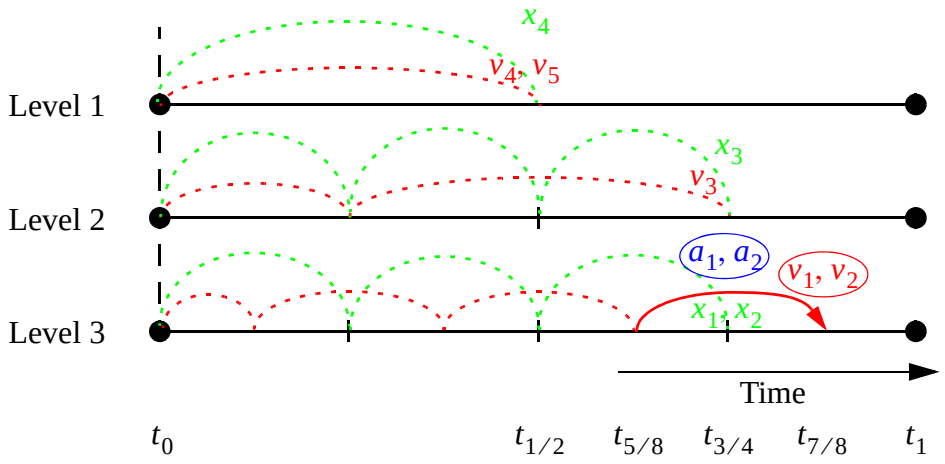


b) Situation at the end of the second cycle

Figure 7 - Second cycle

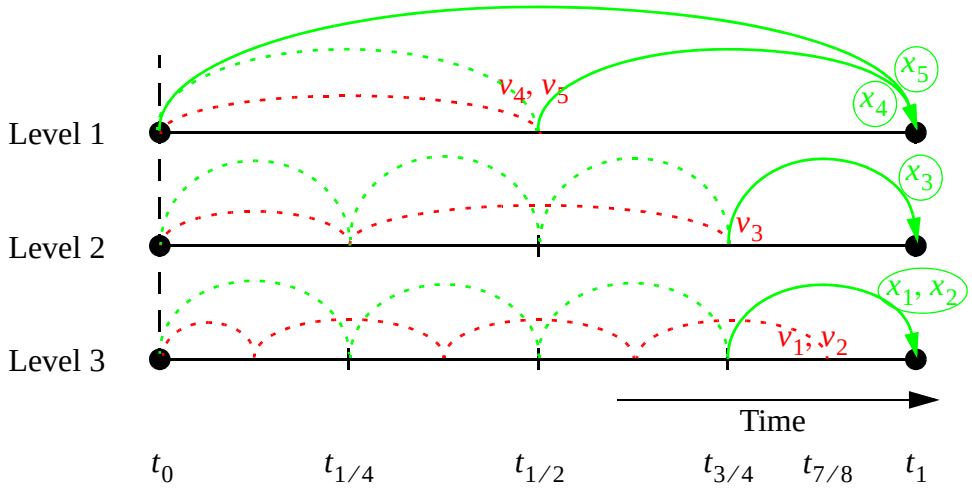


a) Third update of configurations in the macro step

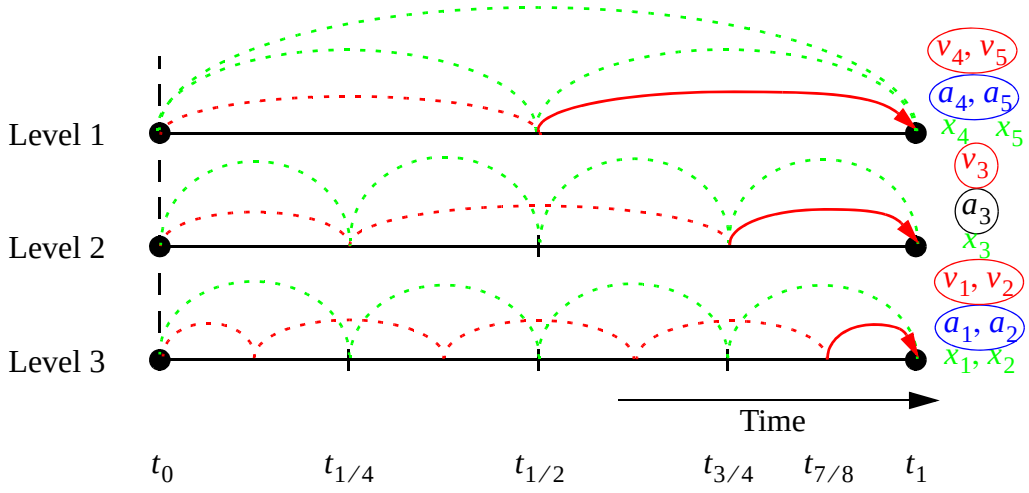


b) Situation at the end of the third cycle

Figure 8 - Third cycle



a) Fourth update of configurations in the macro step



b) Situation at the end of the fourth and last cycle (end of step)

Figure 9 - Fourth cycle

I	$\kappa(I)$	$m(I)$
1	1	16
2	2	8
3	1	16
4	4	4
5	1	16
6	2	8
7	1	16
8	8	2
9	1	16
10	2	8
11	1	16
12	4	4
13	1	16
14	2	8
15	1	16
16 = M	16	1

We define $m(I) = M/\kappa(I)$, where $\kappa(I)$ is the highest power of 2 'contained' as a factor in I. Consequently:

- For odd values of I, it is $m(I) = M$ (= 16), because in these cases the first (and only) level to be integrated is the deepest one.
- For even values of I, it is $m(I) < M$.

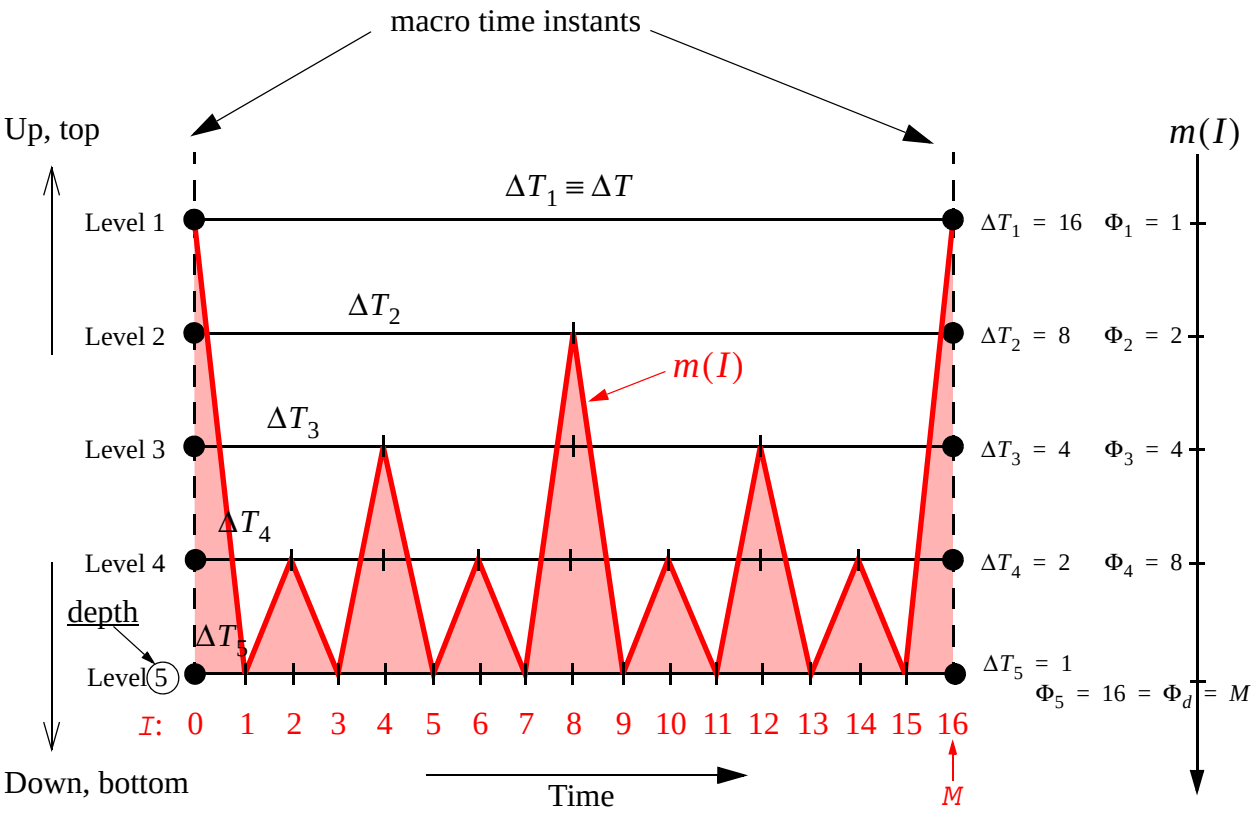


Figure 10 - Illustration of the activity function

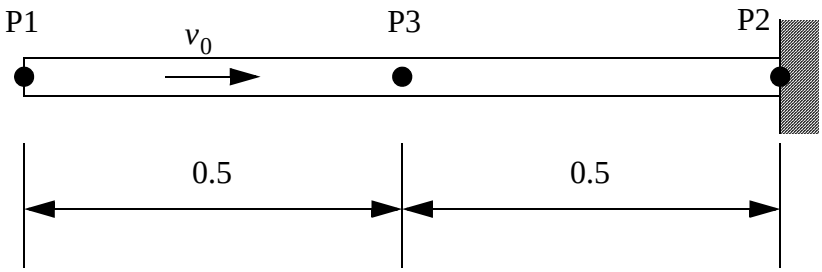
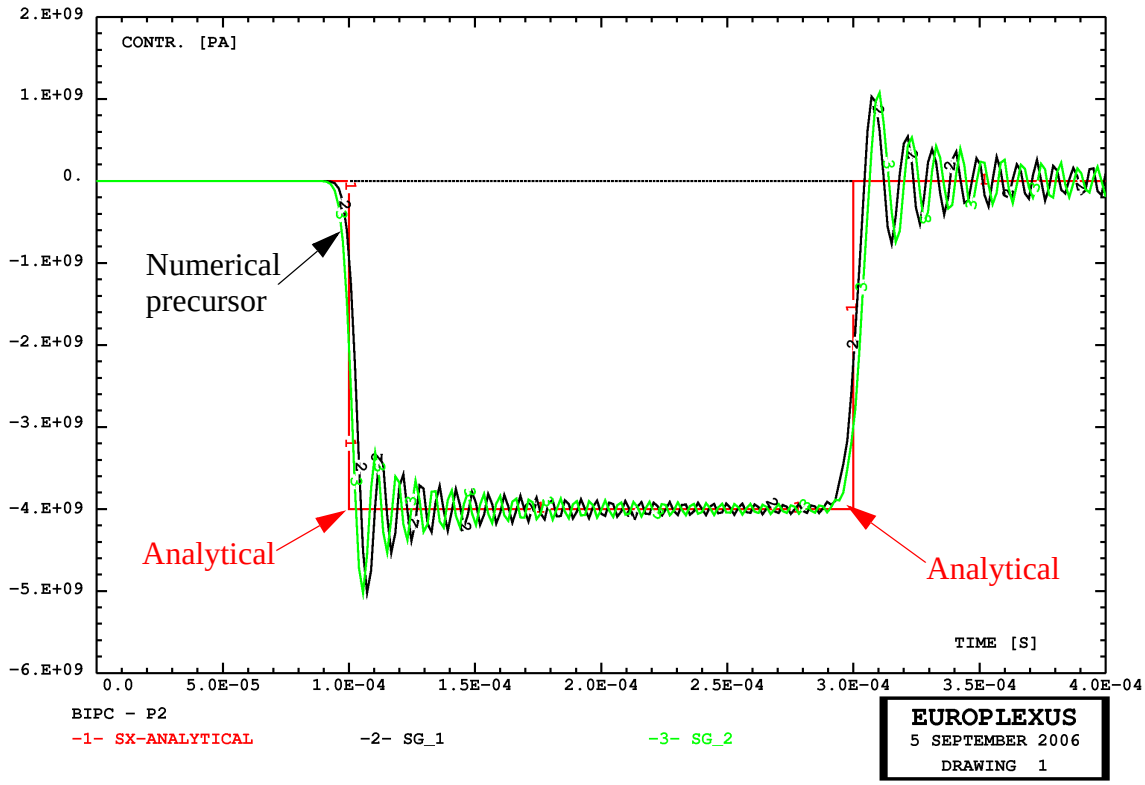
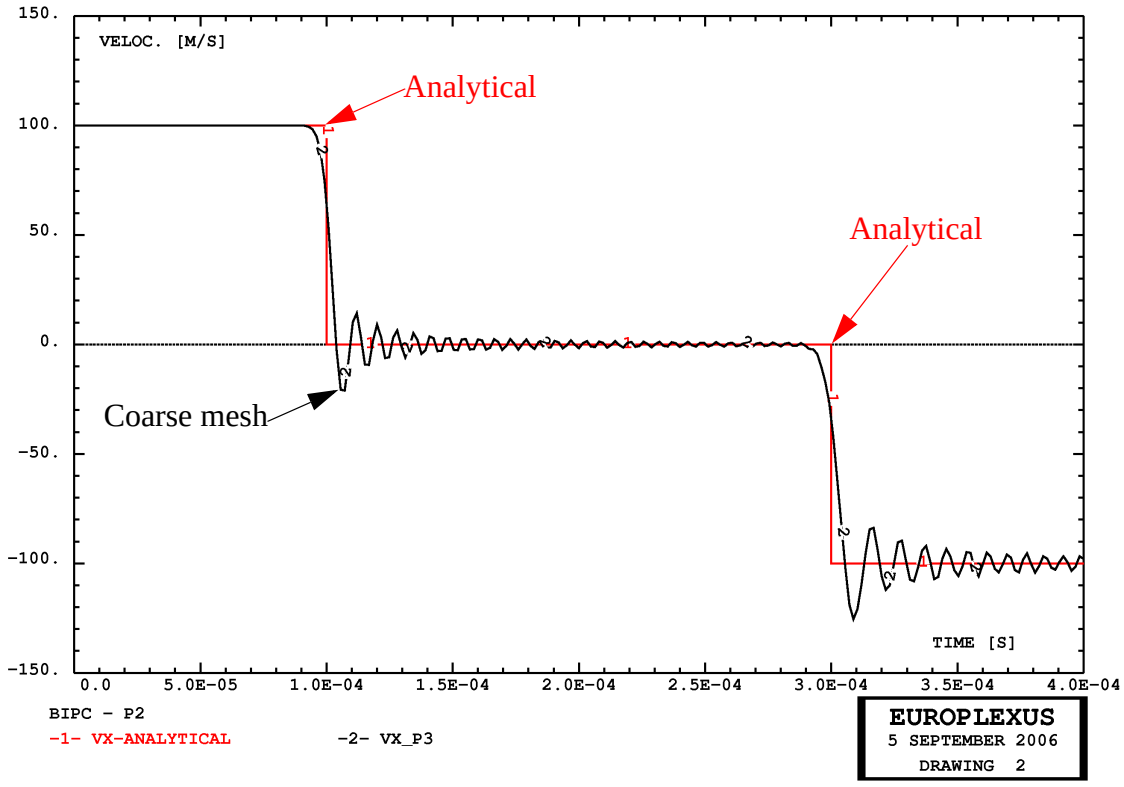


Figure 11 - First test problem: one-dimensional wave propagation in a bar

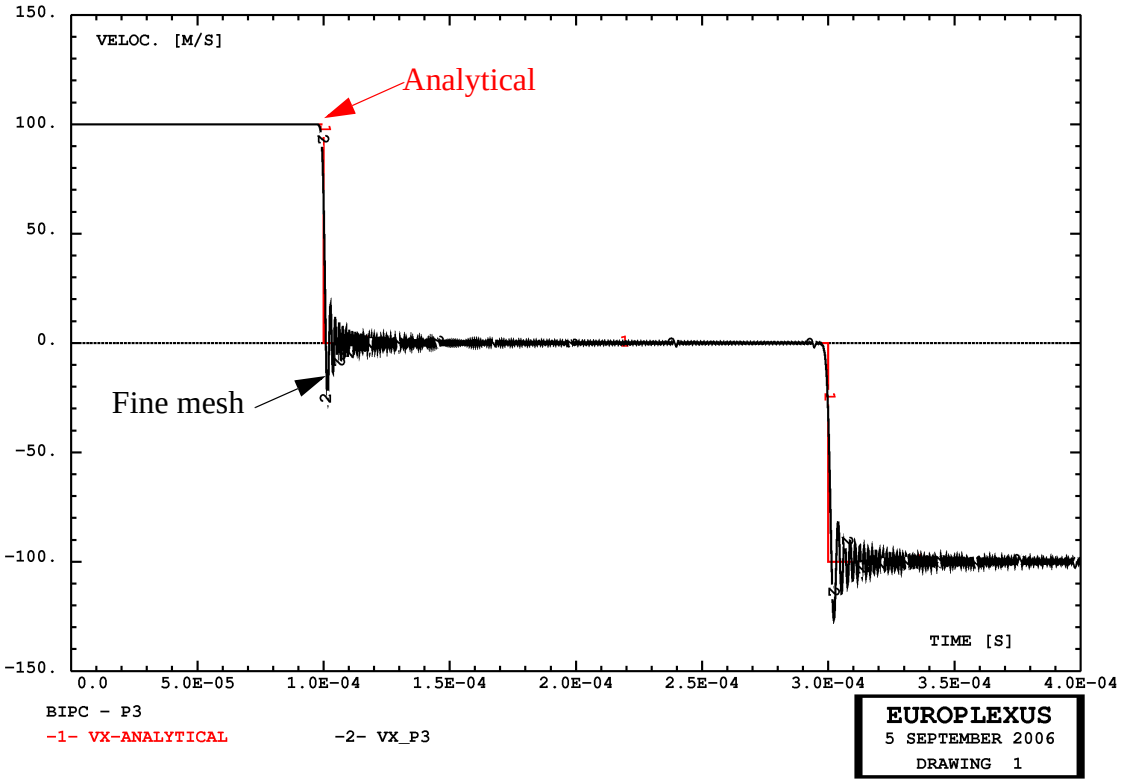


a) Stress

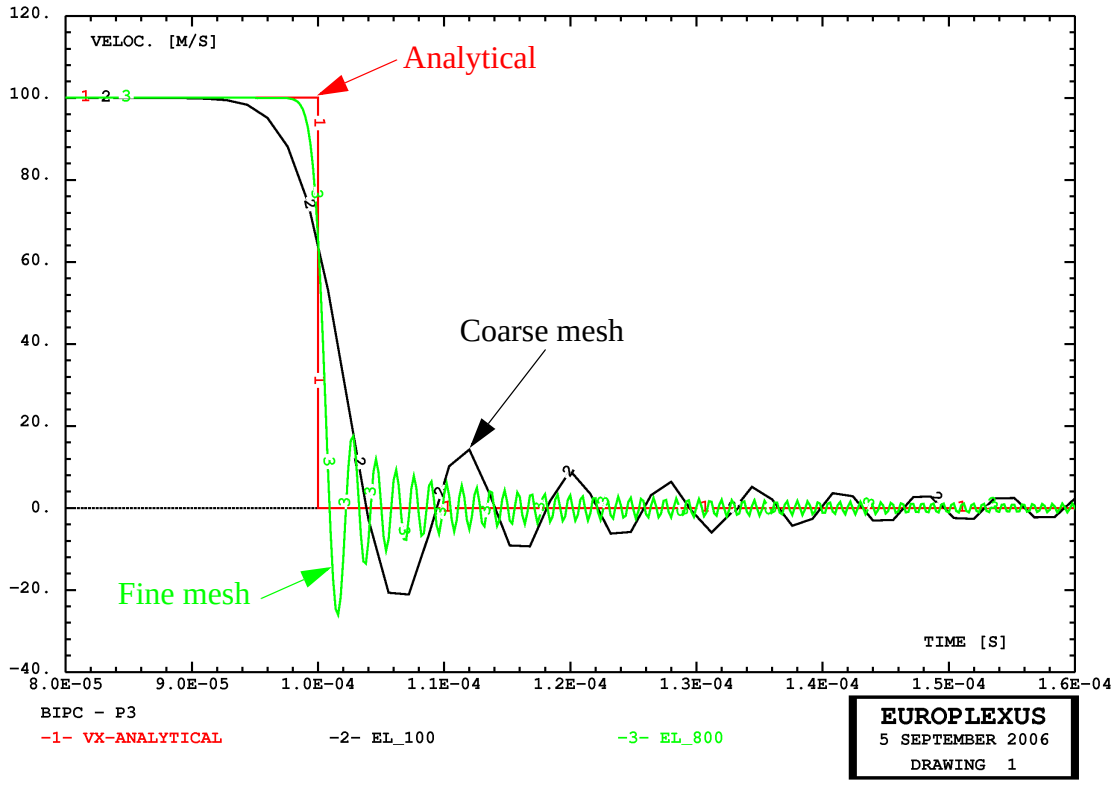


b) Velocity

Figure 12 - First test problem: solution with coarse uniform mesh, no partitioning

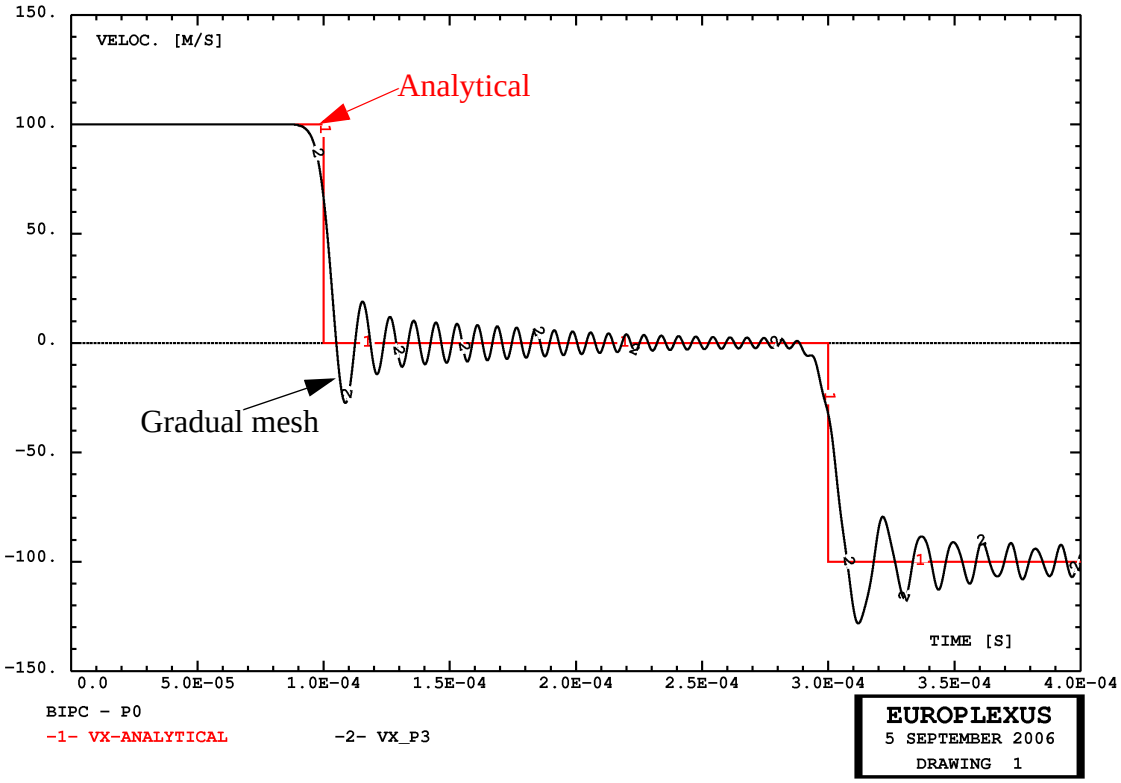


a) Velocity

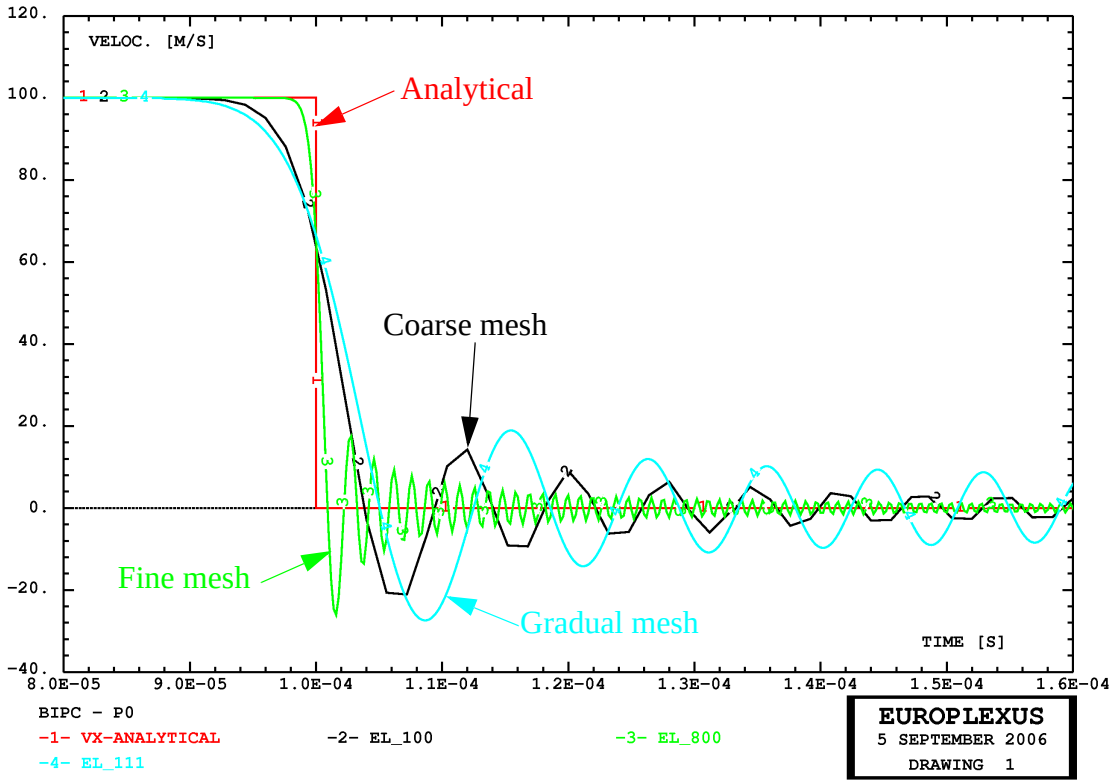


b) Comparison of velocity in coarse and fine mesh cases (zoom)

Figure 13 - First test problem: solution with fine uniform mesh, no partitioning

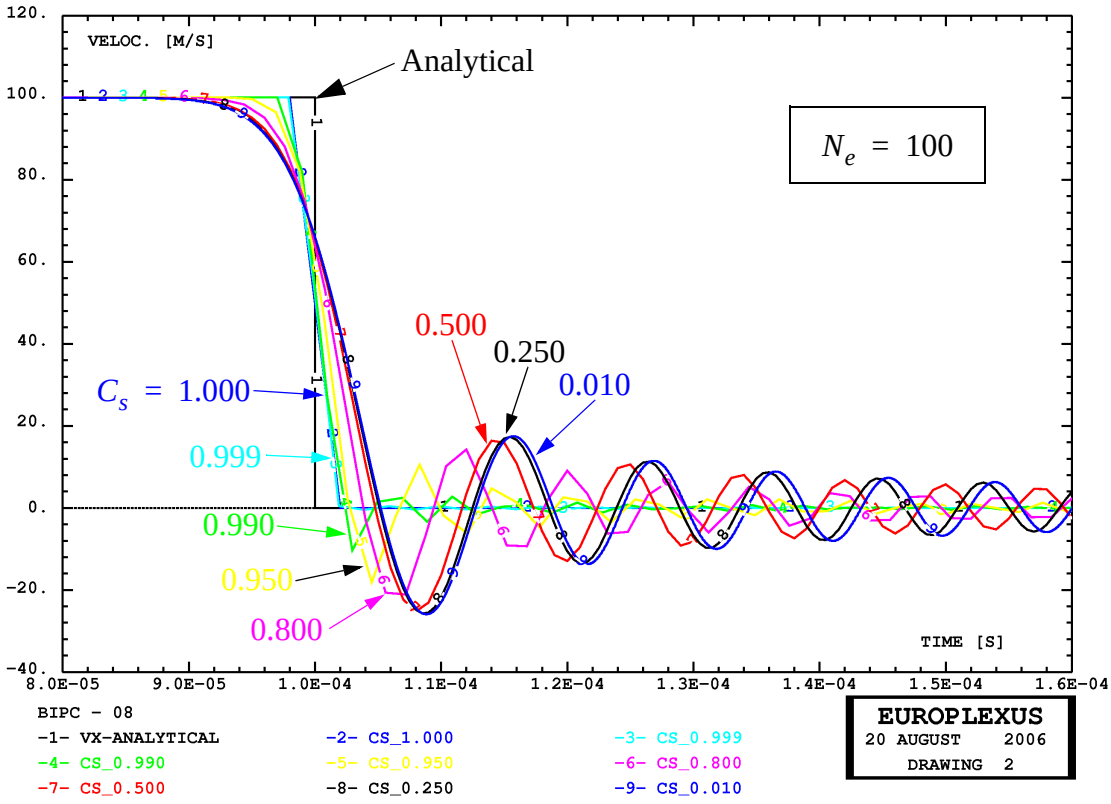


a) Velocity

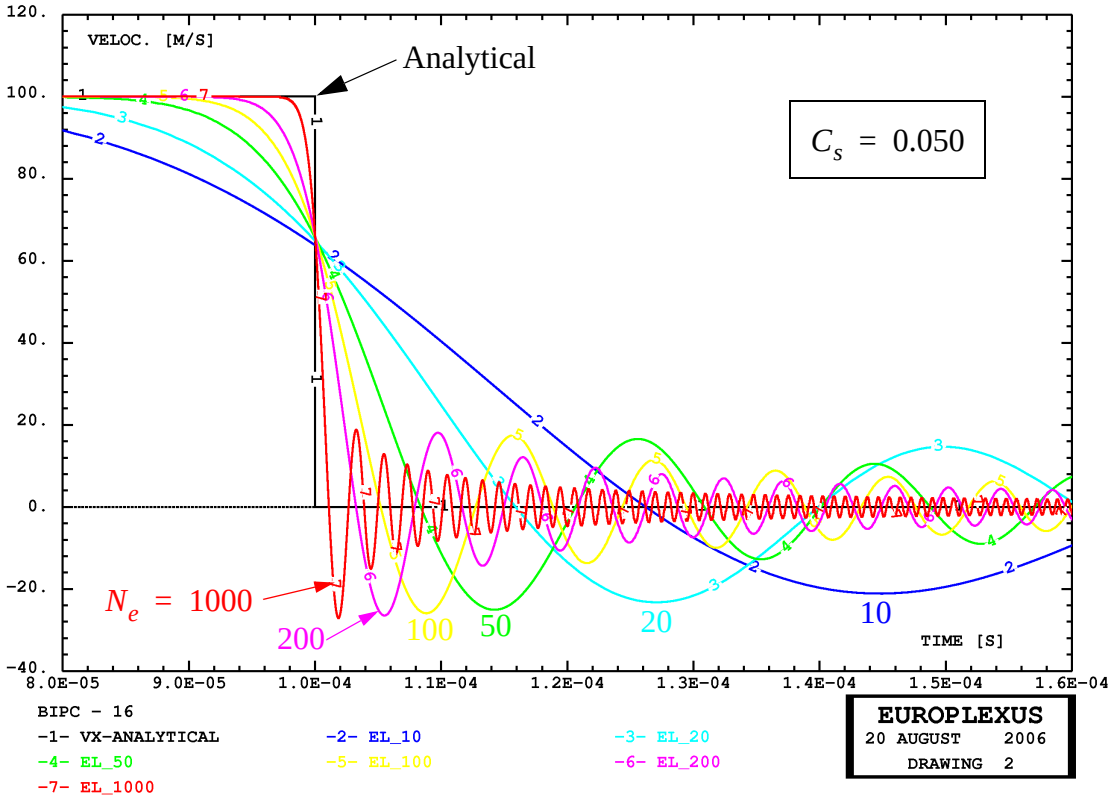


b) Comparison of velocity in coarse, fine and gradual mesh cases (zoom)

Figure 14 - First test problem: solution with gradual mesh, no partitioning

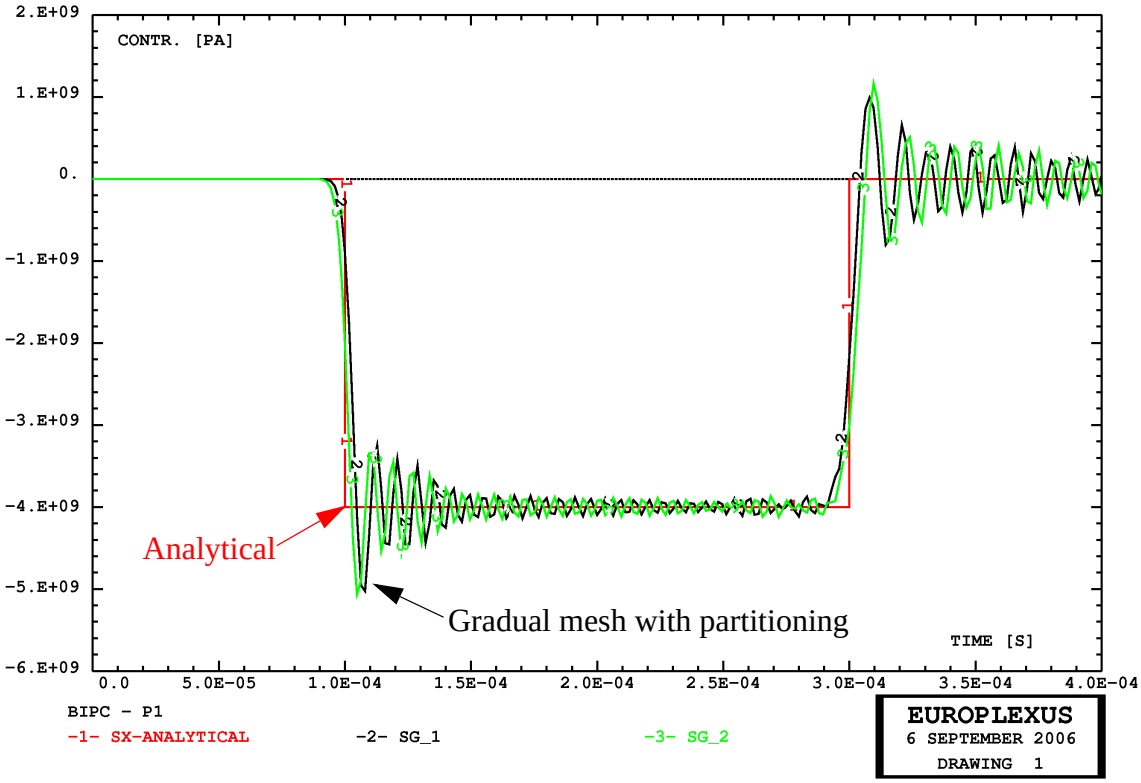


a) Velocity: influence of stability safety factor

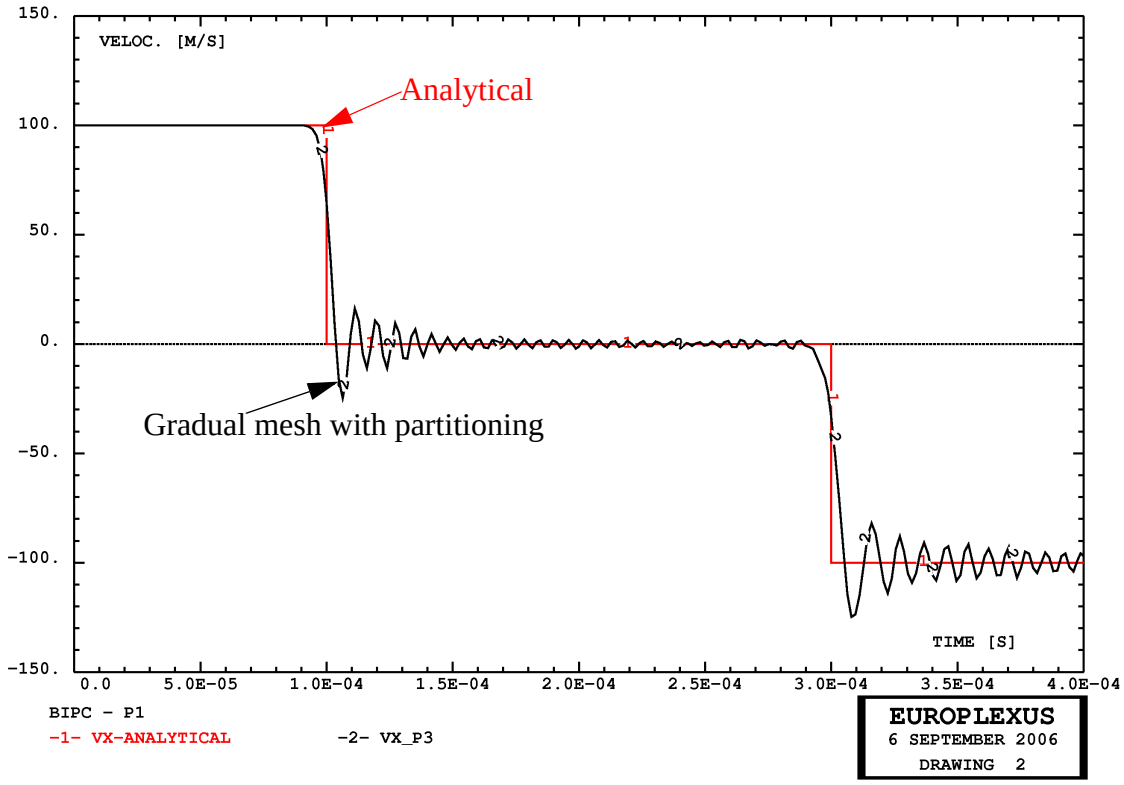


b) Velocity: influence of mesh size (number of elements)

Figure 15 - First test problem: further solutions without partitioning

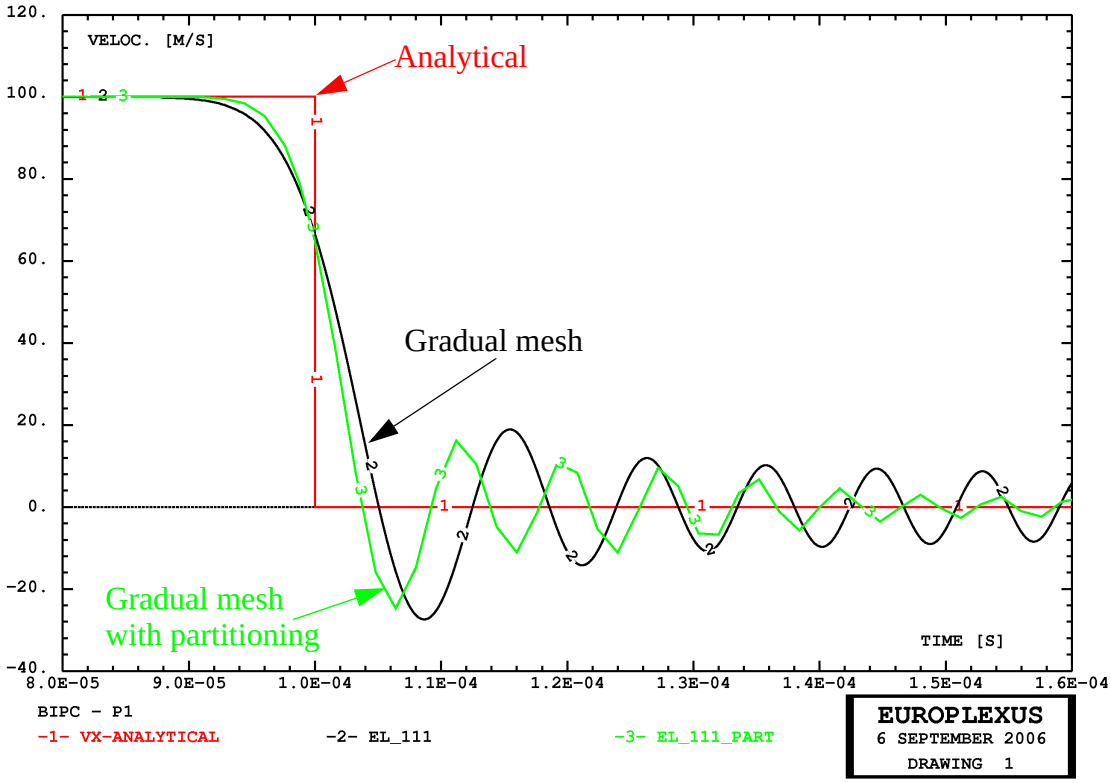


a) Stress

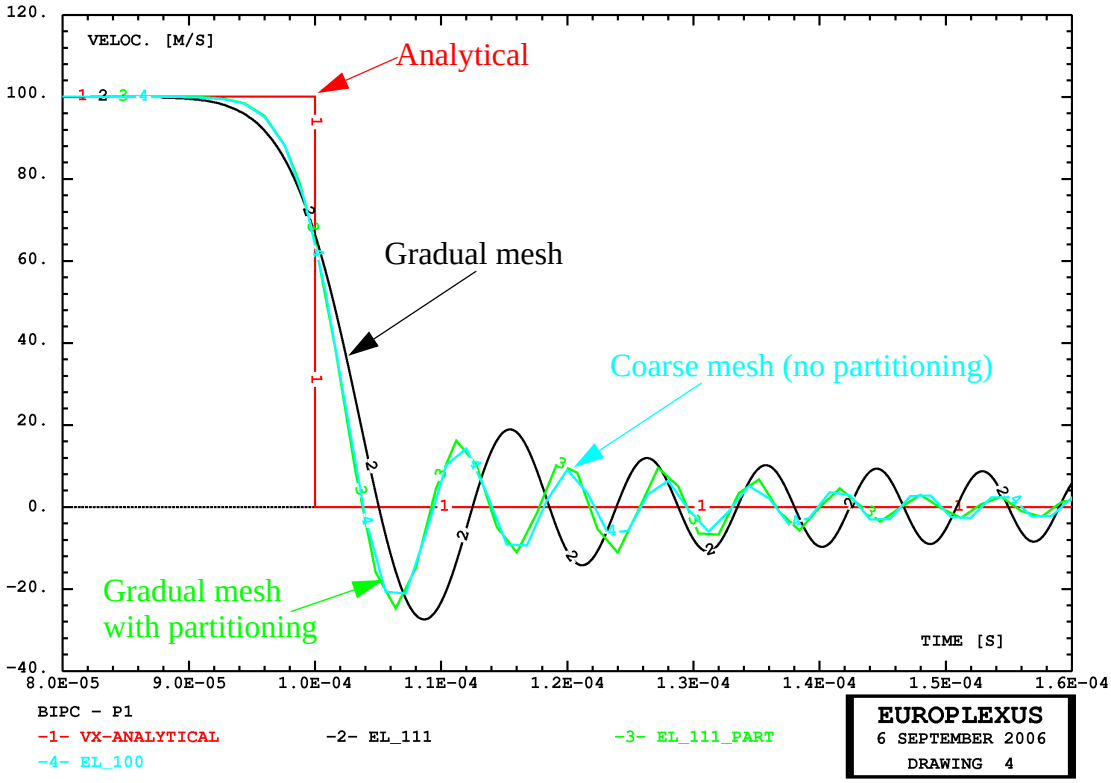


b) Velocity

Figure 16 - First test problem: solution with gradual mesh and spatial partitioning



a) Comparison of velocity in gradual mesh case with and without space partitioning (zoom)



b) Comparison including also case with uniform coarse mesh

Figure 17 - First test problem: solution with gradual mesh and spatial partitioning

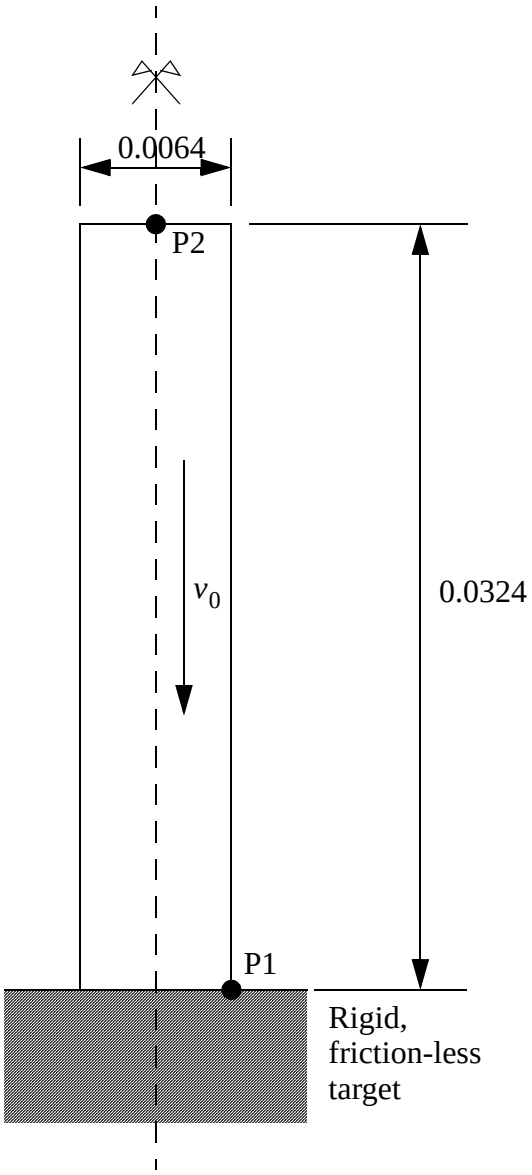
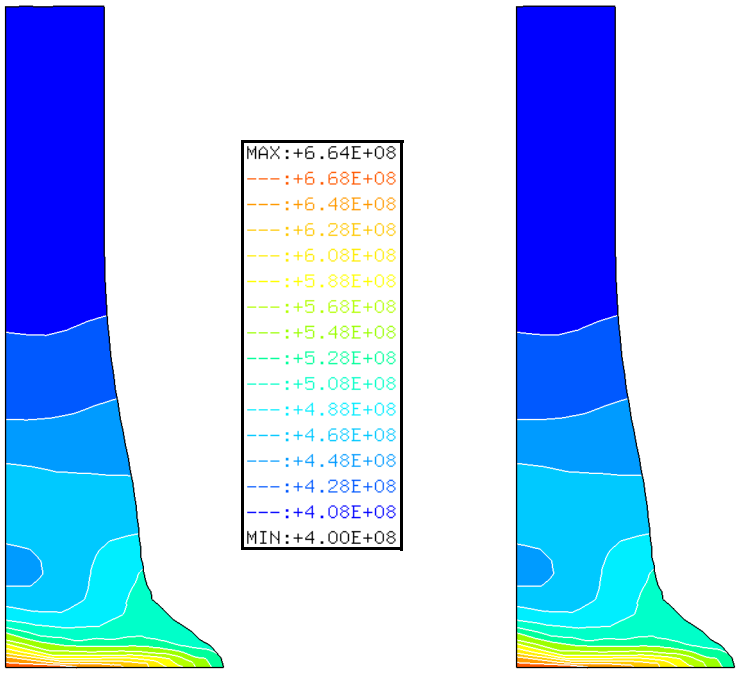
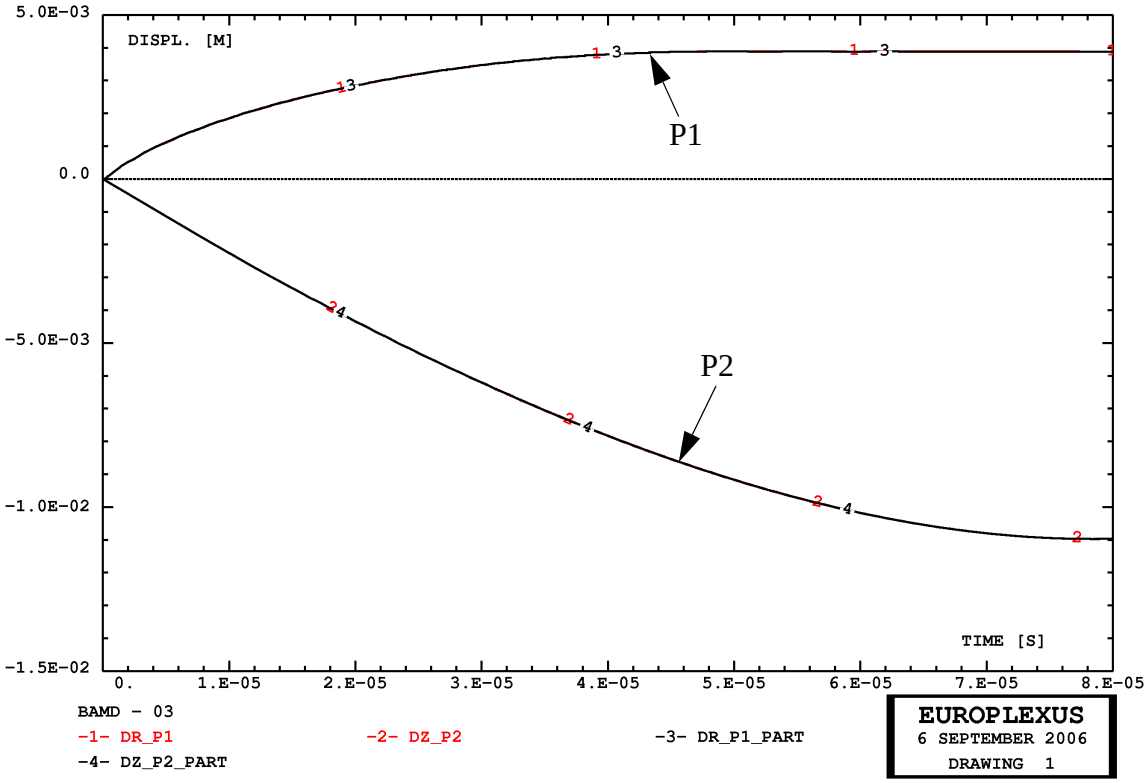


Figure 18 - Second test problem: Taylor bar impact

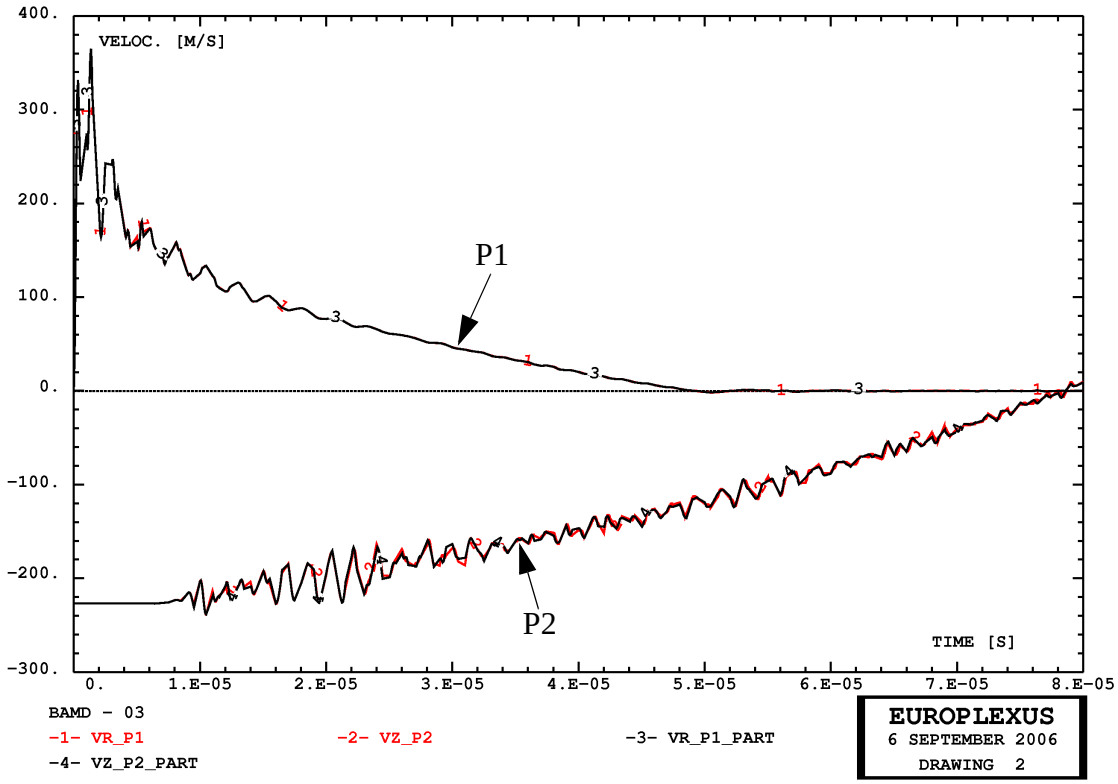


Uniform time increment **Spatial partitioning**

Figure 19 - Second test problem: final yield stress without and with spatial partitioning

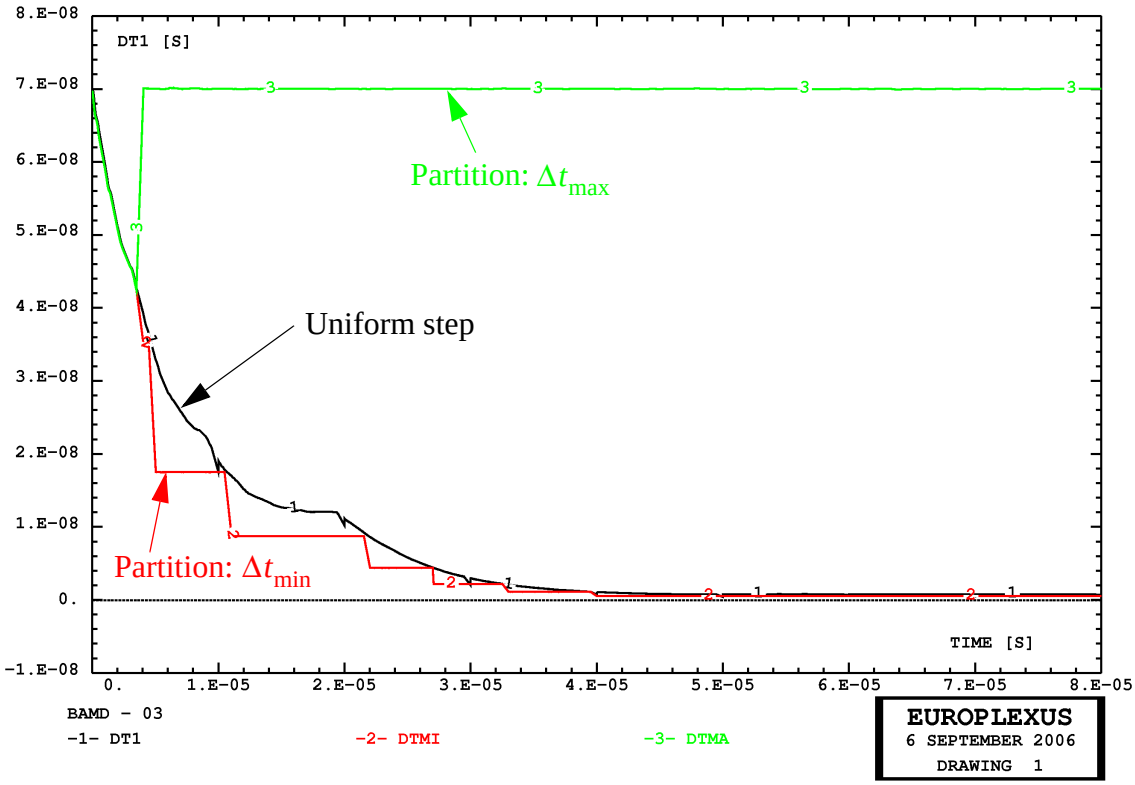


a) Comparison of displacements at points P1 and P2 without and with spatial partitioning

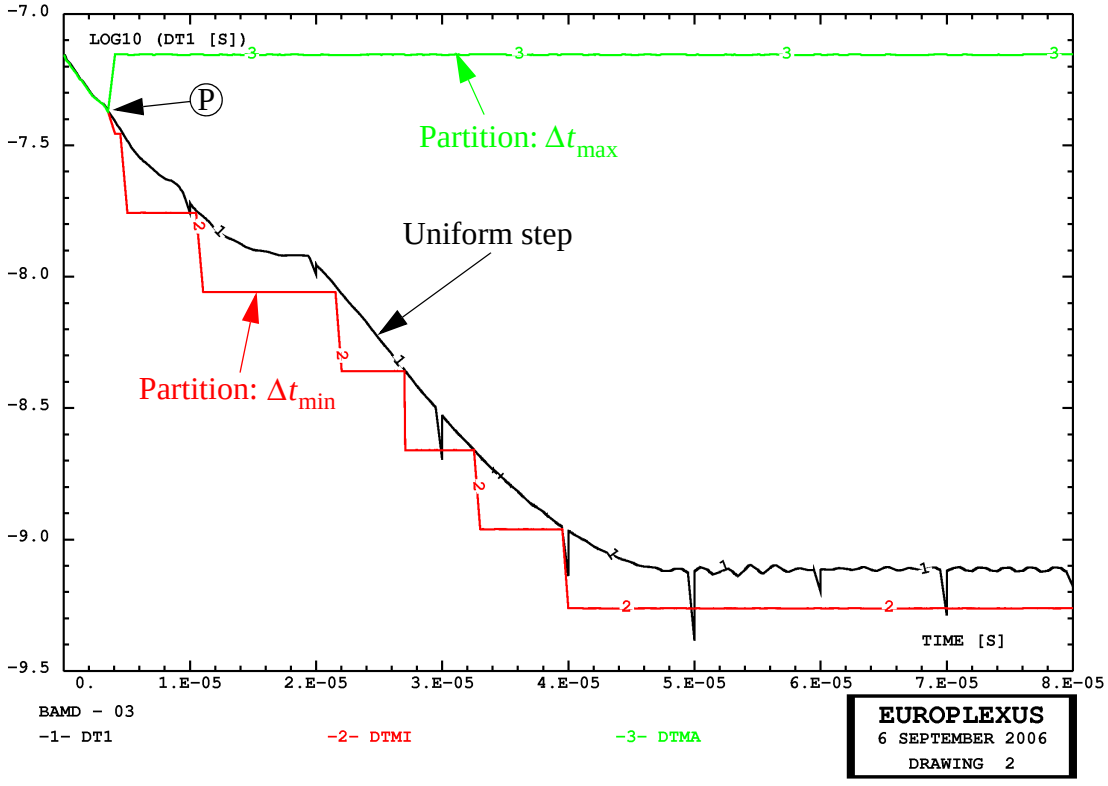


b) Comparison of velocities at points P1 and P2 without and with spatial partitioning

Figure 20 - Second test problem: relevant displacements and velocities



a) Comparison of time increments without and with spatial partitioning



b) Comparison of time increments without and with spatial partitioning (logarithmic scale)

Figure 21 - Second test problem: time increments

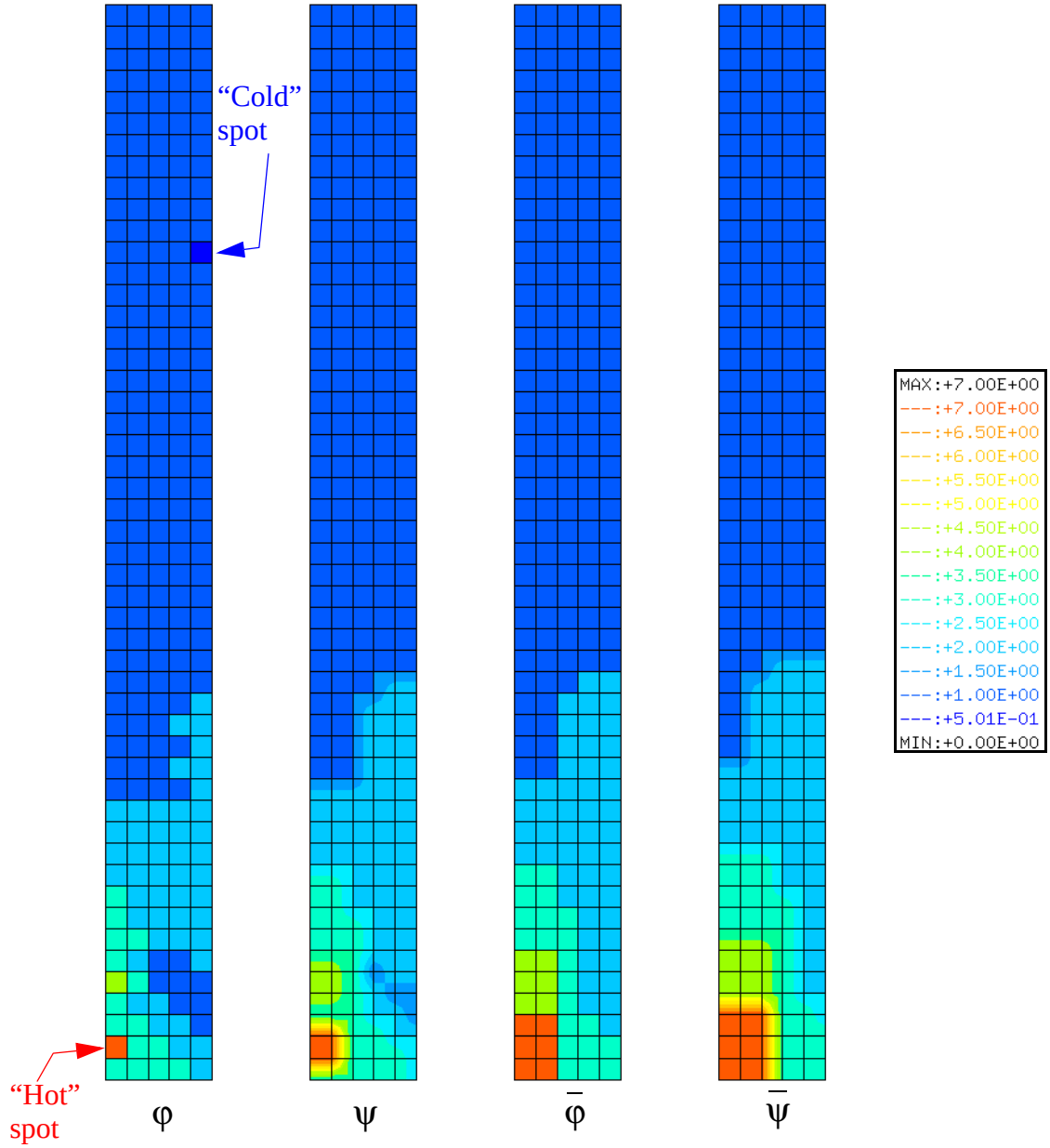


Figure 22 - Second test problem: logarithm in base 2 of partition frequencies at the final time

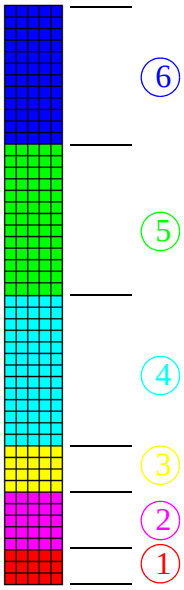
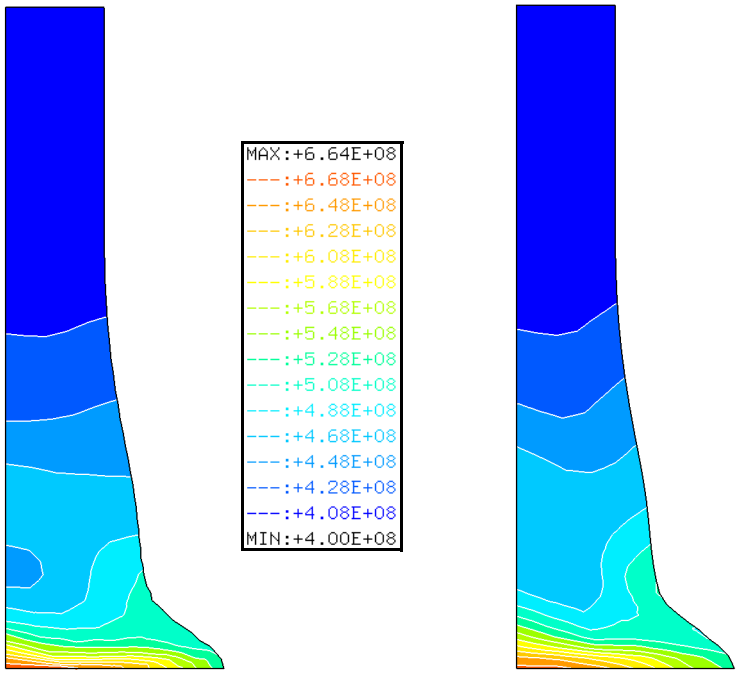
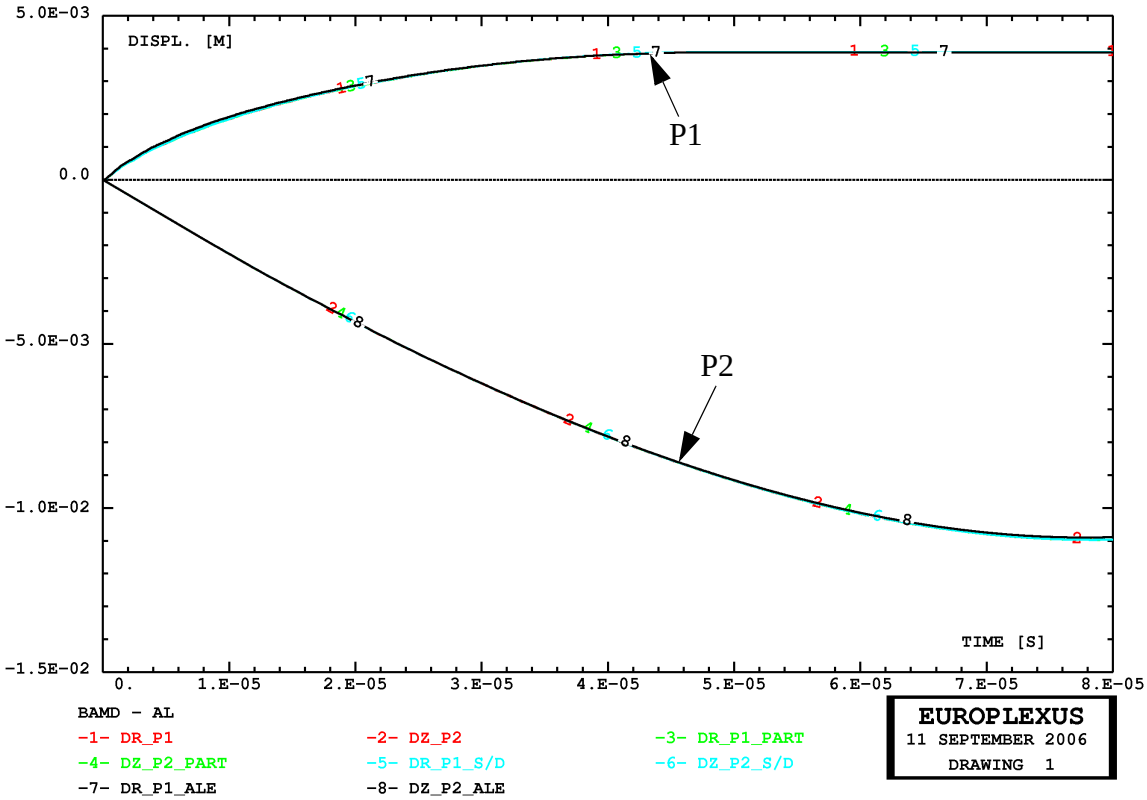


Figure 23 - Domain decomposition used for the Taylor bar impact test

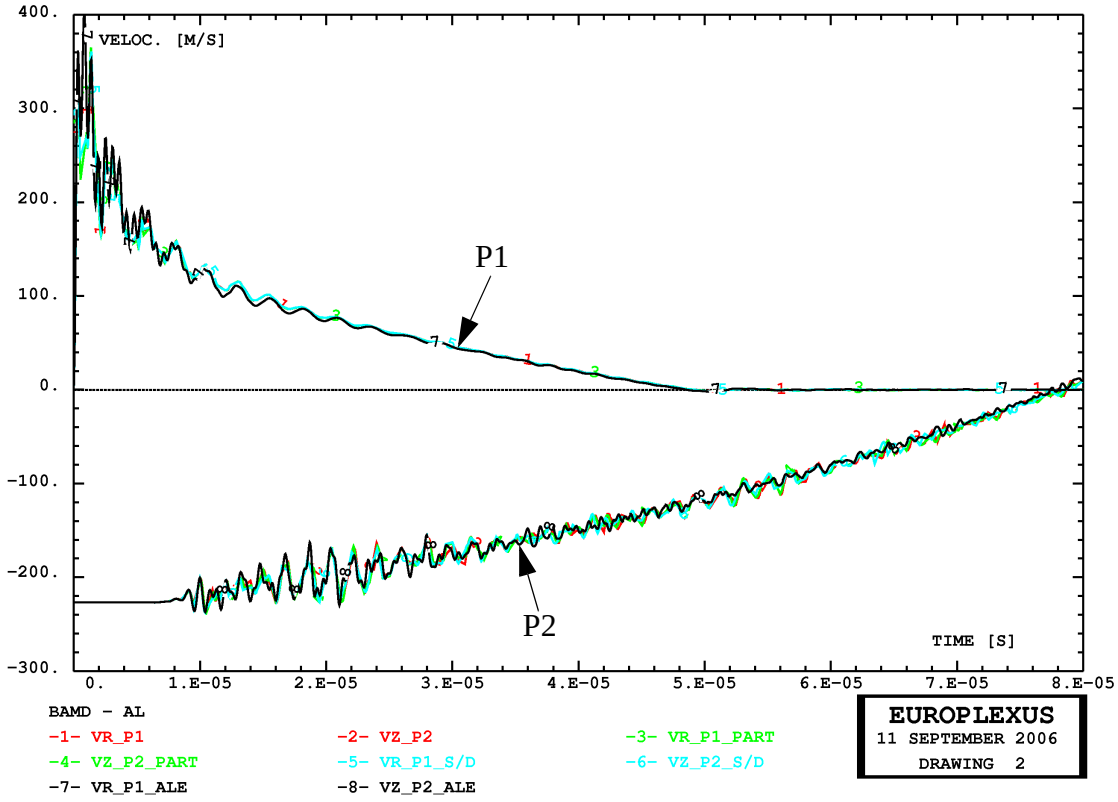


Domain decomposition **ALE**

Figure 24 - Second test problem: final yield stress with domain decomposition and with ALE



a) Comparison of displacements at points P1 and P2 (all solutions)



b) Comparison of velocities at points P1 and P2 (all solutions)

Figure 25 - Second test problem: relevant displacements and velocities (all solutions)

Spatial Time Step Partitioning in Explicit Fast Transient Dynamics — Part II - Treatment of Boundary Conditions and Extension to ALE

1. Introduction

Part I of the present work (see the first part of this report) has presented the basic aspects of an algorithm to achieve spatial partitioning of the time increment in the explicit central-difference time integration scheme commonly used for finite-element modeling of fast transient dynamic phenomena. Simple numerical examples have shown that the proposed technique has the potential for obtaining significative reductions of CPU time in transient explicit analyses, without at the same time losing any of the good properties—accuracy, robustness etc.—of the classical uniform-step time integration algorithm.

However, the sample partitioned calculations considered in Part I were limited to purely structural applications with a Lagrangian description and required only the simplest type of boundary conditions—node blockages—which were imposed via a direct, uncoupled method.

Of course, in order to actually obtain significant benefits in realistically complex applications, the core spatial partitioning technique of Part I is not sufficient. The method must be “industrialized” and several technical aspects of great practical importance must be dealt with. Work in this sense is ongoing within the EUROPLEXUS code, a general finite-element computer program for the fast transient analysis of fluid-structure systems subjected to transient dynamic loading, which is being jointly developed by the French Commissariat à l’Energie Atomique (CEA Saclay) and by the Joint Research Centre of the European Commission (JRC Ispra).

The present paper addresses two such aspects which have been chosen among the most representative and is organized as follows. Section 2 presents the treatment of general fully coupled boundary conditions by a Lagrange multipliers method, including both permanent and non-permanent conditions such as contacts. Section 3 shows the extension to an Arbitrary Lagrangian Eulerian (ALE) formulation suitable for the treatment of fluid and fluid-structure interaction problems. Finally, Section 4 contains some close-to-realistic applications.

For the sake of brevity, the formulas, bibliography, tables and figures contained in Part I of the present report are not repeated here. When necessary, they are simply referenced by indicating them with the prefix “I-”. For example, Figure I-7 is Figure 7 of Part I, equation (I-12) is equation (12) of Part I, etc. Furthermore, mathematical symbols already introduced in Part I are used without being re-defined.

2. Treatment of boundary conditions

A robust and efficient treatment of boundary conditions is a fundamental ingredient of any numerical formulation which aims at the modeling of complex engineering applications. The variety of condi-

tions that may be encountered in realistic cases—especially in 3D—is almost endless, ranging from simple blockages, to symmetry conditions, to fluid-structure interactions, or to unilateral contacts, to name just a few.

The EUROPLEXUS code offers a very general method for dealing with nearly all types of coupled boundary conditions (see Section 2.1 for a precise definition of coupling), called the “links” formulation and based upon the method of Lagrange multipliers. The main advantage of this method is its robustness. In fact, it allows users to combine several types of conditions in the same calculation without having to check *a priori* for their mutual compatibility. If the conditions are indeed compatible, then a solution is found, otherwise a clear error message is issued. Another characteristic of practical importance is that the reaction forces are obtained as an outcome of the method. This information is often quite interesting for engineers.

The main drawback of the method is that it requires an implicit solution, i.e. it leads to a system of linear algebraic equations to be solved numerically. Note incidentally that this is the only implicit component in the whole strategy for fast transient analysis described in Part I of this work, i.e. for a Lagrangian formulation. As a consequence of implicit treatment, the method may in some cases require a slightly larger computational effort than other techniques based upon a direct (uncoupled) treatment of constraints, such as e.g. penalty methods.

This Section starts by a short description of the “direct” treatment of *uncoupled* boundary conditions within the spatial partitioning algorithm, which has been used in the examples presented in Part I. Then we pass to the more general case of links, i.e. of *coupled* boundary conditions. Similarly to what has been done in Section I-2 for the time integration algorithm, we first give a short description of the basic non-partitioned formulation, i.e. of the Lagrange multipliers method, see Section 2.2, and then present two alternative strategies for the case with spatial partitioning. The first technique, given in Section 2.3, is straightforward but reduces or even potentially vanifies the benefits of partitioning in calculations with many links. The second technique, presented in Section 2.4, is more complicated to implement but preserves much better the advantages of the spatial partition in large realistic applications of industrial size.

2.1 Treatment of uncoupled boundary conditions with partitioning

In this paper the term “boundary condition” is used to indicate a linear constraint, i.e. a relationship of the form:

$$c_1 \upsilon_1 + c_2 \upsilon_2 + \ldots + c_n \upsilon_n = b\,, \tag{1}$$

where c_i and b are known coefficients, possibly function of time, while υ_i are degrees of freedom or dofs (typically the velocity components) of the discrete computational model.

Furthermore, in the following a boundary condition (1) will be referred to as “uncoupled” if it satisfies the following two properties: 1) the n degrees of freedom appearing in the constraint belong all to the *same* node of the model, and 2) there are no other constraints acting on the concerned dofs of the node under consideration. In other words, if one imposes several constraints of the form (1) in the same calculation, they are uncoupled only provided each of them involves a single and a different node.

Perhaps the simplest possible example of boundary condition is the perfect (bi-lateral) blockage of a node along one of the global directions, say the x-axis, which reads:

$$\upsilon_x = 0 . \tag{2}$$

Typically, when using simple models which assume uncoupled constraints the responsibility of ensuring that the imposed conditions are actually uncoupled, i.e. that they satisfy *both* properties stated above, is left with the user. To maximize efficiency, computer codes usually do not perform extensive checks and the user must be aware that applying the simplified models to conditions which are not strictly uncoupled may produce unpredictable (and usually wrong) results—hence the interest of a more robust and general method such as the Lagrange multipliers.

The extension of this class of uncoupled models to spatial partitioning presents no particular difficulty. The only difference with respect to the case without partitioning is that, at each *cycle* of the computation (see [1] for a definition of this terminology) the prescribed constraints must be applied only to the currently active nodes. More precisely, with reference to Section I-3.4.3, direct treatment of uncoupled boundary conditions is performed at point 11 of algorithm P, which may be further detailed as follows:

P.11 For all nodes k in a currently active partition level, i.e. such that $\psi_k \geq m(I)$, evaluate the prescribed external forces at the current time. If such a node is subjected to an uncoupled boundary condition, then apply direct treatment of the condition, i.e. direct calculation of the reaction. This reaction is then treated like if it were a user-imposed external force, i.e. it is assembled in the normal way within the vector of external forces $\underline{F}^{\text{ext}}$.

For example, direct treatment of a blockage such as (2) would simply consist in computing:

$$F_x^{\text{ext}} = F_x^{\text{int}} , \tag{3}$$

i.e. the reaction is equal to the current internal force, which is a known quantity at this point of the time integration algorithm. When the acceleration of the node is computed via eq. (I-4), see point P.12 of the above-mentioned algorithm, one obtains:

$$a_x = (F_x^{\text{ext}} - F_x^{\text{int}}) / m_x = 0 , \tag{4}$$

so that, if the node is initially blocked (i.e. if the user has set its initial velocity and displacement to zero) it stays blocked during the whole transient.

Finally, it should be noted that any node k subjected to directly-treated uncoupled boundary conditions is effectively considered as a free node as far as concerns the calculation of "intrinsic node frequency" ψ_k of Section I-3.2.4 (recall from Part I that this is the frequency of sub-cycling in the explicit time integration process). In particular, eq. (I-35) rather than eq. (I-36) holds for this node and according to eq. (I-37) it is $\lambda_k = 0$. The node is therefore totally ignored in point P.10.11 of the partitioning algorithm, described in Section I-3.4.8.

2.2 Non-partitioned treatment of coupled conditions (links)

Consider now the more general case of *coupled* boundary conditions, i.e. of constraints that, at least potentially, violate one of the two assumptions made in Section 2.1. In other words, either a constraint involves the degrees of freedom of more than one node, or more than one constraint is set on the same degrees of freedom. These conditions are denoted as "links" in the following for brevity.

Most types of links may be formulated under the form of constraints on the velocities, which may be represented by a set of linear relationships:

$$\underline{\underline{C}}\,\underline{\upsilon} = \underline{b} , \tag{5}$$

where $\underline{\underline{C}}$ is a matrix of known coefficients, $\underline{\upsilon}$ is the vector (subset) of linked dofs (typically, the velocities) and $\underline{b}$ is a vector of known values. In general, both $\underline{\underline{C}}$ and $\underline{b}$ may be function of time. It is usually considered preferable to express constraints upon the velocities rather than, e.g., upon the displacements or the accelerations, since the velocity—more precisely, the mid-step velocity—is the fundamental variable upon which the whole central difference time integration scheme described in Part I is based.

A quite general and powerful technique to enforce such constraints is the method of Lagrange multipliers, whose advantages in the context of a standard transient dynamic solution strategy were first recognized at CEA [1]. By following e.g. references [I-10] and [I-11], suppose that a configuration $n+1$ at $t = t^{n+1}$ has been reached. The velocity and acceleration corresponding to this configuration are not known yet. The internal forces and the externally applied loads (natural boundary conditions), however, are known, since they depend only upon the current configuration and upon time.

Consider the subset of degrees of freedom for which essential boundary conditions are imposed. The equilibrium equations for this subset can be written, in analogy with eq. (I-1):

$$\underline{\mu}^{n+1}\underline{\alpha}^{n+1} = \underline{\phi}_{\text{ext}}^{n+1} - \underline{\phi}_{\text{int}}^{n+1} + \underline{r}^{n+1}, \tag{6}$$

where $\underline{\mu}$ is the mass matrix, $\underline{\alpha}$ is the vector of accelerations, $\underline{\phi}_{\text{ext}}$ and $\underline{\phi}_{\text{int}}$ are the vectors of externally applied loads and of internal forces, respectively, and $\underline{r}$ indicates the vector of unknown reaction forces produced by the essential boundary conditions. Note that similar, but distinct, symbols have been used here with respect to eq. (I-1)—e.g. $\underline{\mu}$ instead of $\underline{m}$ for the mass matrix, $\underline{\alpha}$ instead of $\underline{a}$ for the vector of accelerations, etc.—to stress the fact that these equations involve only a (usually small) subset of the degrees of freedom which appear in eq. (I-1).

Now assume that the imposed essential boundary conditions be expressed by a linear set of constraints on the velocities of the form (5), i.e:

$$\underline{C}\underline{\upsilon}^{n+3/2} = \underline{b}. \tag{7}$$

The case of constraints imposed on displacements rather than on velocities can be treated in a similar way and will not be considered here for brevity.

Note that constraints are imposed on the next mid-step velocities ($\underline{\upsilon}^{n+3/2}$), and not on the full-step ones ($\underline{\upsilon}^{n+1}$). This choice has been discussed in reference [I-10]. An easily understandable reason in support of it, is the observation that at the initial time t^0 the velocities $\underline{\upsilon}^0$ are given (they must of course satisfy any imposed essential conditions at that time) and may not be altered: the first values upon which constraints may be imposed are therefore the mid-step ones: $\underline{\upsilon}^{0+1/2}$.

For simplicity, all quantities expressed at time t^{n+1}, corresponding to the current configuration, are indicated without the superscript $n+1$ in the following discussion. With this convention, and by posing:

$$\underline{\phi} \equiv \underline{\phi}_{\text{ext}} - \underline{\phi}_{\text{int}}, \tag{8}$$

the equilibrium equation (6) may be re-written more compactly as:

$$\underline{\mu}\,\underline{\alpha} = \underline{\phi} + \underline{r}. \tag{9}$$

In order to solve this equation for the accelerations $\underline{\alpha}$, one should first determine the unknown reaction forces $\underline{r}$. To this end, use is made of Lagrange multipliers associated with the constraint (7). *Without loss of generality,* the unknown reactions can in fact be expressed as:

$$\underline{r} = \underline{C}^T\underline{\lambda}, \tag{10}$$

where $\underline{\lambda}$ is the vector of Lagrange multipliers. Substituting into (9) yields:

$$\underline{\mu}\,\underline{\alpha} = \underline{\phi} + \underline{C}^T\underline{\lambda}. \tag{11}$$

In order to actually introduce into this equation the constraint (7), which is based upon the velocities, one should first transform (7) into the equivalent form expressed on the accelerations. This is achieved by exploiting the time integration scheme described in Part I.

The central difference integration scheme for the velocity, see the last of eqs. (I-16), can be written as:

$$\underline{\upsilon}^{n+3/2} = \underline{\upsilon}^{n+1/2} + \frac{(\Delta t^n + \Delta t^{n+1})}{2}\underline{\alpha}, \tag{12}$$

where again a different symbol $\underline{\upsilon}$ instead of $\underline{v}$ is used for the subset of linked dofs and, as usual, superscript $n+1/2$ indicates the (known) value at the middle of the current step Δt^n, which spans from the previous configuration n to the current configuration $n+1$ (see eqs. I-14 and I-15). By indicating with γ the (known) coefficient:

$$\gamma \equiv \frac{\Delta t^n + \Delta t^{n+1}}{2}, \tag{13}$$

this equation may be re-written in more compact form as:

$$\underline{\upsilon}^{n+3/2} = \underline{\upsilon}^{n+1/2} + \gamma\underline{\alpha}. \tag{14}$$

By using (14) the constraint (7) becomes:

$$\underline{C}\underline{\upsilon}^{n+1/2} + \underline{C}\gamma\underline{\alpha} = \underline{b}, \tag{15}$$

and hence:

$$\underline{C}\gamma\underline{\alpha} = \underline{b} - \underline{C}\underline{\upsilon}^{n+1/2}. \tag{16}$$

Equations (15) or (16) may be interpreted as equivalent forms of the constraint (7), expressed on the accelerations rather than on the velocities. Note in fact that the old velocities $\underline{\upsilon}^{n+1/2}$ are known and therefore the right-hand side of (16) is a known vector.

Multiplying both members of the equilibrium equation in the form (11) by $\underline{C}\gamma\underline{\mu}^{-1}$ gives:

$$\underline{C}\gamma\underline{\alpha} = \underline{C}\gamma\underline{\mu}^{-1}\underline{\phi} + \underline{C}\gamma\underline{\mu}^{-1}\underline{C}^T\underline{\lambda}, \tag{17}$$

and the Lagrange multipliers can be symbolically obtained from:

$$\underline{C}\gamma\underline{\mu}^{-1}\underline{C}^T\underline{\lambda} = \underline{C}\gamma\underline{\alpha} - \underline{C}\gamma\underline{\mu}^{-1}\underline{\phi}. \tag{18}$$

In conclusion, expression (16) substituted into (18) allows to find the Lagrange multipliers $\underline{\lambda}$, from which the reactions are then obtained with expression (10). Finally, the accelerations $\underline{\alpha}$ along the constrained dofs are explicitly obtained from (6)—in fact recall from Part I that the mass matrix $\underline{\mu}$ is lumped—and the time integration procedure may go on.

In the framework of the present formulation, the method outlined above is implemented as follows. By posing:

$$\underline{D} = \underline{C}\gamma\underline{\mu}^{-1}\underline{C}^{T}; \qquad \underline{s} = \underline{C}\gamma\underline{\alpha} = \underline{b} - \underline{C}\underline{\upsilon}^{n+1/2}; \qquad \underline{w} = \underline{s} - \underline{C}\gamma\underline{\mu}^{-1}\underline{\phi} \ , \tag{19}$$

Eq. (18) becomes:

$$\underline{D}\underline{\lambda} = \underline{w}, \tag{20}$$

where $\underline{D}$ is a matrix, called the *matrix of connections*, and $\underline{w}$ a vector. Both $\underline{D}$ and $\underline{w}$ are known, as it appears from the definitions (19).

The code computes both $\underline{D}$ and $\underline{w}$ at each step, since the coefficients $\underline{C}$ and $\underline{b}$ usually vary with time, due e.g. to large geometrical distortions, in all but the most trivial cases (such as, for example, clamped edges). Thus, the problem reduces to the solution of the linear system of equations (20) in order to find the Lagrange multipliers. Symbolically one may indicate the solution by:

$$\underline{\lambda} = \underline{D}^{-1}\underline{w} . \tag{21}$$

Any standard method for the solution of linear algebraic systems can be used for this purpose. It may be useful to note, to this end, that $\underline{D}$ is a square symmetric matrix. This results from the definition of $\underline{D}$ (first of eqs. 19), and from the fact that $\underline{\mu}$ is a square non-singular symmetric matrix, hence $\underline{\mu}^{-1}$ is also symmetric.

This completes the description of the Lagrange multipliers method in the case of spatially uniform time step. Next, two alternative strategies for extending the method to the case with spatial partitioning are presented.

2.3 Simplified treatment of links in the partition

A straightforward method to deal with links in the presence of spatial partitioning consists in setting the intrinsic nodal level factor $\underline{\psi}_k^{\text{link}}$ for any node k subjected to a link to the maximum level (i.e. to the smallest time increment) in the partition, see eq. (I-36) and point 11 of Algorithm P.10 in Section I-3.4.8. This technique was indeed used in the prototype implementation of spatial partitioning mentioned in Part I.

While, as already stated in Part I of this paper (see Section I-3.2.5), this very simple approach of course preserves all properties of the proposed partitioned integration scheme, it is inherently inefficient since it unnecessarily restrains the local time step, and thus reduces or even potentially vanifies the benefits of the partition. In the limit, if (nearly) all nodes are subjected to links one recovers in practice the case without partitioning, in which the same time increment is used (almost) everywhere.

Although penalizing as far as concerns computational efficiency—and unfortunately this becomes especially true in calculations with *many* links, like is typical of most industrial simulations—such an approach dramatically simplifies the implementation of the model since it guarantees that all nodes subjected to any link belong to the same partition level (the deepest one) and therefore they are updated together (i.e. they stay "synchronized") during the whole calculation.

Thanks to this inherent synchronization process of the linked nodes, the method of Lagrange multipliers described in the previous Section may be applied also in the case of partitioning without any changes, except for the fact that in this case the time increments Δt^n and Δt^{n+1} appearing (also through the coefficient γ) in eqs. (12) to (19) should be replaced by the old and new values, respectively, of the smallest time increment in the partition $\Delta t_{\min}$. Thus the terms γ and $\underline{s}$ appearing in eqs. (13) to (19) become in this case:

$$\gamma \equiv \frac{\Delta t_{\min}^{\text{old}} + \Delta t_{\min}^{\text{new}}}{2}; \qquad \underline{s} = \underline{C}\gamma\underline{\alpha} = \underline{b} - \underline{C}\underline{\upsilon}^{\text{old}}, \tag{22}$$

where $\underline{\upsilon}^{\text{old}}$ represent the old mid-step (or rather mid-cycle in this case) velocities of the linked nodes. The fact that $\Delta t_{\min}$ may vary during the calculation (both at the end of each macro step and within a macro step, i.e. from cycle to cycle) is not a problem. It is properly accounted for, thanks to the presence of both $\Delta t_{\min}^{\text{old}}$ and $\Delta t_{\min}^{\text{new}}$ in the expression (22). In other words, this is the partitioned counterpart of the algorithm for variable time step (in time) detailed in Section 2.2 for the non-partitioned case.

However, a further limitation of this very simple approach (in addition to the loss of efficiency) is that the linked nodes must remain linked over the whole computed transient. In other words, the method deals only with *permanent* links. This limitation stems from the fact that eq. (22) uses the same value $\Delta t_{\min}^{\text{old}}$ for all the linked nodes. Note in fact that, while $\Delta t_{\min}^{\text{new}}$ is guaranteed to be the same for all nodes (since they are indeed linked at the current cycle), the same might not be true for the *previous* cycle, if one would also admit non-permanent links.

2.3.1 Numerical example

As a first numerical example, it is interesting to consider again the Taylor bar impact problem already solved in four different ways within Part I of this paper, see Section I-4.2 and Table I-4. Two additional solutions (labelled 5 and 6) are obtained here and are summarized in Table 1 together with the four solutions obtained in Part I, which are listed again for ease of comparison.

The new solutions 5 and 6 are repetitions of solutions 1 and 2, respectively, but are obtained via the algorithm making use of the simplified coupled treatment of links described in Section 2.3 instead of the direct uncoupled treatment of Section 2.1, indeed a licit approach in the current case. The

obtained results are, as expected, identical as can be verified by comparing Figure 1 with Figure I-19 for the final yield stresses, and Figure 2 with Figure I-20 for the displacements and the velocities. Also the maximum depth of the partition ($d = 8$, corresponding to $\Phi_d = 128$), which is reached towards the end of the calculation, is the same in both cases.

However, one may observe, as can indeed be expected, that the speed-up obtained in the partitioned solution 6 with respect to the non-partitioned solution 5 is much lower than previously. In fact from Table 1 one sees that the cost of solution 5 is identical to that of solution 1—within the accuracy of CPU measurements for such short calculations—while solution 6 is about two times more expensive than solution 2. Consequently, the speed up drops from about 9 to "only" about 4.

This efficiency degradation of the partitioned solution may be precisely understood by comparing the "elements $\times$ cycles", i.e. the total number of element updates, listed in the sixth column of Table 1. Solution 6 performs many more updates than solution 2, although the two calculations use roughly the same number of macro time steps N_p and the same number of cycles M_s to reach the final time.

The difference in performance is due to the fact that the distribution of elements and nodes among the various partition levels is very different in the two cases. This may be observed graphically by comparing the partitioning frequencies (i.e., frequencies of sub-cycling) in Figure 3 with those in Figure I-22. The intrinsic element frequencies φ are exactly the same in both cases, as is normal, but the derived frequencies ψ, $\bar{\varphi}$ and $\bar{\psi}$ (which are those actually used in the update operations) are quite different: there are many more elements and nodes in the lowest partition levels than in the previous case. This is simply because all nodes subjected to blockages, i.e. those at the impacting end and on the axis of symmetry of the bar, are put in the lowest partition level in solution 6, while they are treated as free nodes in solution 2.

2.4 Full treatment of links in the partition

In order to avoid the degradation of performance in partitioned calculations caused by the simplified treatment of links shown in Section 2.3 it is of course sufficient, rather than associating a node k subjected to link(s) with the minimum level of the partition (*absolute* minimum), to associate it with the minimum level among those of the linked-together nodes (*relative* minimum).

The former strategy—while unfortunately too "conservative", as shown in the example of Section 2.3.1—is of course much simpler to implement, since the absolute minimum is known and readily available. The latter strategy is potentially much more efficient, but only if the cost for searching the relative minimum can be kept low enough. Various algorithms for an efficient search are being tested, which take into account and try to exploit the different situations that may be encountered in

practice, e.g. presence or not of non-permanent links (such as unilateral contacts) etc. Their precise description is beyond the scope of the present paper, though.

2.4.1 Treatment of a single link

Consider first for simplicity just one link condition between dofs, represented by the generic (i-th) row of the system (5):

$$c_{ij}\upsilon_j = b_i, \tag{23}$$

where the summation convention over the repeated index j has been assumed. This relationship involves dofs from a certain set of m nodes k, for $k = n_1, \ldots, n_m$. Because of the imposed link, clearly all these nodes must be updated together during the current cycle in the space partitioning procedure, exactly like if they would belong to the same finite element—i.e. exactly like if they were topologically connected. Therefore, for the determination of partitioning frequencies in the presence of a link, one may proceed as follows:

- Compute the intrinsic element frequencies φ_i for all elements, as described in Section I-3.2.3 (see eq. I-34).
- Compute the intrinsic node frequencies ψ_k^{free} for all nodes like if they were free of any links, as described in Section I-3.2.4 (see eq. I-35).
- Find out the nodes $k = n_1, \ldots, n_m$ affected by the link.
- Evaluate the *relative* maximum intrinsic frequency:

$$\psi_{\max} = \max_k(\psi_k^{\text{free}}) \qquad k = n_1, \ldots, n_m. \tag{24}$$

- Set the intrinsic nodal frequencies of all nodes affected by the link to the relative maximum value:

$$\psi_k^{\text{link}} = \psi_{\max} \qquad k = n_1, \ldots, n_m \quad . \tag{25}$$

  Consequently, all the nodes affected by the link will be synchronous, i.e. they will be integrated in time by the same increment δt_L, given by:

$$\delta t_L = \frac{\Delta t_{\max}}{\psi_k^{\text{link}}} = \frac{\Delta t_{\max}}{\psi_{\max}}. \tag{26}$$

- Compute the neighboring element frequencies $\bar{\varphi}_i$ and the neighboring node frequencies $\bar{\psi}_k$ as described in Sections I-3.2.6 and I-3.2.7, respectively (eqs. I-38 and I-40).

Similarly to what already mentioned in Section 2.3, the method of Lagrange multipliers described in Section 2.2 may be applied also in the case of partitioning without any changes, except for the fact that in this case the time increments Δt^n and Δt^{n+1} appearing (also through the coefficient γ) in

eqs. (12) to (19) should be replaced by the old and new values, respectively, of the time increment δt_L defined by eq. (26). Thus the terms γ and $\underline{s}$ appearing in eqs. (13) to (19) become in this case:

$$\gamma \equiv \frac{\delta t_L^{\text{old}} + \delta t_L^{\text{new}}}{2}; \qquad \underline{s} = \underline{C}\gamma\underline{\alpha} = \underline{b} - \underline{C}\underline{\upsilon}^{\text{old}}, \tag{27}$$

where $\underline{\upsilon}^{\text{old}}$ represent the old mid-step (or rather mid-cycle in this case) velocities of the linked nodes.

As already noted in the previous Section, the form (27) is valid only for permanent links, due to the fact that it uses the same value of δt_L^{old} for all the linked nodes. The case of non-permanent links is more general and will be treated in Section 2.4.3.

2.4.2 Presence of multiple links

Since constraints come seldom alone in practice, consider now what happens when there are several links, i.e. a full set of conditions (5), applied to the numerical model. In this case, examining each condition separately and applying eqs. (24) and (25) independently to each relationship might not be sufficient, because the conditions are in general coupled with one another if some of the dofs (and therefore some of the nodes) appear in more than one condition.

In such circumstances, it is clear that the maximization process (24) and (25) must be applied to the *union* of all nodes which are linked to one another, in principle by examining all relationships together. This may be relatively expensive computationally and it is worthwhile to note, incidentally, that the simplified treatment of links of Section 2.3 completely by-passes this issue by simply setting all linked nodes in the deepest partition level (with no attempt to find the actual maximum).

Going too far in implementation details is out of scope in this paper, but it is worthwhile to give at least an outline of the strategies that may be adopted for the relative maximization process. Several alternatives are being tested at the moment.

A promising technique relies upon splitting of the “monolithic” system of constraints (20) into a number of sub-systems, independent from one another. In fact one may observe that in many practical cases the matrix of constraints $\underline{D}$ has a rather narrow bandwidth. For example, if only bilateral blockages are imposed, like in all numerical examples considered so far, the matrix is diagonal so that each constraint could effectively be considered separately from the others and there is no need for a system solution.

In more complex cases, one may still decompose (20) into a (usually large) number of sub-systems, each containing just one or a few constraints involving a relatively small group of nodes, and satisfying the following conditions:

- All nodes in a group are actually linked with one another, i.e. groups may not be further decomposed.
- Each linked node appears in one, and only one, of the groups, i.e. groups are disjoint.

Such a decomposition has been implemented as an option in the code because it greatly facilitates the manipulation and re-arrangement of constraints in the case of non-permanent links such as contacts, whereby new links continuously appear and disappear during the transient calculation. It seems therefore natural to exploit the decomposition also for the relative maximization process. In fact, in this case it is sufficient to find the maximum nodal frequency for each group (not for the whole set of constrained nodes) and to apply it to all nodes in the group. In other words, the m nodes appearing in eqs. (24) and (25) are simply the nodes belonging to the group under consideration.

2.4.3 The case of non-permanent links

As anticipated in the previous Sections, the treatment of non-permanent links—such as e.g. Lagrangian contacts between impacting bodies—within a space partitioning formulation introduces an additional difficulty. This is related to the fact that nodes that at a certain time suddenly become subjected to a new (thus by definition non-permanent) link, and which are necessarily synchronous at the current cycle in virtue of eq. (25) or (26), might have been *asynchronous* during their respective previous sub-cycles.

The situation is schematically illustrated in Figure 5. Two nodes a and b become linked at the current time t^{current}, while they were *not* linked during the previous respective cycles. The link ensures, via eq. (25) or (26), synchronization for the current cycle, indicated by the fact that δt_L^{new} is the same for both nodes. However, the new link may of course have no retro-active effect on the previous cycle, i.e. on the one that has led to the current configuration. In fact in the example the two nodes had two different time increments in the previous cycle, $\delta t_a^{\text{old}} \neq \delta t_b^{\text{old}}$. Clearly, eqs. of the form (22) or (27) may not be applied in such a case, since they assume a single, common value of the “old” time increment δt^{old}, valid for all the linked nodes.

In order to obtain a more general expression that allows the treatment of asynchronous links, the method of Lagrange multipliers introduced in Section 2.2 is reconsidered and eq. (14), expressing the advancement in time of velocities (valid only for the synchronous case), is replaced by the more general (asynchronous) expression:

$$\underline{\upsilon}' = \underline{v} + \underline{\Gamma}\,\underline{\alpha}, \tag{28}$$

where, with reference to Figure 5:

- $\underline{\upsilon}'$ are the new mid-step velocities (all expressed at the same time $\dfrac{t^{\text{current}} + t^{\text{new}}}{2}$, i.e. synchronous).

- $\underline{v}$ are the old mid-step velocities, each one expressed at the old mid-step time of the corresponding node (this time is different in general from node to node). The symbol v is used instead of υ to stress the fact that the nodes were not linked at the previous time.
- $\underline{\alpha}$ are the current accelerations (all expressed at the current time t^{current}).
- $\underline{\Gamma}$ is a matrix of coefficients which replaces the scalar coefficient γ of eq. (14).

The matrix $\underline{\Gamma}$ is a diagonal matrix (hence square and symmetric) that has the following structure:

$$\underline{\Gamma} = \text{diag}\,(\gamma_i) \qquad i = 1, \ldots, n_d, \tag{29}$$

where n_d is the total number of linked degrees of freedom, and the scalars γ_i are defined as:

$$\gamma_i \equiv \frac{\delta t_i^{\text{old}} + \delta t_L^{\text{new}}}{2}. \tag{30}$$

For example, in the case illustrated in Figure 5 one would have:

$$\underline{\Gamma} = \begin{bmatrix} \dfrac{\delta t_a^{\text{old}} + \delta t_L^{\text{new}}}{2} & 0 \\ 0 & \dfrac{\delta t_b^{\text{old}} + \delta t_L^{\text{new}}}{2} \end{bmatrix}, \tag{31}$$

where it is tacitly assumed that the link involves one dof of node a and one of node b (in fact, if the two dofs would belong to the same node, they would be synchronous by definition, the expression for the synchronous case would suffice and the matrix $\underline{\Gamma}$ would be replaced by the scalar γ).

By following the application of the Lagrange multipliers method of Section 2.2 one sees that eqs. (6) to (11) are still valid. The constraint (eq. 7) is re-written as:

$$\underline{C}\underline{\upsilon}' = \underline{b} \tag{32}$$

for consistency with the notation used in the present Section.

Multiplying both members of the equilibrium equation in the form (11) by $\underline{C}\underline{\Gamma}\underline{\mu}^{-1}$ (instead of $\underline{C}\gamma\underline{\mu}^{-1}$ like in the synchronous case), one obtains instead of (17):

$$\underline{C}\underline{\Gamma}\underline{\alpha} = \underline{C}\underline{\Gamma}\underline{\mu}^{-1}\underline{\phi} + \underline{C}\underline{\Gamma}\underline{\mu}^{-1}\underline{C}^T\underline{\lambda}, \tag{33}$$

and the Lagrange multipliers can be symbolically obtained from (instead of 18):

$$\underline{C}\underline{\Gamma}\underline{\mu}^{-1}\underline{C}^T\underline{\lambda} = \underline{C}\underline{\Gamma}\underline{\alpha} - \underline{C}\underline{\Gamma}\underline{\mu}^{-1}\underline{\phi}. \tag{34}$$

In order to actually obtain $\underline{\lambda}$, the term $\underline{C}\underline{\Gamma}\underline{\alpha}$ (instead of $\underline{C}\gamma\underline{\alpha}$) has to be determined. To this end, the constraint (in the form 32) can be used together with the time integration scheme, represented by eq. (28) in place of (14). By replacing (28) into (32) one obtains, instead of (15):

$$\underline{C}\underline{v} + \underline{C}\underline{\Gamma}\underline{\alpha} = \underline{b}, \tag{35}$$

and hence, in place of (16):

$$\underline{\underline{C}}\,\underline{\underline{\Gamma}}\,\underline{\alpha} \;=\; \underline{b} - \underline{\underline{C}}\,\underline{v}\;. \tag{36}$$

Note the subtle difference between the right-hand-side terms of eqs. (16) and (36): the former contains the vector of synchronous old velocities $\underline{v}^{n+1/2}$ while the latter contains the vector of asynchronous old velocities $\underline{v}$ (i.e., each velocity is expressed at a different time, in general).

In conclusion, expression (36) substituted into (34) allows to find the Lagrange multipliers $\underline{\lambda}$, from which the reactions are then obtained with expression (10).

Therefore, in the asynchronous case, the definitions (19) must be replaced by:

$$\overline{\underline{\underline{D}}} = \underline{\underline{C}}\,\underline{\underline{\Gamma}}\,\underline{\underline{\mu}}^{-1}\underline{\underline{C}}^{T}; \qquad \underline{\bar{s}} = \underline{\underline{C}}\,\underline{\underline{\Gamma}}\,\underline{\alpha} \;=\; \underline{b} - \underline{\underline{C}}\,\underline{v}; \qquad \underline{\bar{w}} \;=\; \underline{\bar{s}} - \underline{\underline{C}}\,\underline{\underline{\Gamma}}\,\underline{\underline{\mu}}^{-1}\underline{\phi}\;, \tag{37}$$

where $\overline{\underline{\underline{D}}}$ is the asynchronous form of the matrix of connections, i.e. the one valid also for asynchronous cases. The formal system to be solved (20) becomes:

$$\overline{\underline{\underline{D}}}\,\underline{\lambda} \;=\; \underline{\bar{w}}\,. \tag{38}$$

It is well known (see e.g. references [1], [I-10] or [I-11]) that the matrix of connections as defined by the first of eqs. (19) for the synchronous case is symmetric (always) and positive definite (at least when the imposed constraints are consistent). These properties are important because they affect the numerical methods that may be adopted for the solution of the system of equations (20). It is easily verified, although it is not detailed here for brevity, that the same properties are retained also in the asynchronous form of the matrix of connections, as given by the first of eqs. (37).

To conclude this Section, it is useful to investigate what is the difference in the practical implementation of the Lagrange multipliers method between the general asynchronous case and the synchronous case, which was the only one that had been considered prior to the present work. Only the case of a lumped mass matrix is considered, because it is the type most frequently used in explicit transient analyses. In other words, in the present Section it is assumed that:

$$\underline{\underline{\mu}} \;=\; \mathrm{diag}(\mu_i) \qquad i \;=\; 1, \ldots, n_d\,, \tag{39}$$

where n_d is the total number of linked degrees of freedom.

Then, the inverse mass matrix is simply:

$$\underline{\underline{\mu}}^{-1} \;=\; \mathrm{diag}(1/\mu_i) \qquad i \;=\; 1, \ldots, n_d\,, \tag{40}$$

and the term $\gamma\underline{\underline{\mu}}^{-1}$ which appears in the synchronous expressions (19) of the matrix of connections $\underline{\underline{D}}$ and of the right-hand-side vector $\underline{w}$ is:

$$\gamma\underline{\underline{\mu}}^{-1} \;=\; \mathrm{diag}(\gamma/\mu_i) \qquad i \;=\; 1, \ldots, n_d\,. \tag{41}$$

From the definition (29) of the $\underline{\Gamma}$ matrix, one obtains for the term $\underline{\Gamma}\underline{\mu}^{-1}$ which appears in the asynchronous expressions (37) of the matrix of connections $\overline{\underline{D}}$ and of the right-hand-side vector $\overline{\underline{w}}$:

$$\underline{\Gamma}\underline{\mu}^{-1} = \mathrm{diag}(\gamma_i/\mu_i) \qquad i = 1, \ldots, n_d \,. \tag{42}$$

It may be concluded that, in order to pass from the synchronous to the asynchronous case in the presence of a lumped mass matrix (and for a set of velocity-based links), it is sufficient to introduce the following modification: in the calculation of $\underline{D}$ and of $\underline{w}$, replace γ/μ_i by γ_i/μ_i to obtain $\overline{\underline{D}}$ and $\overline{\underline{w}}$ instead.

2.4.4 Numerical example

A further solution of the Taylor bar impact problem, indicated as number 7 in Table 1, is obtained with the full treatment of partitioned links described in this Section. The obtained results are, as expected, practically identical to those of solutions 5 and 6, as can be observed in Figures 1 and 2 for the final yield stress distribution and for the displacements and velocities, respectively.

This calculation is roughly as efficient as that of solution I-2, which used spatial partitioning and uncoupled constraints. The total number of steps, the total number of cycles and the maximum partition depth are identical and almost the same speed-up is achieved (8.4 instead of 9.2). Thus, the degradation of performance that was observed with the simplified treatment of links in the partition (solution 6) is almost completely avoided. By comparing Figure 4 with Figure I-22, one sees in fact that the pattern of partitioning frequencies is identical to that obtained in solution I-2, instead of the much less favorable one of Figure 3.

3. Extension of spatial partitioning to ALE

The previous Sections have shown the space partitioning algorithm applied to a standard, Lagrangian description in the finite element method. This is adequate and sufficient to cover purely structural problems. However, an important class of problems—namely those involving fluid-structure interaction—requires in addition the modeling of a fluid domain and is usually treated by the more general Arbitrary Lagrangian Eulerian (ALE) formulation.

In ALE, the governing relationships are so-called Euler equations, which express the conservation of mass, momentum (equilibrium) and energy. They are more complex than in the Lagrangian formulation—where only the momentum equation is necessary—since they contain so-called transport terms depending upon the relative velocity between the particles ($\underline{v}$) and the mesh or grid ($\underline{w}$). The grid velocity $\underline{w}$ is in principle completely arbitrary, as indicated in the method's acronym. It is usually chosen (by suitable automatic algorithms) so as to limit excessive fluid mesh distortions.

A detailed description of the ALE formulation is out of scope here, but the interested reader may e.g. consult the publications of J. Donea and co-workers [2-4]. Following these references, time integration of Euler equations may not be achieved as simply as in the Lagrangian description, and is performed by resorting to a fractional step method: each time step is ideally subdivided into three "phases":

- A "Lagrangian" phase, where motion is assumed to be Lagrangian (by ideally setting $\underline{w} = \underline{v}$), so that all transport terms vanish out.
- An implicit phase whereby the obtained pressure value is iterated (just one time in practice) to stabilize the numerical solution.
- A transport phase, where the actual value of the grid velocity $\underline{w}$ is used, as dictated by mesh regularization considerations, to compute the transport terms.

The first two phases do not present any difficulty as concerns space partitioning, compared with a standard Lagrangian calculation. In fact, all calculations in these phases involve just one element at a time, so it is sufficient to perform them just for the "active" elements at each time cycle, as has been seen in Part I.

However, the third phase is more delicate because transport occurs (at least for the mass and energy equations, see below) between couples of adjacent elements and special care must of course be used to make it compatible with the space partitioning procedure. The actual adaptations may depend upon the specific way in which the treatment of transport is implemented. In the following, we consider as an example a specific technique chosen among those most commonly used in explicit codes.

3.1 Transport of mass and energy

Consider for simplicity the mass conservation equation, which contains only the transport term (and in fact it is identically satisfied in a Lagrangian description). The equation reads:

$$\frac{dM^e}{dt} = \frac{d}{dt}\oint_{S(t)} \rho(\underline{w} - \underline{v}) \cdot \underline{n} dS, \tag{43}$$

where M^e is the total mass contained in a closed control volume (typically a finite element e) embedded in the fluid and bounded by a surface S of unit outward normal $\underline{n}$, ρ is the fluid density while $\underline{w}$ and $\underline{v}$ are, as previously defined, the mesh and material velocities, respectively. This equation states simply that the mass variation within the control volume equals the net flux of matter across its surface. A similar transport term occur also in the energy equation.

Upon spatial discretization of the problem, the control volume coincides with a finite element. The velocities and other kinematic quantities are discretized at the element nodes and are interpolated by

linear shape functions over the element volume. Only linear elements are employed for the fluid. Material-related quantities such as the density ρ, the specific internal energy i and the pressure p are element-wise uniform, i.e. they assume just one value for each element. The pressure results from the former two quantities via the fluid's equation of state, assumed to be of the form:

$$p = p(\rho, i). \tag{44}$$

Transport is treated in the third phase of the fractional step time integration procedure, according to the strategy outlined in Figure 6 and described hereafter.

3.1.1 Case without spatial partitioning

Transport between adjacent finite elements is computed side-by-side in 2D or face-by-face in 3D. This means that each finite element I performs transport only with its direct neighbor elements, namely elements J, K, L in Figure 6a. These are defined as the elements that have a full side (i.e. two successive nodes) in 2D or a full face in 3D, in common with the element I under consideration. Note that the other elements appearing in the Figure, which have only one node in common with element I, do *not* exchange mass and energy with I. This assumption introduces a certain approximation, but it allows to greatly simplify the calculations.

Phases 1 and 2 of the time step are performed in a loop over all finite elements in the model, so that the (intermediate) "Lagrangian" state is computed for all fluid elements. Next, another loop over all elements is performed, whereby each element is inspected in turn to compute the transport terms (Phase 3).

In order to save CPU time, transport of mass and energy is computed only once for each couple of neighbor elements, by using the following strategy. When computing transport for element I, its sides are inspected in turn. For each side, one considers the neighbor element, if any. Then, transport is computed only if the index of the neighbor element is larger than the index of the current element, and the computed term is added to one of the two elements (depending on the sign of flux) and subtracted from the other one. For example, assume that $I < J$ and $I < K$, but $I > L$. Then, mass and energy transport between I and J, and between I and K, is evaluated when updating element I, while transport between I and L is evaluated when updating element L.

This strategy saves not only CPU time, since transport between each couple of neighbor elements is computed only once, but also memory storage. In fact, it ensures that at the end of the transport calculation of a generic element, its internal state may be safely updated from the intermediate, "Lagrangian" value to the "true" end-of-step value, because the current element will not be considered any more when computing the following ones (i.e. the elements with larger indexes). Thus, there is no need to keep the Lagrangian and final states in two separate memory locations: just one

location is sufficient since the old values may be immediately replaced by the new ones at the end of each element's transport calculation. Note also that this algorithm does not depend upon the specific numbering of the elements. This last property is of course of utmost importance.

To illustrate the process in a practical case, consider again the 10-element mesh of Figure 6a to which an arbitrary numbering of the elements is added, from 1 to 10, see Figure 6b. There are 12 common sides, i.e. 12 couples of adjacent elements, and correspondingly 12 transport calculations to be performed: these are labeled 'a' to 'l' in the Figure, in the order in which they occur according to the above strategy. Considering again the central element (element 4), one can see that its transport is the sum of three contributions (in the order):

- when updating element 2, the transport between 2 and 4 is computed (term "d");
- when updating element 4, the transport between 4 and 8 is computed (term "g");
- finally, when updating element 4, the transport between 4 and 5 is computed (term "h").

Thus, at the end of transport calculations for element 4, all its transport terms have been evaluated and the element may be definitely updated. This strategy is implemented according to the following computational scheme.

Algorithm T (without space partitioning)

0. Initializations for the current time step: assume for simplicity that the calculation involves only fluid elements and that all elements have already been subjected to phases 1 and 2 of the fractional step algorithm. Consequently, their internal variables represent the intermediate, so-called "Lagrangian" state. We are ready to compute the transport terms. Initialize to zero mass (M) and internal energy ($\Im$) transport arrays $\Delta M_I \leftarrow 0$, $\Delta \Im_I \leftarrow 0$ for all elements $I = 1, \ldots, N_e$. Set the element counter $I \leftarrow 0$.
1. Increment the element counter: $I \leftarrow I + 1$.
2. If $I > N_e$, then **GO TO point 11** below (end of calculation).
3. Set the element's side (or face) counter $S \leftarrow 0$.
4. Increment the side counter: $S \leftarrow S + 1$.
5. If $S > N_{SI}$, where N_{SI} is the number of sides of element I, then **GO TO point 9** below.
6. Let J be the neighbor fluid element to the current side ($J = 0$ if there is no neighbor element). If $J < I$, then **GO TO point 4** above (next side).

7. Compute transport of mass and energy between elements I and J: ΔM_{IJ}, $\Delta \mathfrak{I}_{IJ}$ and add them (with appropriate sign, depending upon the flux orientation) to the transport arrays: $\Delta M_I \leftarrow \Delta M_I \pm \Delta M_{IJ}$, $\Delta \mathfrak{I}_I \leftarrow \Delta \mathfrak{I}_I \pm \Delta \mathfrak{I}_{IJ}$ and $\Delta M_J \leftarrow \Delta M_J \mp \Delta M_{IJ}$, $\Delta \mathfrak{I}_J \leftarrow \Delta \mathfrak{I}_J \mp \Delta \mathfrak{I}_{IJ}$.
8. `GO TO` **point 4** above (next side).
9. Transport terms for element I have all been computed and cumulated in ΔM_I, $\Delta \mathfrak{I}_I$. Update the state of element I from the Lagrangian to the final value.
10. `GO TO` **point 1** above (next element).
11. End of the calculation for the current time step (`GO TO` **point 0** above for the next time step).

3.1.2 Case with spatial partitioning

In order to adapt the mass and energy transport strategy described above to the space partitioning algorithm, special care is required, especially if one wants to retain the strategy of computing the transport just once for each couple of adjacent elements—a technique which appears to be quite attractive since it offers both CPU and memory savings.

A first observation is that, since at the generic cycle not all elements are active in general, and since transport may occur between two elements that are integrated with different time steps, one may not simply cumulate the transport terms over a cycle and then zero them out for the next one. The cumulation process thus extends potentially over the whole macro step, in general (i.e. over several cycles). At the end of each cycle, one must reset the cumulated values only for those elements which have been actually updated during the cycle, i.e. the active elements in that cycle.

The strategy is illustrated by means of a simple example. Consider just one couple of adjacent elements A and B, as depicted in Figure 7a. Assume that the element numbering is such that $A < B$ so that, according to the transport strategy outlined above for the normal case, transport is computed when computing element A (and *not* when computing element B). One may distinguish two cases:

Case 1 (see Figure 7a): $\Delta t_A < \Delta t_B$, say $\Delta t_A = \Delta t_B / 2$, so that element A is integrated twice as often as element B. A macro time step contains two cycles in this case. In the first cycle, only element A is active, while in the second one (as always, at the end of a macro step), all elements are active.

At cycle 1 algorithm T considers only element A, it computes (by considering just the mass transport for simplicity and by assuming a rightward oriented flux) $\Delta M_{AB}^{(1)}$ and sets: $\Delta M_A = \Delta M_{AB}^{(1)}$, $\Delta M_B = -\Delta M_{AB}^{(1)}$. Note that the term $\Delta M_{AB}^{(1)}$ is computed according to the time integration step of element A, thus corresponding to the flux occurring in the first half of the macro step. Finally, the algorithm updates the state of element A at the end of cycle 1.

At cycle 2, algorithm T considers first element A, computes $\Delta M_{AB}^{(2)}$ and sets: $\Delta M_A = \Delta M_{AB}^{(1)} + \Delta M_{AB}^{(2)}$, $\Delta M_B = -\Delta M_{AB}^{(1)} - \Delta M_{AB}^{(2)}$. Note that, again, the term $\Delta M_{AB}^{(2)}$ is computed according to the time integration step of element A, thus corresponding to the flux occurring in the second half of the macro step. The algorithm then updates the state of element A at the end of cycle 2. Next, it considers element B, but it computes no more transport terms since $B > A$. Finally, it updates the state of element B at the end of cycle 2 by using the latest value of ΔM_B obtained above, namely $\Delta M_B = -\Delta M_{AB}^{(1)} - \Delta M_{AB}^{(2)}$.

This behavior is consistent, and requires just a small correction of the non-partitioned algorithm, see point 0 of Algorithm T above:

Correction 1: when space partitioning is chosen, the transport values for an element are not reset systematically at the end of each cycle but only when the element is active, i.e. whenever it is actually updated.

Case 2 (see Figure 7b): $\Delta t_A > \Delta t_B$, say $\Delta t_A = 2\Delta t_B$, so that element B is integrated twice as often as element A. Like in the previous case, a macro time step contains two cycles. However, in the first cycle, only element B is active (not A), while in the second one all elements are active like before.

At cycle 1 algorithm T considers only element B, and it does not compute any transport since $B > A$ (this is obviously wrong!). Finally, it updates the state of element B at the end of cycle 1, but by using a wrong flux (zero in this specific case).

At cycle 2, algorithm T considers first element A, it computes $\Delta M_{AB}^{(2)}$ and sets: $\Delta M_A = \Delta M_{AB}^{(2)}$, $\Delta M_B = -\Delta M_{AB}^{(2)}$. Note that the term $\Delta M_{AB}^{(2)}$ is computed according to the time integration step of element A which, unlike in the previous case, corresponds to the flux occurring over the whole macro step. The algorithm then updates the state of element A at the end of cycle 2. Next, it considers element B, but it computes no more transport terms since $B > A$. Finally, it updates the state of element B at the end of cycle 2 by using the latest value of ΔM_B obtained above, namely $\Delta M_B = -\Delta M_{AB}^{(2)}$.

Thus it appears that, in order to cope correctly with situations like the one outlined in Case 2, one must add further corrections to algorithm T:

Correction 2: when space partitioning is chosen, transport between the current element I and its neighbor element J is computed not only when $I < J$, but also in the case $I > J$, provided J is not active at the current cycle. In fact, if J is active, transport between I and J has already been computed at the current cycle when updating element J.

Correction 3: when space partitioning is chosen, transport between the current element I and its neighbor element J is computed (in any of the cases stated in Correction 2 above) by using the smallest time increment between those of the two adjacent elements.

Thus the ***Algorithm T*** of the previous Section becomes, in the case with space partitioning:

Algorithm T_p (with space partitioning)

0. Initializations for the current time cycle: assume for simplicity that the calculation involves only fluid elements and that all elements which are active in the current time cycle have already been subjected to phases 1 and 2 of the fractional step algorithm. Consequently, their internal variables represent the intermediate, so-called "Lagrangian" state. We are ready to compute the transport terms. Initialize to zero mass and energy transport arrays $\Delta M_I \leftarrow 0$, $\Delta \Im_I \leftarrow 0$ for all elements I that were active in the previous time cycle (if we are at the beginning of a "macro" time step, this applies therefore to all elements). Set the element counter $I \leftarrow 0$.
1. Increment the element counter: $I \leftarrow I + 1$.
2. If $I > N_e$, then **GO TO point 12** below (end of calculation).
3. If element I is not active at the current time cycle, i.e. if $\bar{\varphi}_I < m$, then skip it, i.e. **GO TO point 1** above.
4. Set the element's side (or face) counter $S \leftarrow 0$.
5. Increment the side counter: $S \leftarrow S + 1$.
6. If $S > N_{SI}$, where N_{SI} is the number of sides of element I, then **GO TO point 10** below.
7. Let J be the neighbor fluid element to the current side ($J = 0$ if there is no neighbor element). If $J = 0$ or if $J < I$ and at the same time element J is active at the current cycle, then **GO TO point 5** above (next side).
8. Compute transport of mass and energy between elements I and J: ΔM_{IJ}, $\Delta \Im_{IJ}$ by using the smallest time increment between those of the two neighboring elements (see below for a more detailed description of this process) and add them (with appropriate sign, depending upon the flux orientation) to the transport arrays: $\Delta M_I \leftarrow \Delta M_I \pm \Delta M_{IJ}$, $\Delta \Im_I \leftarrow \Delta \Im_I \pm \Delta \Im_{IJ}$ and $\Delta M_J \leftarrow \Delta M_J \mp \Delta M_{IJ}$, $\Delta \Im_J \leftarrow \Delta \Im_J \mp \Delta \Im_{IJ}$.
9. **GO TO point 5** above (next side).
10. Transport terms for element I have all been computed and cumulated in ΔM_I, $\Delta \Im_I$. Update the state of element I from the Lagrangian to the final value.
11. **GO TO point 1** above (next element).

12. End of the calculation for the current time cycle (**GO TO point 0** above for the next time cycle).

One may easily verify that by applying ***Algorithm T_p*** to both cases in Figure 7, correct results are obtained. The proof is left to the reader for brevity.

The rule stated above in Correction 3 to algorithm T is somewhat subtle and perhaps deserves some clarification on its actual implementation. Transport terms depend upon the relative velocity $\underline{w} - \underline{v}$, as shown above in the time-continuous (i.e., semi-discrete) expressions such as (43). In the time discretization process, the surface integral term occurring both in the mass and in the energy equation gets multiplied by the time increment Δt, so that one typically obtains the velocity flux across a generic side or face S_{IJ} (e.g. between elements I and J) in the form:

$$\Phi_{IJ} = \Delta t \oint_{S_{IJ}} (\underline{w} - \underline{v}) \cdot \underline{n} dS . \tag{45}$$

Thus in the actual calculation use is not made of the relative velocity, but rather of the relative displacement increment, defined as:

$$\Delta \underline{u}_R = \Delta t (\underline{w} - \underline{v}) . \tag{46}$$

The question is then: when space partitioning is active, what is the value of time increment Δt to be used in expression (46)? Note in fact that when considering a generic couple of adjacent elements I and J, it may well be $\bar{\varphi}_I \neq \bar{\varphi}_J$ and thus $\Delta t_I \neq \Delta t_J$.

The answer to the above question is that, as stated by Correction 3 above, the smallest time increment of the two elements must be used to compute the relative displacement increments, so that eq. (46) should be written more precisely as:

$$(\Delta \underline{u}_R)_{IJ} = \Delta t_{IJ} (\underline{w} - \underline{v}) \qquad \text{with} \qquad \Delta t_{IJ} = \min (\Delta t_I, \Delta t_J) = \Delta t_{\min} \cdot \frac{M}{\max (\bar{\varphi}_I, \bar{\varphi}_J)} . \tag{47}$$

3.2 Transport of momentum

Like mass and energy transport terms, transport of momentum is also depending upon the relative velocity $\underline{w} - \underline{v}$, but it assumes the following form (see e.g. references [2-4]):

$$F^e_{tiI} = \int_{V^e} N_I \rho [(\underline{w} - \underline{v}) \cdot \underline{\nabla} v_i] dV , \tag{48}$$

where F^e_{tiI} is the force contribution due to transport of momentum in element e, component i, at node I, and N_I are the element's shape functions.

Note that, unlike expression (43) for the mass transport or the similar one holding for energy transport, momentum transport results from an integral over the element's volume rather than on the element's surface. For this reason, in the case of partitioning momentum transport is always computed

for the current element (which must of course be active at the current time cycle) and does not depend on the status of its neighbor elements. Thus, no special treatment of momentum transport terms appears to be necessary when space partitioning is chosen.

3.3 Mesh rezoning algorithms

Besides the treatment of transport terms that has been detailed in the previous Sections, the ALE formulation requires another ingredient which is present neither in Lagrangian, nor in Eulerian formulations, namely the prescription of the grid (or mesh) velocity $\underline{w}$. As the name of the method indicates, this velocity is arbitrary, and is usually prescribed via so-called “automatic rezoning” algorithms which aim at minimizing distortions in the bulk fluid mesh.

The term *rezoning* should not be mis-understood. In fact, the mesh topology stays constant, and these algorithms should not be confused with re-meshing or adaptive techniques. Only the motion of the mesh is prescribed, but the number of nodes, the number of elements, and the topology (nodes) of each element are not changed.

In the EUROPLEXUS code there are several “automatic” rezoning algorithms available. The ones used most often are perhaps Giuliani’s algorithm (see [5]) and the so-called “mean value” algorithm. In addition, there are also “manual” rezoning directives. This Section will present the modifications in these algorithms which are necessary to cope with spatial time step partitioning.

3.3.1 The mean value algorithm

This algorithm is based upon simple averaging of nodal positions (or of nodal displacements). The algorithm tries to keep each node in the mean position among its direct neighbors, so as to avoid mesh entanglement. The rezoning expression is quite simple and requires only the knowledge of the neighbor nodes (which are sought just once, since topology stays constant) and of their positions (or displacements). No cumulated quantities across different elements are used, so the implementation is quite simple.

To adapt this algorithm to the case of space partitioning, the following two simple modifications are necessary:

- In the loop over nodes subjected to mean value rezoning, only the active nodes (with respect to velocities, i.e. based upon ψ_k) at the current cycle should be treated. In other words, currently inactive nodes should simply be skipped.

- Once computed the optimal rezoning displacement $\delta\underline{x}$ for a node, the algorithm needs to convert this into a corresponding value of mesh velocity $\underline{w}$. To this end, the following expression is used:

$$\underline{w} = \underline{w}^{\text{old}} + \frac{\delta\underline{x}}{\Delta t} \tag{49}$$

where $\underline{w}^{\text{old}}$ is the previous mesh velocity and Δt the current time increment. When space partitioning is active, Δt is computed as follows:

$$\Delta t = \Delta t_{\min}\frac{M}{\psi_k} \tag{50}$$

where ψ_k is the intrinsic node frequency of the node k under consideration. Use is made of ψ_k and not of $\bar{\psi}_k$ here because after all we are dealing with (grid) velocities, although these are computed via displacements.

3.3.2 Giuliani's algorithm

Giuliani's algorithm [5] is more complicated but also more performing than the mean value algorithm. It is based upon geometrical considerations leading to the minimization of the distortion of the grid, and takes into account both the "shear" and the "stretch" of the elements.

The same two corrections listed above for the mean value algorithm must of course be applied also to Giuliani's algorithm. In addition, one should consider the fact that Giuliani's algorithm makes use of cumulated geometric quantities at nodes, obtained by assembling contributions from all neighboring elements, and stored in a dedicated array. Care should therefore be taken with the build-up and the re-initialization of this array, similar to what has been described above when dealing with transport terms.

As concerns the cumulation process, it may be noted that, when an element i is computed (based upon $\bar{\varphi}_i$) its generic node k may or may not be active as concerns velocities and accelerations, based upon ψ_k (in fact they are only guaranteed to be active with respect to configurations, based upon $\bar{\psi}_k$). Therefore, a third correction is needed:

- In the cumulation process, values should be cumulated only for those nodes which are currently active (with respect to velocities, i.e. based upon ψ_k) at the current cycle. In other words, currently inactive nodes should simply be skipped.

As concerns the re-initialization of the dedicated array containing the cumulated values, no further modification is needed since this was already done only for the nodes whose grid velocity is updated, i.e. only for the nodes which are active (with respect to velocities, i.e. based upon ψ_k) at the current cycle.

3.4 Shock tube sample problem

A first set of numerical solutions involving fluid-only calculations is presented, by assuming an ALE formulation, to illustrate the behavior of the space partitioning algorithm compared with calculations without partitioning. The performed tests are based upon a classical shock tube problem, described hereafter.

The chosen test is based upon Harlow and Amsden's [6] analytical solution to the shock tube problem. Consider a rigid cylinder divided into two semi-infinite sections by a diaphragm, as shown in Figure 8a. Initially, an ideal gas is considered to be at rest (particle velocity $u_0 = 0$) on both sides, and the same temperature T_0 is assumed all over. To the left of the diaphragm, the gas is initially at higher density and pressure, ρ_1 and p_1, than those to the right, ρ_2 and p_2. The initial internal energy is i_0 on both sides.

For a polytropic gas the equation of state reads:

$$p = (\gamma - 1)\rho i = R\rho T \tag{51}$$

where $\gamma = c_p / c_v$ is a gas property, c_p and c_v are the specific heats at constant pressure and volume and R is the gas constant. The problem is to determine the variation in time and space of the thermodynamic quantities involved, once the diaphragm is removed at the initial time $t = 0$.

At any later time there is observed a *shock S*, moving to the right, a *contact discontinuity C* moving to the right (this follows the motion of the particles that were initially at the diaphragm) and a *rarefaction wave R*, bounded by points a and b, moving to the left, as shown in Figure 8b. There is no significant length to the system: the appearance of the configuration at a later time is a magnification of an earlier appearance.

Both the velocity u_C (since no gas passes over the discontinuity) and the pressure p_C (otherwise there would be an infinite acceleration) are continuous across the contact discontinuity, but the density passes from ρ_L to ρ_R. Through the equation of state (51), the internal energies i_L, i_R (and the temperatures T_L, T_R) may be determined if p_C, ρ_L and ρ_R are known. Thus, the four unknowns of the problem are ρ_L, ρ_R, u_C and p_C.

To describe the solution in the rarefaction wave region, assume a normalized coordinate ξ that varies from 0 to 1 between b and a, as shown in Figure 8c. It may be found analytically that in this region the density ρ is linear only when $\gamma = 3$, but for $\gamma < 3$ (as for many gases), $\rho(x)$ is concave upwards. The specific set of parameter values assumed to solve the shock tube problem is summarized in Table 2.

3.4.1 Analytical results

The analytical solution is worked out in [6] and is not repeated here for brevity. As noted above, the solution is self-similar, i.e. there is no significant length to the system. However, in order to compare it with numerical solutions, one must assume a well-defined geometry and one or more reference times at which solutions are compared. The problem is one-dimensional in nature. An initial length of the tube of 50.0 units is assumed. The initial position of the diaphragm is at the middle of the tube ($x_{D0} = 25.0$). The main reference time chosen is $t_1 = 10.\times10^{-3}$. The rarefaction wave region is subdivided into 10 equally spaced zones, and the values of the variables of interest are computed at each point. In Table 3 the main results of the analytical solution are listed. The sampled values in the rarefaction zone are listed in Table 4.

3.4.2 Numerical solutions

The problem has been numerically solved several times by using either 2D or 3D meshes that simulate a 1D tube. The spatial discretization consists of 100 square elements (cubes in 3D) of side length 0.5 in a row, for a total length of 50.0 units. The boundary conditions simulate a rigid tube: all nodes are blocked in the y direction (and also in the z direction, in 3D), and the nodes at the two extremities ($x = 0.0$ and $x = 50.0$) are blocked in the x direction. This causes reflections of the pressure waves at the extremities for times greater that the travel time to reach the ends. Since the analytical solution does not take into account such reflections (an infinite tube is assumed) one may compare solutions only up to that time. All these blockages are imposed by the uncoupled boundary conditions technique described in Section 2.1.

Two of the obtained numerical solutions are summarized in Table 5. Solution 1 uses an ALE description whereby all nodes are treated as ALE except those initially placed on the diaphragm and at each extremity of the tube, which are treated as Lagrangian. This choice forces the rezoning algorithm to continuously regularize the mesh following the motion of the diaphragm, i.e. of the contact discontinuity. In fact, treating the whole mesh as ALE would not be interesting since the mesh is initially regular so the rezoning algorithms would have nothing to do in this specific case. This solution uses uniform time step in space. Solution 2 is identical to solution 1, except for the fact that it uses spatial partitioning.

Figure 9a compares the fluid pressure distribution along the tube at time t_1 in the two numerical solutions (solid curves) and against the analytical solution (dashed curve). Figure 9b shows the same comparison for the density and Figure 10a for the specific internal energies. In all these diagrams, the two numerical solutions obtained without and with partitioning are almost identical. Further-

more, their agreement with the analytical solutions is quite good, especially as concerns the pressure, which is the primary quantity of interest in fluid-structure interaction calculations.

Figure 10b illustrates the behavior of the mesh rezoning algorithm, which is practically identical in the two solutions. The plotted curve represents the axial displacement of the mesh nodes. The highest point of the curve corresponds to the contact discontinuity, which is the point with the largest displacement. Superposed to the diagram are also the deformed meshes obtained in the two solutions, which are again identical. In these calculations Giuliani's rezoning algorithm is used (see Section 3.3.2), but similar results are obtained also with the mean value algorithm of Section 3.3.1.

These results confirm that also in ALE the proposed spatial partitioning technique does not introduce any degradation of the quality and accuracy of numerical results, while it may considerably reduce the computational costs. This particular numerical example requires too little CPU time for an accurate measurement of speed-up. However, from Table 5 one may see that, at least qualitatively, there is a considerable reduction of cost (of the order of 3).

The numerical tests have highlighted the fact that a correct treatment of transport in the space partitioning technique, as defined in Algorithm T_p of Section 3.1.2, is crucial. Even a small imprecision, such as failing to implement any of the corrections illustrated in that Section, produces a severe degradation of results with respect to the non-partitioned solution.

The reason for the considerable speed-ups obtained in these calculations becomes apparent by inspecting the partitioning frequencies of sub-cycling. For example, Figure 11a shows the spatial distribution of element intrinsic frequencies along the tube at three times: $t = 0$, $t = 10$ ms and $t = 20$ ms. Figure 11b shows the neighboring node frequencies. More precisely, the figures represent the logarithm in base 2 of the frequencies rather than the frequencies themselves, i.e. in practice the corresponding partition level minus 1, according to eq. (I-33). One can see that at the beginning of the calculation the same frequency holds over the whole mesh, so that the partition contains just one level and is *de facto* inactive. This is because the mesh is initially uniform and the sound speed is the same in both regions of the tube, despite the differences in initial density and pressure. Therefore, the stability time increment is initially the same in all elements. However, as the calculation proceeds the elements to the right of the shock front become relatively shorter (despite the regularizing action of the automatic rezoning algorithm) and at the same time the sound speed varies locally so that the spatial partitioning starts to operate and at the end of the transient quite an interesting overall speed-up of the calculation is achieved.

Additional solutions of the shock tube problem have been obtained not only in ALE, but also with either a Lagrangian or an Eulerian formulation. They are not illustrated here for brevity, but in all

cases the solutions with partitioning are nearly identical to the corresponding ones with uniform time step. Some very small discrepancies are indeed observed, but they are simply due to the fact that the effective average stability safety factor is quite different (i.e. much larger) in partitioned solutions than in non-partitioned ones, as illustrated in the purely structural 1-D wave propagation example considered in Part I, see Section I-4.1.

It may be interesting to mention that in the shock tube problem a substantial speed-up is obtained even in a purely Eulerian solution using a uniform mesh. In this case the benefit does not come from the shortening of elements (since obviously the mesh stays completely fixed all the time), but from the fact that the sound speed varies a lot in fluids depending on local physical conditions, which is normally not the case in structural materials. Thus, in general one may expect even larger speed-ups in fluid-structure calculations than in purely structural ones.

4. Numerical examples

This Section presents a set of relatively complex numerical examples that allow to estimate the benefits that may be expected from the spatial partitioning technique in close-to-real applications. All the examples, including the ones presented in Part I, have been performed by means of the EUROPLEXUS code running on a Pentium IV PC with a 3 GHz clock speed and 1 GB of RAM.

4.1 Explosion in an helicoidal tube

A helicoidal tube closed at both ends is filled with gas at uniform pressure, except at the larger extremity where the same gas has an initially larger pressure, to simulate an explosion. Several calculations are performed, assuming either 2 or 4 complete whorls of the helix (see Figure 12) and with either rigid or deformable tube walls.

The overall size of the geometrical model is about 2.6 m. The high-pressure fluid is a perfect gas with an initial density of 1.22 Kg/m^3, an initial pressure of 10 bar and a ratio of specific heats $\gamma = c_P/c_V$ of 1.269. The low-pressure fluid is the same gas at an initial density of 0.1237 Kg/m^3 and at an initial pressure of 1 bar. In the calculations with deformable walls, these are assumed to have a thickness of 1 mm and are made of an elastic material with density 8000 Kg/m^3, Young's modulus 2×10^{11} Pa and Poisson's coefficient 0.3.

The computational mesh for the 4-whorls case consists of 12528 hexahedra and 4032 prisms in the fluid region and 10772 triangles in the structure region, for a total number of 22778 nodes. In the calculations with deformable walls a nodally conforming fluid-structure interface is assumed, i.e. the structural nodes are located at the same positions as the fluid nodes on the surface of the tube.

The performed calculations are summarized in Table 6. In all solutions, the appropriate boundary conditions along the tube surface are imposed in a fully coupled manner by means of the links models presented in Section 2.2, for the solutions with uniform step, and in Section 2.4 for the solutions with spatial partitioning.

Four reference solutions (number 1, 3, 5 and 7) using a spatially uniform time step are obtained by using the two geometries (2 whorls or 4 whorls) without and with deformable structural walls. These solutions are then repeated with spatial partitioning (solutions 2, 4, 6 and 8). In the first two solutions, which use the coarser mesh and rigid walls (fluid-only) a relatively modest speed-up of about 1.5 is obtained, despite the fact that there is a ratio of 16 between the time steps of the largest and of the smallest element at the initial time (raising up to 32 at some instants during the transient calculations). This is due to the fact that in this example the mesh is gradually and smoothly refined passing from the coarser to the finer region, see Figure 12. Therefore, the distribution of elements in the various partition levels is quite uniform. To obtain more favorable speed-ups one should refine the mesh much more abruptly.

By passing to the 4-whorls mesh (solutions 3 and 4) the speed-up raises to 2.8, thanks to the increased ratio between the maximum and minimum element sizes.

Further ameliorations of relative computational efficiency are obtained when the structural walls are included in the model, see solutions 5 to 8, thus resulting in calculations with fluid-structure interaction. The obtained speed-ups become 3.1 for the 2-whorls case and 4.9 for the 4-whorls case, thanks to the fact that the structural elements used to model the tube walls are more rigid—and therefore have smaller intrinsic critical time steps—than the corresponding fluid elements.

Some results of reference solution 1 (2-whorls, fluid-only mesh) are illustrated in Figure 13, which shows the fluid pressures in the tube at regular intervals of 1 ms during the transient. The corresponding solution with spatial partitioning yields results that may be considered identical from an engineering viewpoint. For example, the fluid pressure levels at the centres of the initial and final cross-sections of the tube obtained in solutions 1 (dashed lines) and 2 (solid lines) are compared in Figure 14a. The curves are practically superposed to each other.

The corresponding pressure results in the case with 4-whorls mesh and deformable walls (solutions 7 and 8) are compared in Figure 14b. The pressure peak generated by the pressure wave reflection at the narrow end of the tube is much higher and sharper than in the previous case, because the final cross-section is smaller and thus the pressure amplification phenomenon is stronger. The difference between the non-partitioned (dashed lines) and the partitioned solution (solid lines) appears larger than in the previous case, but much of this apparent discrepancy is due to the fact that in the parti-

tioned solutions results are stored (and plotted) only at the end of each “macro” step, so there are many less data points in the curves.

This fact is confirmed by comparing the displacements of the tube walls, see Figure 15 (these curves are much smoother than the pressure curves). One can see in fact that the two solutions are identical from an engineering viewpoint. Much larger displacements occur at the large-diameter end of the tube than at the small-diameter end, as is natural. Moreover, at the first end the motion occurs mainly in the direction normal to the tube cross-section (Y-component) while at the second end the three components of displacement have the same order of magnitude. This is due to the fact that the tube is completely unrestrained in space. Upon propagation of the pressure wave the tube tends to “straighten up” by slightly unfolding the whorls and the effect is proportionally larger at the small-diameter end, which undergoes the largest pressure peak.

4.2 Explosion in a corridor

The second example, taken from reference [7], aims at showing that spatial partitioning may be combined with other techniques that are useful to increase computational efficiency, such as the use of non-conforming interfaces in fluid-structure interaction problems, see [7] for details. The model simulates an explosion in a narrow corridor, which is in communication at one extremity with a much larger room, as sketched in Figure 16. The size of the room is 10×10 m. The structure is 0.01 m thick, the structural material has density 8000 kg/m^3, Young's modulus 2×10^{11} Pa, yield limit 4×10^{8} Pa and plastic modulus 2×10^{9} Pa. The low-pressure gas is air at an initial density 1 kg/m^3, the high-pressure gas has an initial density 10 kg/m^3, and the ratio of specific heats for both gases is 1.4. Three walls of the square-shaped room are assumed as rigid, while the fourth one is deformable. The two walls separating the corridor from the room are also deformable.

Because of the narrow and relatively complicated flow path from the corridor to the room, a fine fluid mesh must be assumed in the corridor, while in the room the mesh may be much coarser. Two meshes, A (conforming) and B (non-conforming), are tested, see Figure 17, and four solutions are obtained, two with uniform time step (solutions 1 and 3, already obtained in [7]) and two with spatial partitioning (solutions 2 and 4). All solutions are summarized in Table 7.

The first two solutions use a conforming fluid-structure interaction (FSI) model, with the mesh shown in Figure 17a (5497 fluid elements and 74 structural elements). The structural elements in the corridor walls are very small. The last two solutions use a nonconforming FSI model, which allows to use a uniform, much coarser, structural mesh (3513 fluid elements and only 14 structural elements). Note that in this case not only the structural but also the fluid mesh may be kept coarser than in the conforming case, as is shown in Figure 17b. The reason is that in the conforming model the

fluid on both sides of the deformable corridor walls must have the same size, and thus the fluid in the room close to the corridor wall is forced to use an unnecessarily fine mesh, while this constraint does not exist in the nonconforming model. In this way the number of fluid elements used in the nonconforming analysis may be reduced by 1/3 and a global speed-up factor of as much as 8.6 is obtained in this case, as shown in Table 7, without sacrificing the quality of results. Compare for example the fluid pressures obtained in the four calculations at regular intervals of 10 ms, shown in Figure 18.

By using spatial partitioning with the conforming mesh (solution 2) a speed-up factor of 2.8 is achieved, while the combined effect of partitioning and a non-conforming mesh (solution 4) raises the speed-up to a value of 11.2.

4.3 Building vulnerability study

The third and final numerical example is taken from building vulnerability studies which are ongoing at JRC, in an attempt to show a quasi-realistic application. The purpose of these studies is to assess the vulnerability of civil engineering buildings to possible explosive events of various origin. The assumed geometrical model is a baptistery located in the Greek island of Kos. The choice of this particular structure has no real relevance as far as explosions are concerned: it is taken because a finite element mesh of this relatively complex building is readily available from seismic studies performed in a previous project.

The overall geometry of the building is shown in Figures 19 (outside views), 20 (transparency views) and 21 (inner views). The building has a relatively complex geometry, with a large central dome supported by 8 columns and surrounded by 8 smaller domes.

The numerical analysis presented below is not completely realistic since the used mesh is relatively too coarse, and the material properties assumed for the explosive (modelled by a high-pressure perfect gas) and for the structure (assumed as homogeneous and elastic) are drastically simplified. However, this is consistent with the purpose of the present work, which is that of showing the potential advantages of space partitioning.

The structural model consists of 6894 solid 4-node tetrahedra and 5378 solid 6-node prisms. To model the explosion, the internal volume of the building is filled by a fluid finite-element mesh, representing the air, by assuming for simplicity a conforming fluid-structure interface. An explosive bubble with a 100 bar initial pressure and a volume of about 1.3 m^3 is placed between the two columns in front of the main entrance. To model the fluid outflow from the main door and from the window after the explosion, the fluid mesh is extended for a short distance to the outside of the structure. The floor of the building and the surrounding ground are supposed rigid, while the external envelope

of the fluid mesh is subjected to special absorbing conditions. The scope is to avoid any spurious reflections of pressure waves along the external surface of the numerical model and thus to approximate the behavior of the real (ideally infinite) atmosphere. The fluid mesh consists of 72748 fluid 4-node tetrahedra. The total number of nodes in the model is 22605 and the total number of elements is 90234, including 5214 special “boundary-condition elements” used to impose the above-mentioned absorbing conditions. The simulation is performed for an overall time duration of 50 ms.

Two solutions are obtained for this problem, as reported in Table 8. The first solution uses a uniform time step while the second one uses spatial partitioning. Some typical numerical results of this type of calculations are presented in Figure 22: a) the pressure field and velocity vectors in the fluid at 25 ms, b) the same seen from below and c) the displacement norm (in the form of iso lines) and the velocity vectors in the structure, at the same time.

As shown in Table 8, by partitioning the CPU time passes from about 12000 to 1600 s, yielding a speed-up factor of about 7, despite the fact that the chosen mesh is relatively uniform.

Even larger speed-ups may be expected as the numerical simulation is made more and more realistic. In fact, this typically requires, for example, to further refine the mesh locally in order to represent more precisely the explosive charge or some critical parts of the structure.

5. Conclusions

The final remarks presented in Section I-5 (see Part I) are repeated here for completeness and are integrated with the conclusions relative to Part II.

The numerical examples of Sections I-4 and 4 indicate that the spatial partitioning technique proposed in this paper has the potential for obtaining significative reductions of CPU time in transient explicit analyses, without at the same time losing any of the good properties of the classical uniform-step time integration algorithm.

Since the partitioning mechanism is fully automatic, explicit code users may activate it without any effort, simply by specifying an option in the input data file. The actual speed-up obtained with respect to a uniform-step solution depends of course upon the degree of non-uniformity of the time step in the chosen numerical model.

Partitioned solutions are by construction as safe and as accurate as uniform-step solutions because, according to Section I-3.1, the proposed partitioning technique introduces no (additional) approximations in the time integration algorithm. In special cases of concern, e.g. when using for the first time a specific model never tested before with partitioning, there is always the possibility of obtaining also the corresponding non-partitioned solution for comparison. As shown in the examples, the

two solutions should be identical except for irrelevant effects—in particular irrelevant from the engineering viewpoint—such as e.g. higher-frequency numerical oscillations in purely elastic cases.

It is important to underline that, besides allowing solution of existing test cases in less CPU time, the partitioning technique has the potential for opening the way to simulations that are simply out of reach with the uniform-step algorithm. A significative change of mentality becomes possible. Experienced users of explicit codes are traditionally quite concerned by mesh size because they know that the cost of a calculation is, roughly speaking, inversely proportional to the size of the smallest element in the model. With spatial partitioning this constraint almost disappears and, at least potentially, one becomes free to refine the mesh locally almost at will to reach the desired precision, by paying only modest CPU overheads.

Of course, in order to actually obtain all these benefits in realistically complex applications, the "core" spatial partitioning technique presented in Part I is not sufficient. The method must be "industrialized" and several technical aspects of great practical importance must be dealt with. The present paper has addressed two such aspects, namely the treatment of general fully coupled boundary conditions by a Lagrange multipliers method, including both permanent and non-permanent conditions such as contacts, and the extension to an Arbitrary Lagrangian Eulerian (ALE) formulation suitable for the treatment of fluid and fluid-structure interaction problems.

As shown in Sections 2 and 3, the modifications needed to implement spatial partitioning in a typical explicit computer code with respect to the version for uniform time step involve mainly the routine which drives the time integration loop (where most of the modifications are concentrated), plus a small number of ancillary routines (which require comparatively few modifications). Indeed, most of the models need no modification at all, at least provided they are written in a "clean" and rigorous manner, i.e., for example, provided they do not make *direct* use of the notion of *time increment* internally. This systematic model validation work is ongoing within the EUROPLEXUS code and will gradually open the way to more and more complex applications of partitioning, up to the solution of realistic industrial problems.

Perhaps the most important extension still missing is the application of partitioning to Finite Volume rather than to Finite Element formulations. This has not been attempted yet but, at least conceptually, it should not present any overwhelming difficulties.

6. References

[1] M. Lepareux, B. Schwab, A. Hoffmann, P. Jamet, H. Bung: "*Un Programme Général pour l'Analyse Dynamique Rapide—Cas des Tuyauteries*", Colloque: Tendances Actuelles en Calcul des Structures, Bastia, 6-8 November, 1985.

[2] J. Donea, S. Giuliani, J.P. Halleux: "*An arbitrary Lagrangian-Eulerian finite element method for transient dynamic fluid-structure interactions*", Computer Methods in Applied Mechanics and Engineering, Vol. 33, pp. 689-723, 1982.

[3] J. Donea: "*Arbitrary Lagrangian-Eulerian finite element methods*", in Computational Methods for Transient Analysis, Vol. 1 in Computational Methods in Mechanics, edited by T. Belytschko and T.J.R. Hughes, North Holland, 1983.

[4] J. Donea, A. Huerta: "*Finite Element Methods for Flow Problems*", Wiley, 2003.

[5] S. Giuliani: "*An algorithm for continuous rezoning of the hydrodynamic grid in Arbitrary Lagrangian Eulerian computer codes*", Nuclear Engineering and Design, Vol. 72, pp. 205-212, September 1982.

[6] F.H. Harlow, A.A. Amsden: "*Fluid Dynamics*", Los Alamos Scientific Laboratory of the University of California, Los Alamos, New Mexico, Report LA-4700, UC-34, June 1971.

[7] F. Casadei, S. Potapov: "*Permanent Fluid-Structure Interaction with Non-Conforming Interfaces in Fast Transient Dynamics*", Computer Methods in Applied Mechanics and Engineering, Vol.193, issues 39-41, pp. 4157-4194, October 2004.

Solution	Description	Steps (N_p)	Cycles (M_s)	Max level frequency (Φ_d)	Elements × cycles	CPU (s)	Speed-up (theor)/ actual
I-1 (bamd01)	Lagrangian, uniform step, uncoupled constraints	59886	—	—	14971750	61.89	—/—
I-2 (bamd03)	Lagrangian, spatial partitioning, uncoupled constraints	1162	85537	128	1296171	6.75	11.6/9.2
I-3 (bamd05)	Lagrangian, domain decomposition, uncoupled constraints	1152	—	—	—	6.05	—/10.2
I-4 (bamdal)	ALE, uniform step, uncoupled constraints	1950	—	—	487750	4.34	30.7/14.3
5 (baim01)	Lagrangian, uniform step, links model	59886	—	—	14971750	60.56	—/—
6 (baim03)	Lagrangian, spatial partitioning, links model (simplified treatment)	1162	85473	128	5387250	15.22	2.8/4.0
7 (baim13)	Lagrangian, spatial partitioning, links model (full treatment)	1162	85537	128	1296171	7.20	11.6/8.4

Table 1 - Numerical solutions for the Taylor bar impact example

ρ_1	1.22
ρ_2	0.1237
γ	1.269
p_1	1.0×10^6
p_2	1.013×10^5
$i_1 = i_2 = i_0$	3.046×10^6

Table 2 - Specifications assumed for the numerical solutions of the shock tube problem

u_C	925.4	v_S	1672.
p_C	2.927×10^5	v_a	30.12
ρ_R	0.2771	v_b	−1020.
ρ_L	0.4635	c_0	1020.
i_R	3.928×10^6	c_a	895.2
i_L	2.348×10^6	c_b	1020.

Table 3 - Analytical solution of the shock tube problem

ξ	$x(t_1)$	c	ρ	p	i
.0	14.80	1.020E+03	1.220E+00	9.996E+05	3.046E+06
.1	15.85	1.007E+03	1.114E+00	8.903E+05	2.972E+06
.2	16.90	9.948E+02	1.015E+00	7.917E+05	2.899E+06
.3	17.95	9.824E+02	9.245E-01	7.031E+05	2.827E+06
.4	19.00	9.699E+02	8.409E-01	6.234E+05	2.756E+06
.5	20.05	9.575E+02	7.639E-01	5.519E+05	2.686E+06
.6	21.10	9.450E+02	6.930E-01	4.878E+05	2.616E+06
.7	22.15	9.326E+02	6.280E-01	4.304E+05	2.548E+06
.8	23.20	9.201E+02	5.683E-01	3.792E+05	2.480E+06
.9	24.25	9.077E+02	5.136E-01	3.334E+05	2.414E+06
1.0	25.30	8.952E+02	4.635E-01	2.927E+05	2.348E+06

Table 4 - Analytical solution in the rarefaction wave region for the shock tube problem

Solution	Description	Steps (N_p)	Cycles (M_s)	Max level frequency (Φ_d)	Elements × cycles	CPU (s)	Speed-up (theor)/ actual
1 (shoa24)	ALE, uniform step, uncoupled constraints	531	—	—	53200	0.64	—/—
2 (shop24)	ALE, spatial partitioning, uncoupled constraints	82	761	16	30774	0.22	1.73/2.91

Table 5 - Numerical solutions for the shock tube problem

Solution	Description	Steps (N_p)	Cycles (M_s)	Max level frequency (Φ_d)	Elements × cycles	CPU (s)	Speed-up (theor)/ actual
1 (soli05)	2 whorls, EULE (fluid only), uniform step	2437	—	—	20186640	191.0	—/—
2 (soli07)	2 whorls, EULE (fluid only), spatial partitioning	161	3532	32	12005774	131.9	1.68/1.45
3 (soli11)	4 whorls, EULE (fluid only), uniform step	22376	—	—	370563120	3309.9	—/—
4 (soli12)	4 whorls, EULE (fluid only), spatial partitioning	106	32000	512	105456848	1200.1	3.51/2.76
5 (solt05)	2 whorls, ALE (fluid/structure), uniform step	14219	—	—	197202960	2116.8	—/—
6 (solt07)	2 whorls, ALE (fluid/structure) spatial partitioning	174	24704	128	54788541	687.6	3.60/3.08
7 (solt15)	4 whorls, ALE (fluid/structure) uniform step	146923	—	—	4015726768	35191.4	—/—
8 (solt16)	4 whorls, ALE (fluid/structure), spatial partitioning	104	209920	2048	496122799	7157.8	8.09/4.92

Table 6 - Numerical solutions for the helicoidal tube problem

Solution	Description	Steps (N_p)	Cycles (M_s)	Max level frequency (Φ_d)	Elements × cycles	CPU (s)	Speed-up (theor)/ actual
1 (dens01)	conforming mesh, uniform step	8400	—	—	46801971	313.6	—/—
2 (corr01)	conforming mesh, spatial partitioning	52	12800	256	7800294	111.6	6.0/2.8
32 (dens02)	non-conforming mesh, uniform step	1433	—	—	5057718	36.4	9.3/8.6
4 (corr02)	non-conforming mesh, spatial partitioning	50	2496	64	3154534	28.1	14.8/11.2

Table 7 - Numerical solutions for the explosion in the corridor problem

Solution	Description	Steps (N_p)	Cycles (M_s)	Max level frequency (Φ_d)	Elements × cycles	CPU (s)	Speed-up (theor)/ actual
1 (test14)	uniform step	16060	—	—	1449248274	11648	—/—
2 (test13)	spatial partitioning	100	25600	256	112280998	1650	12.9/7.1

Table 8 - Numerical solutions for the explosion in the building

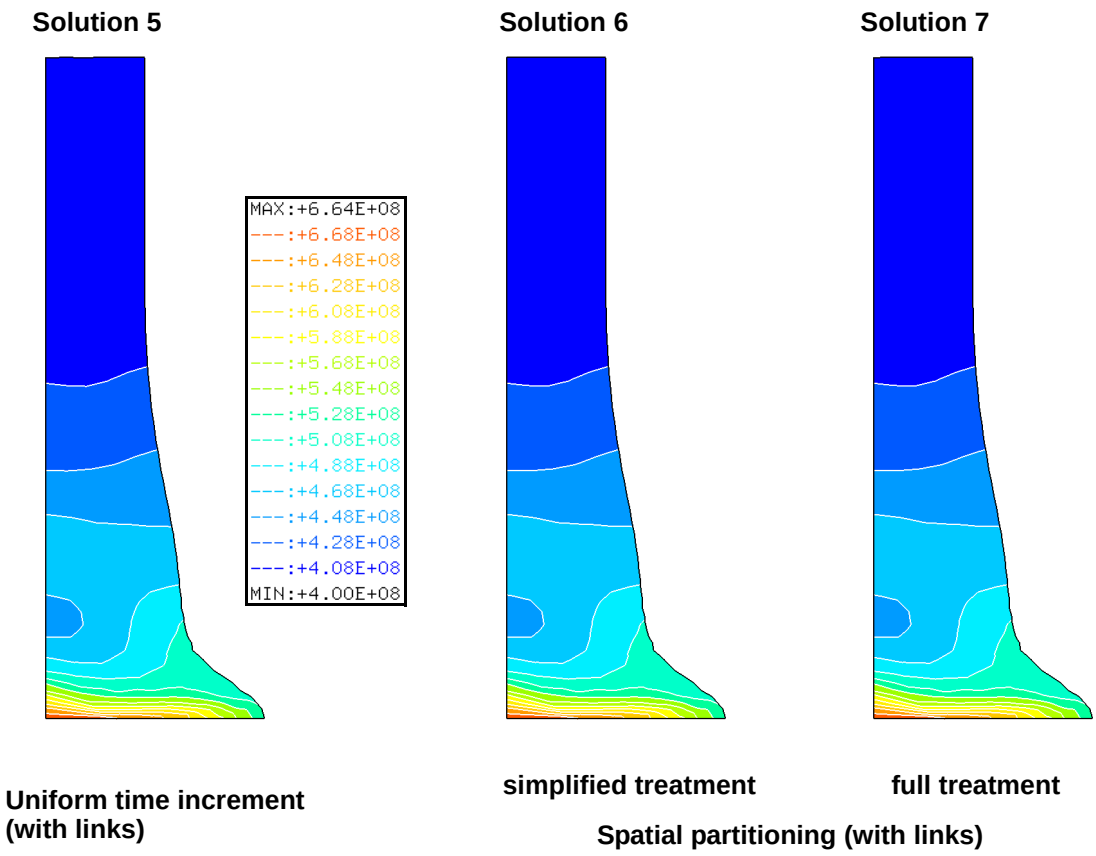
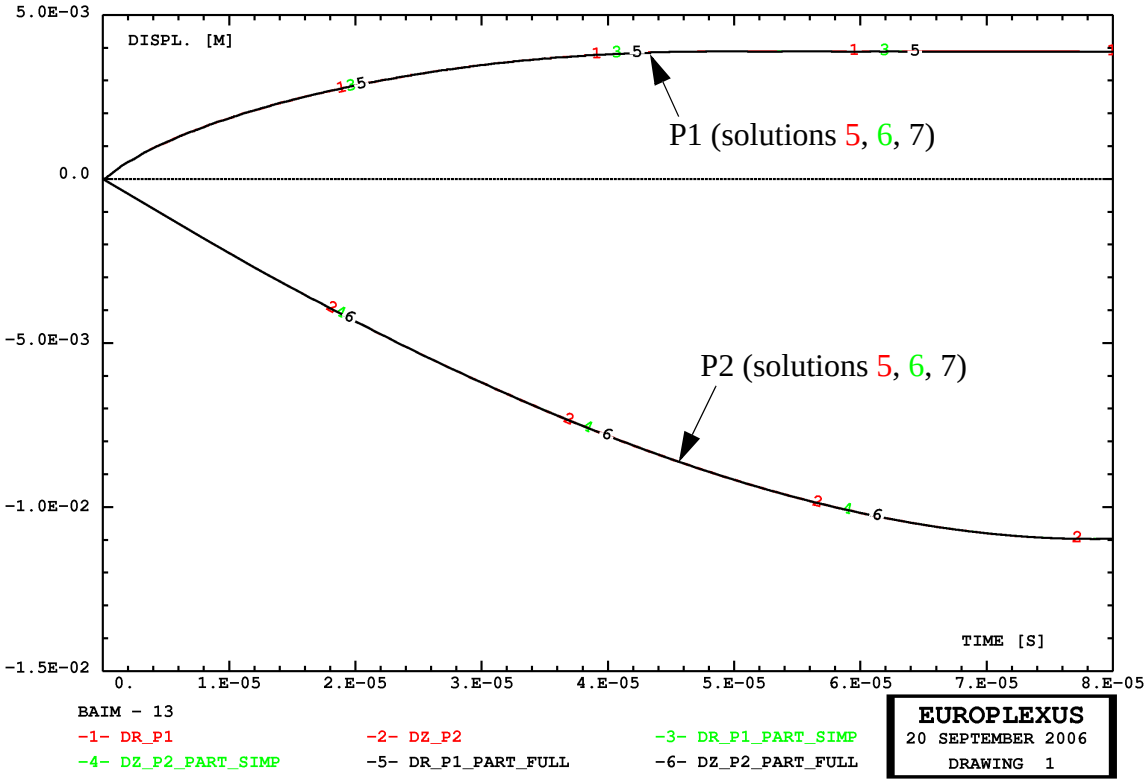
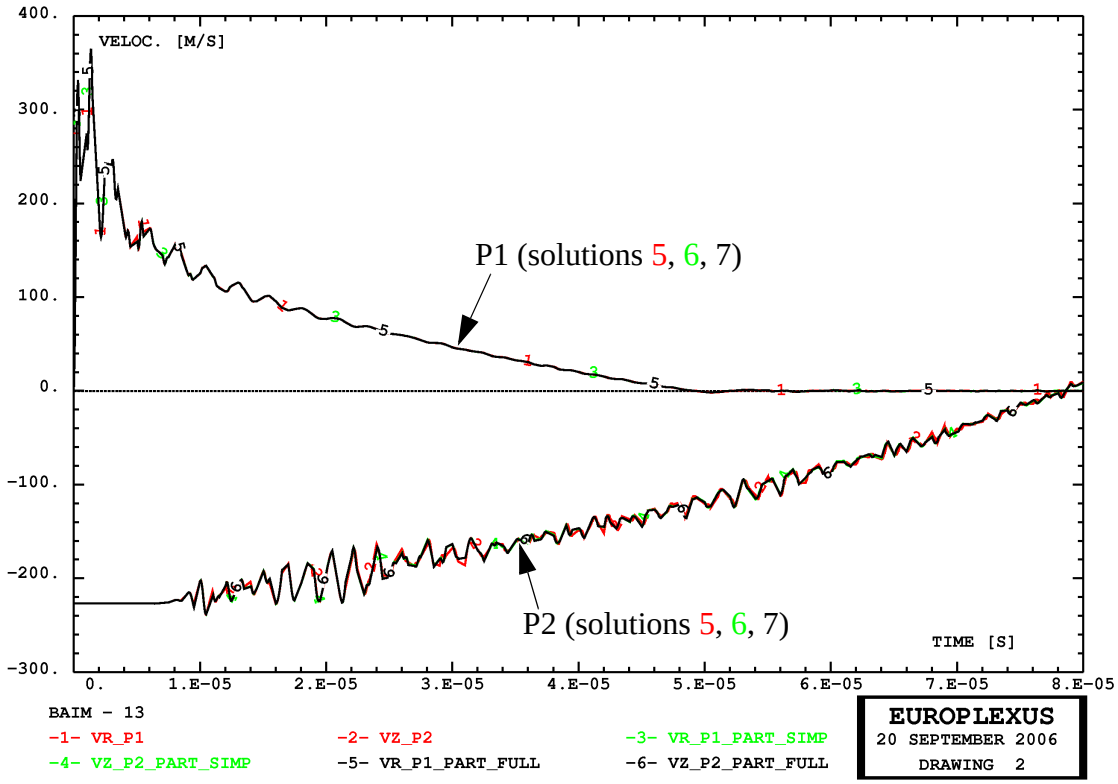


Figure 1 - Taylor bar impact solved by the links method: final yield stress without and with spatial partitioning



a) Comparison of displacements at points P1 and P2



b) Comparison of velocities at points P1 and P2

Figure 2 - Taylor bar impact solved by the links method: relevant displacements and velocities without and with spatial partitioning

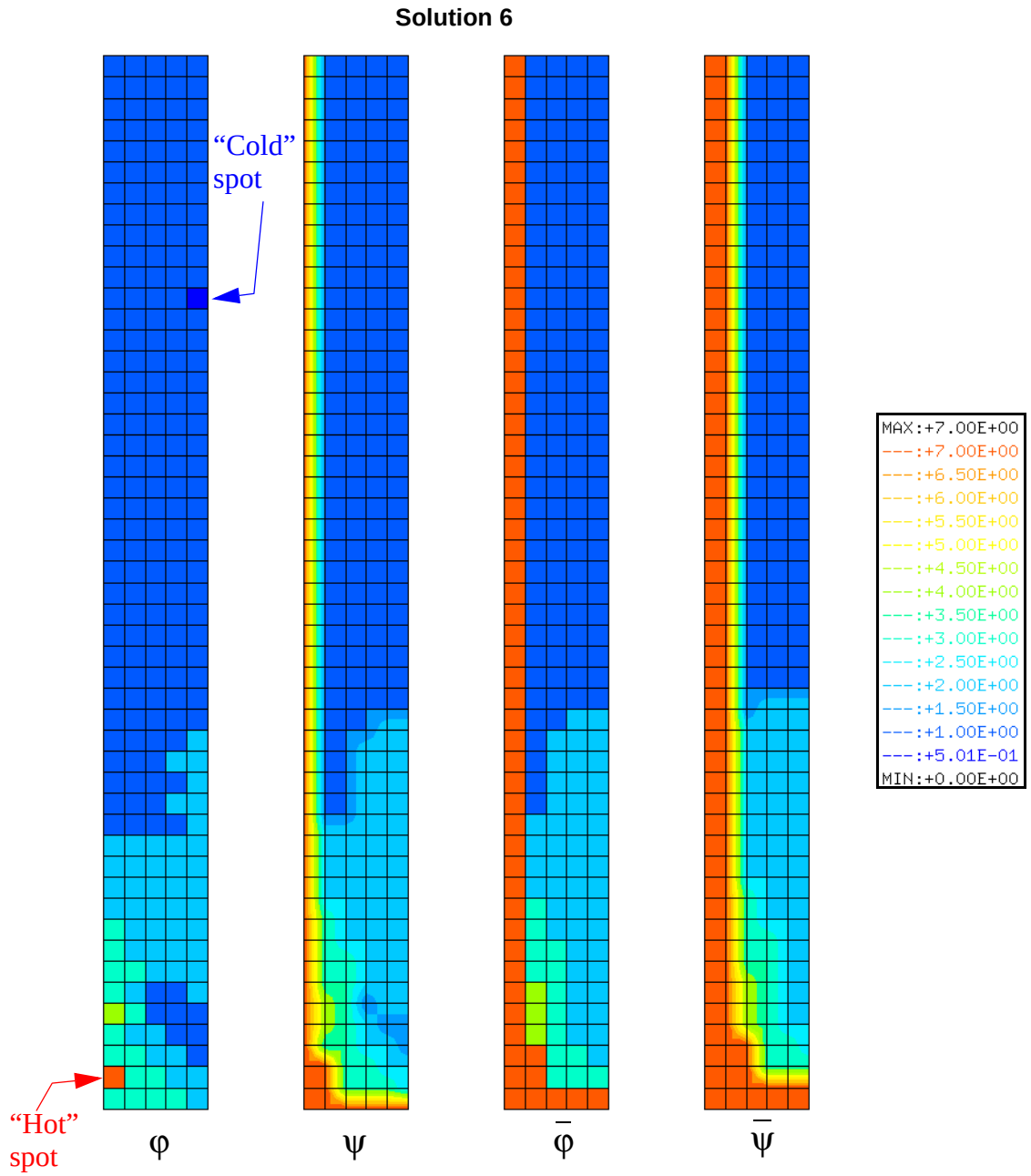


Figure 3 - Taylor bar impact: logarithm in base 2 of partition frequencies at the final time, solution with simplified treatment of links

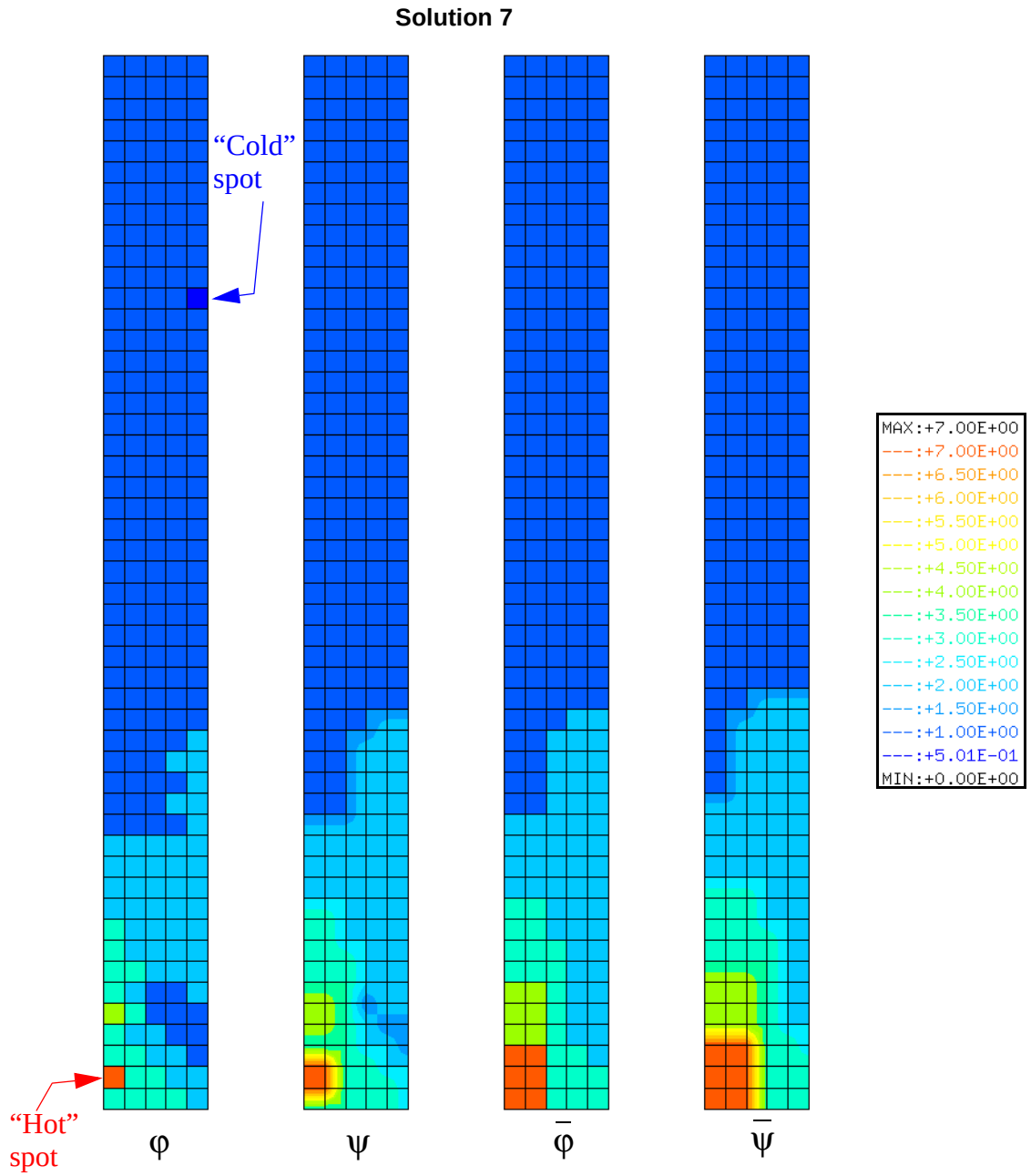


Figure 4 - Taylor bar impact: logarithm in base 2 of partition frequencies at the final time, solution with full treatment of links

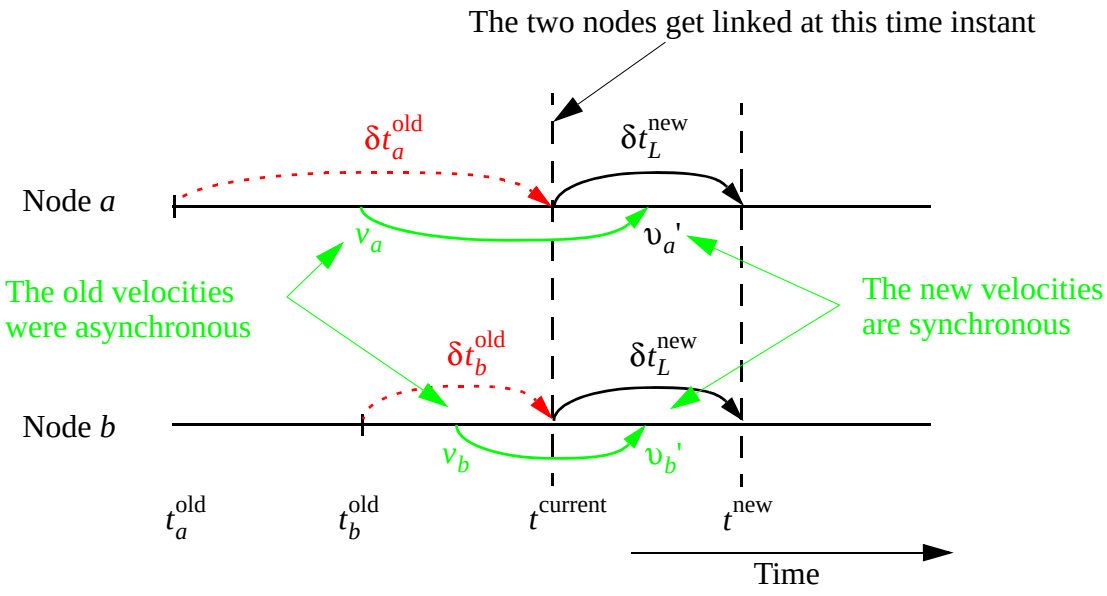
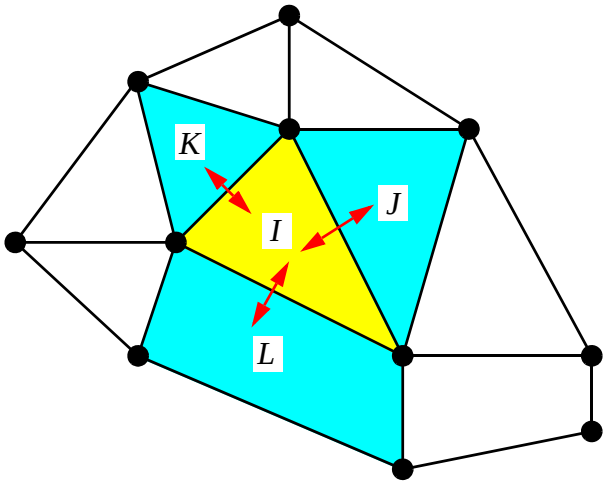
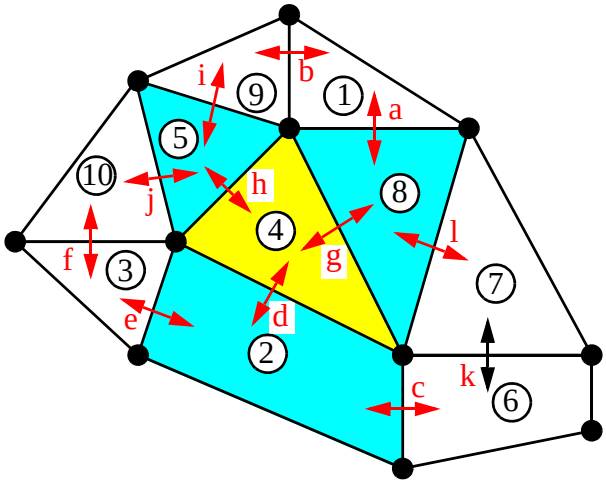


Figure 5 - Lack of synchronization in a non-permanent link

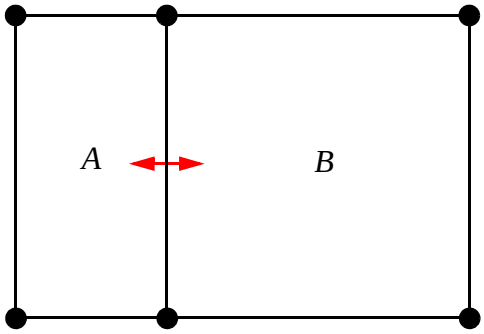


a) Neighboring elements



b) Transport calculations

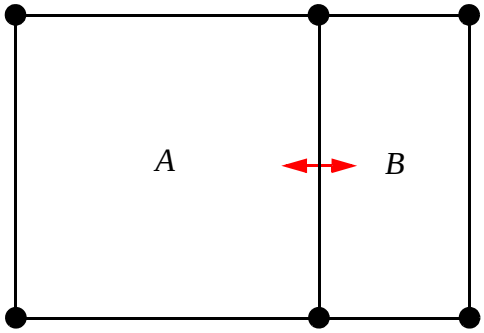
Figure 6 - Mass and energy transport between adjacent fluid finite element



$A < B$

$\Delta t_A = \Delta t_B / 2$

a) Case 1

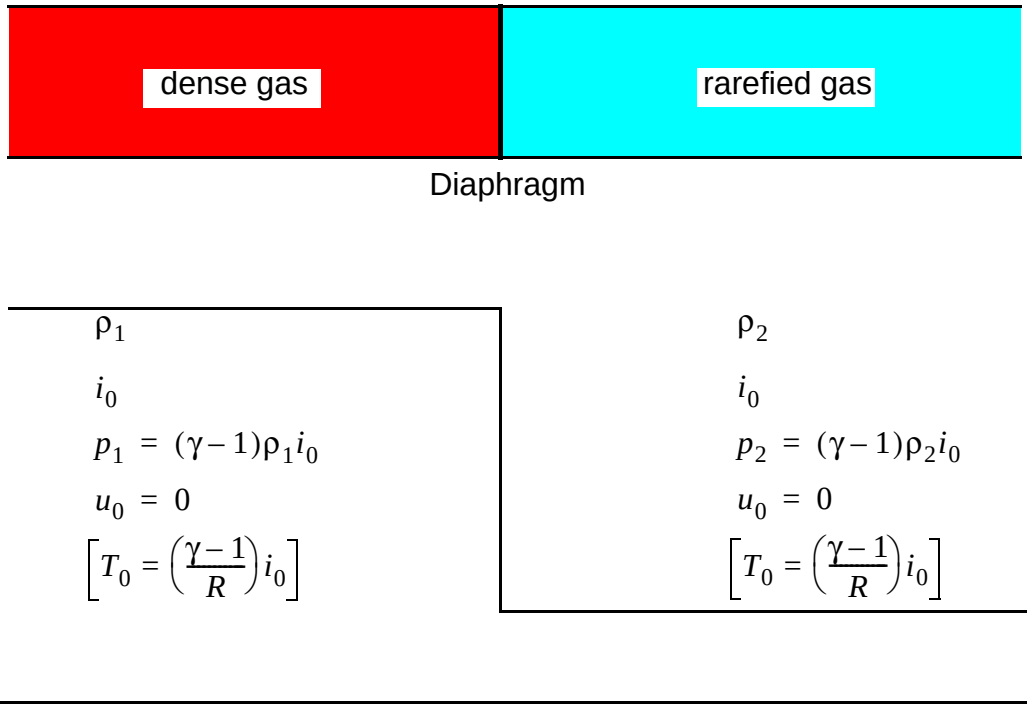


$A < B$

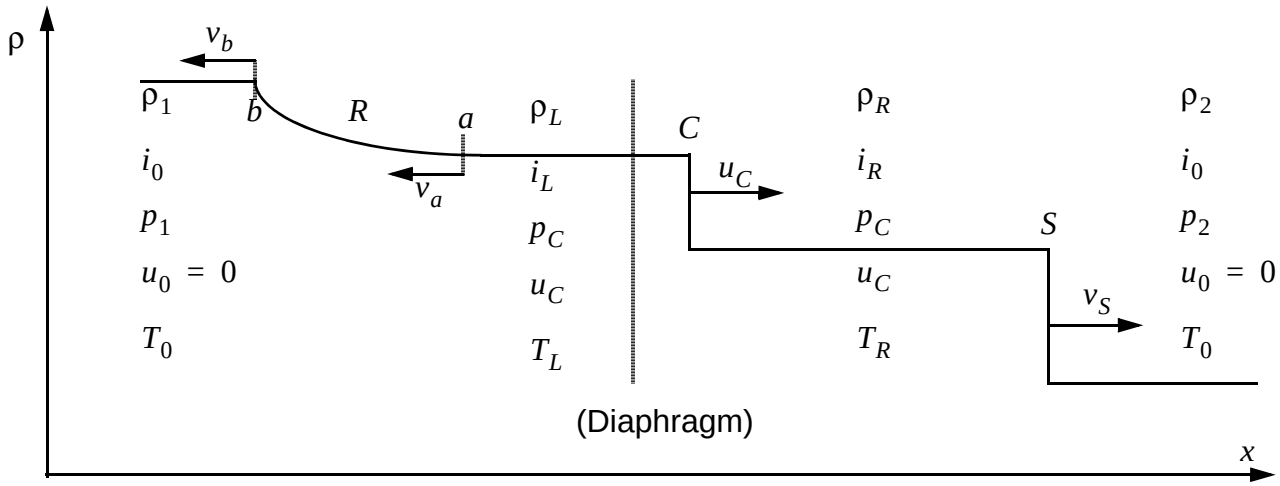
$\Delta t_A = 2\Delta t_B$

b) Case 2

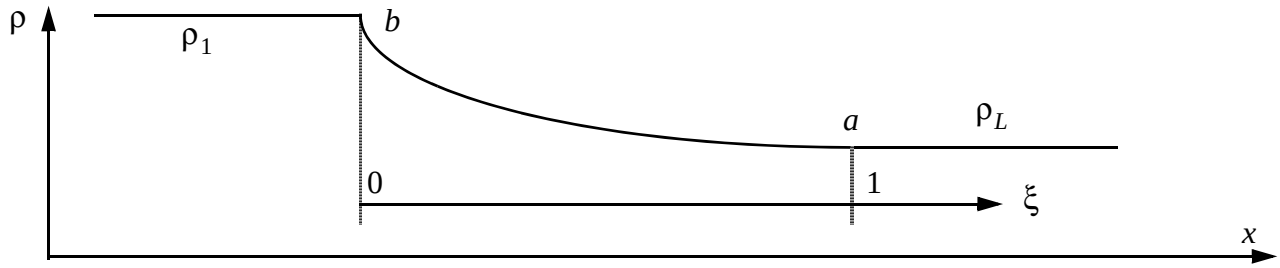
Figure 7 - Mass and energy transport with space partitioning



a) Geometry and initial conditions

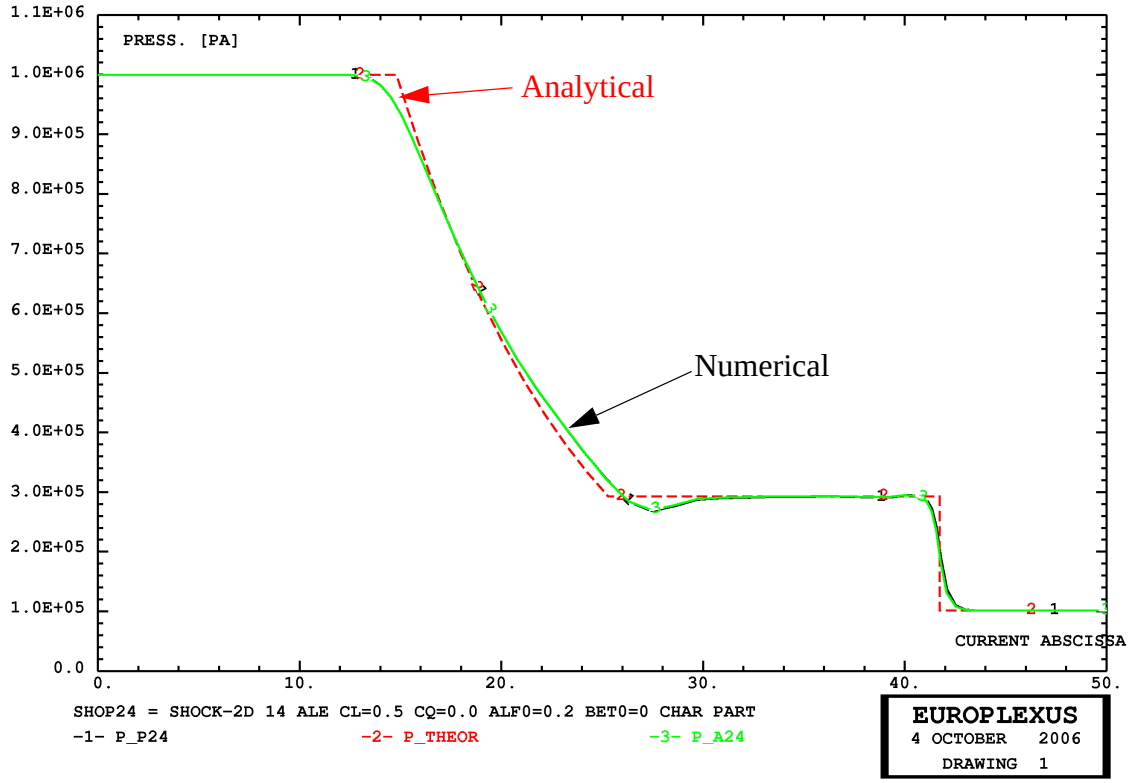


b) Density distribution after removal of the diaphragm

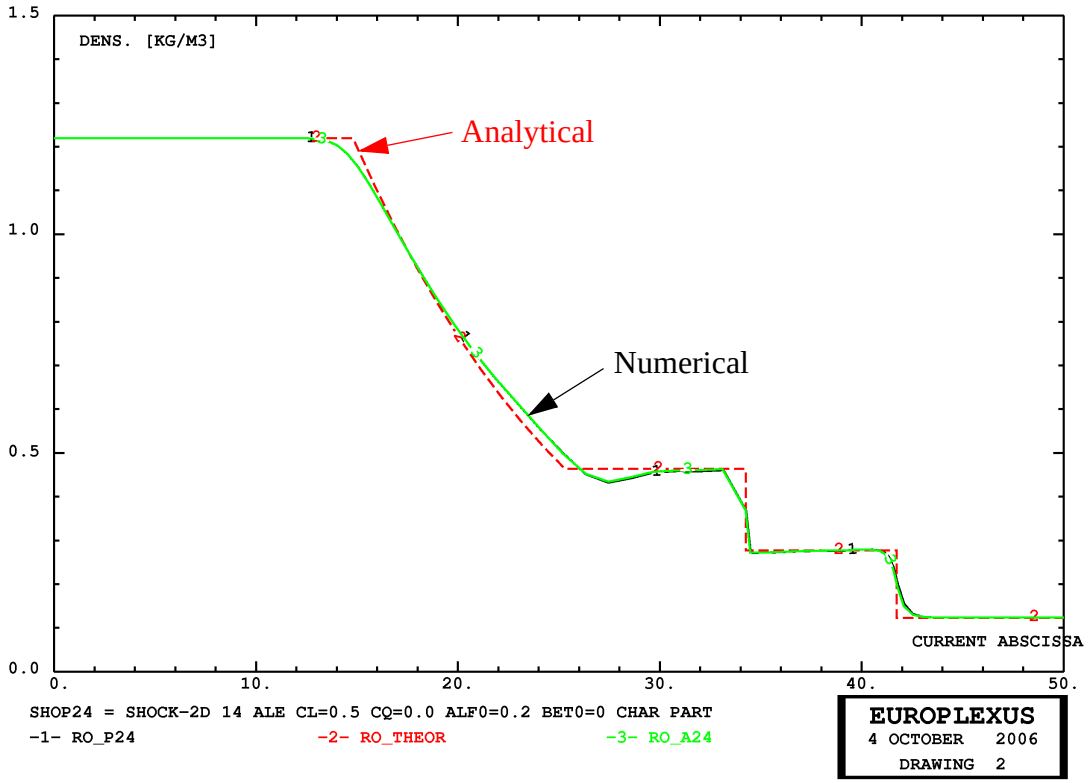


c) Shape of the rarefaction wave region

Figure 8 - Shock tube problem

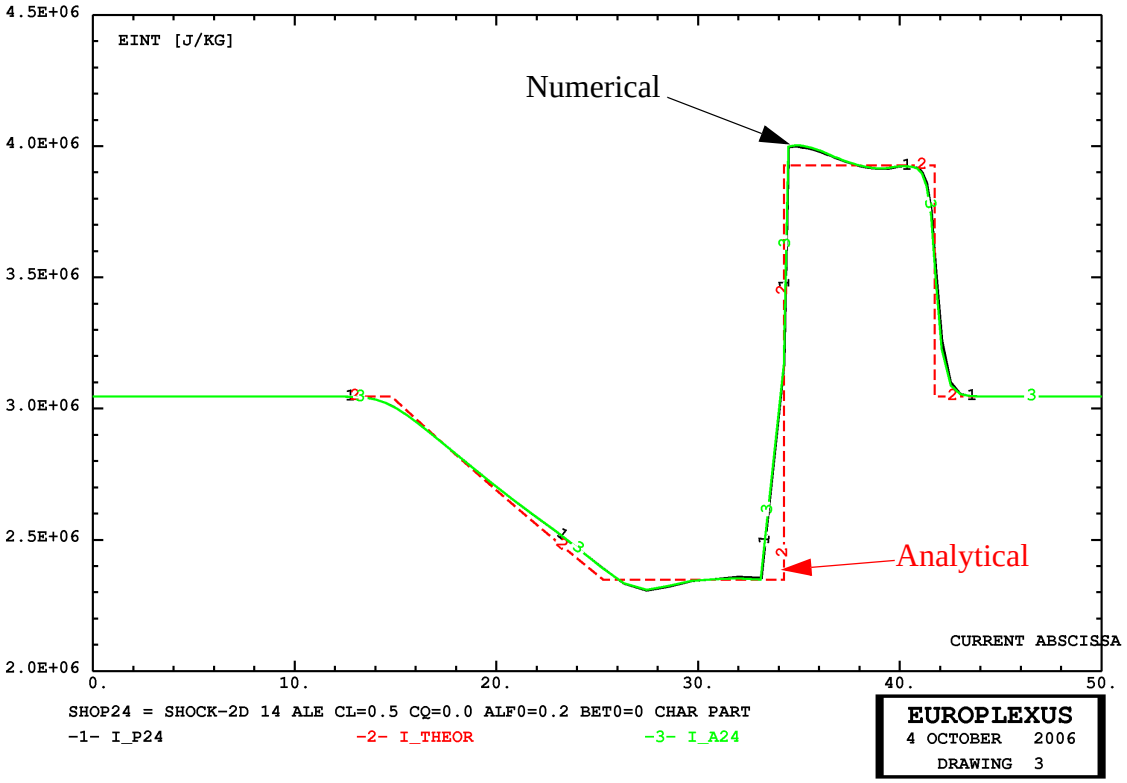


a) Pressure

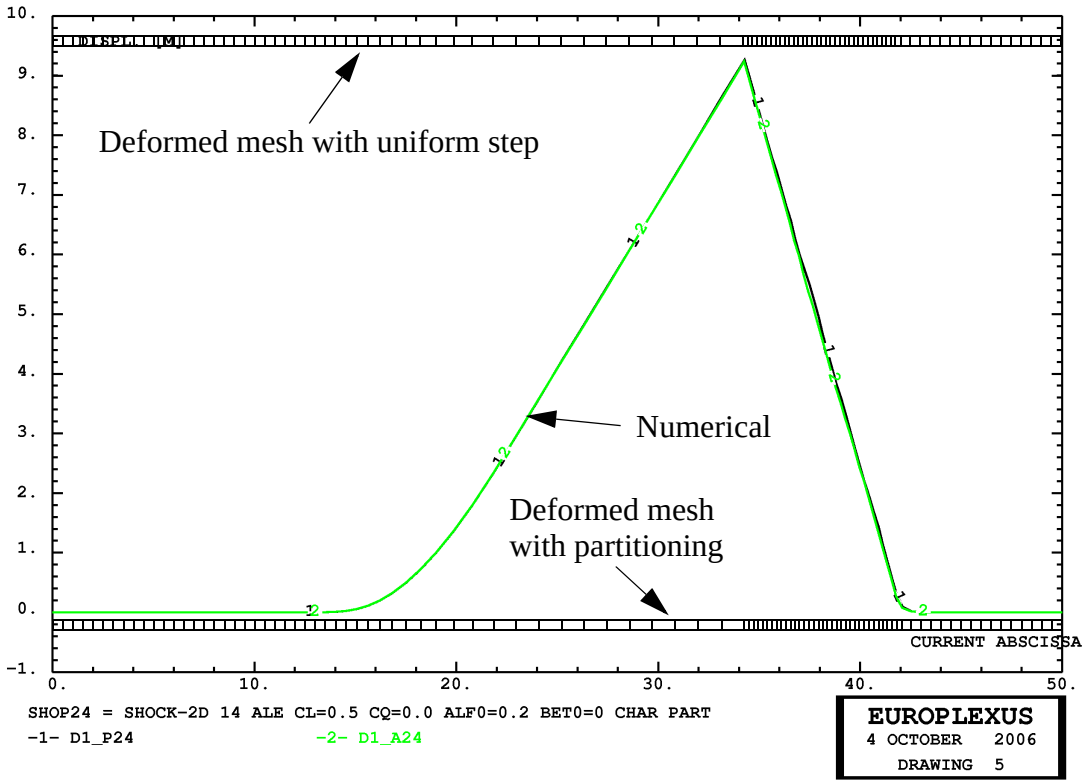


b) Density

Figure 9 - Shock tube problem: relevant fluid pressures and densities without and with spatial partitioning

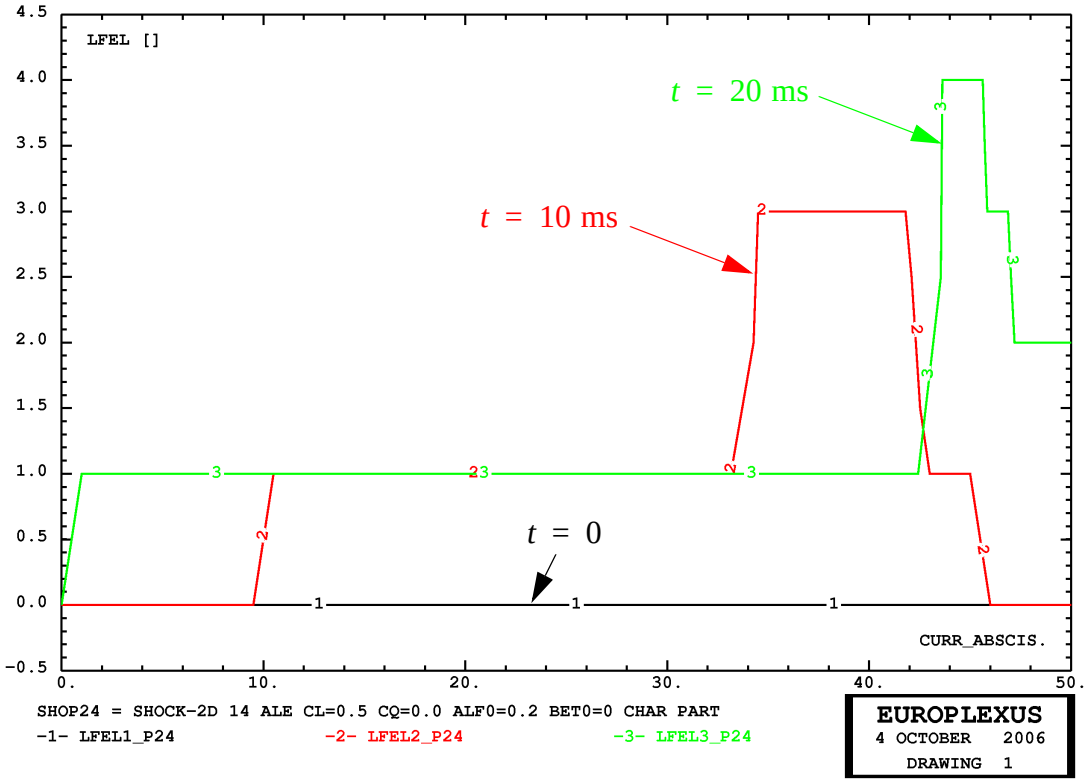


a) Internal energy

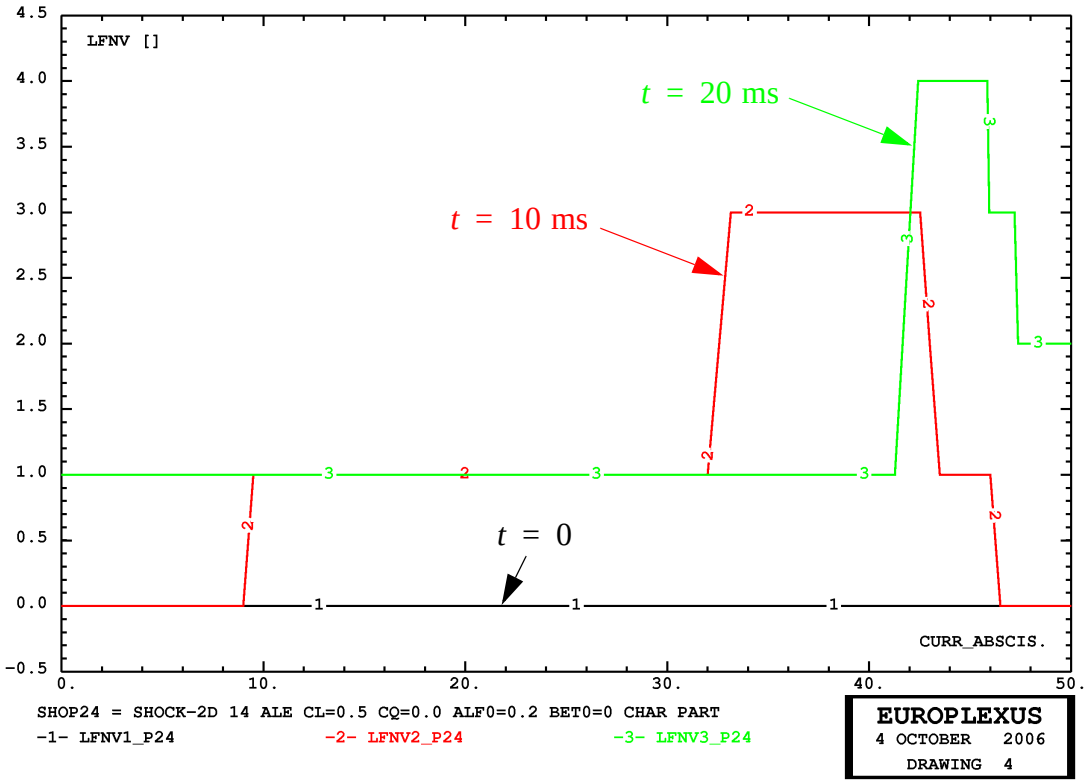


b) Mesh displacement and deformed meshes

Figure 10 - Shock tube problem: relevant internal energy and mesh displacements without and with spatial partitioning

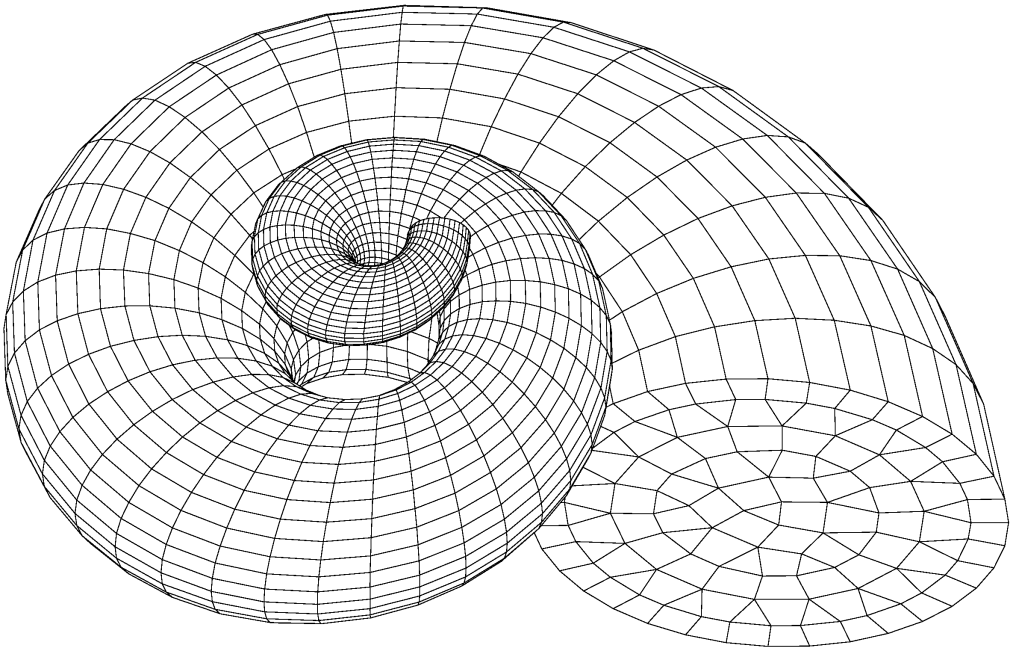


a) Intrinsic element frequency

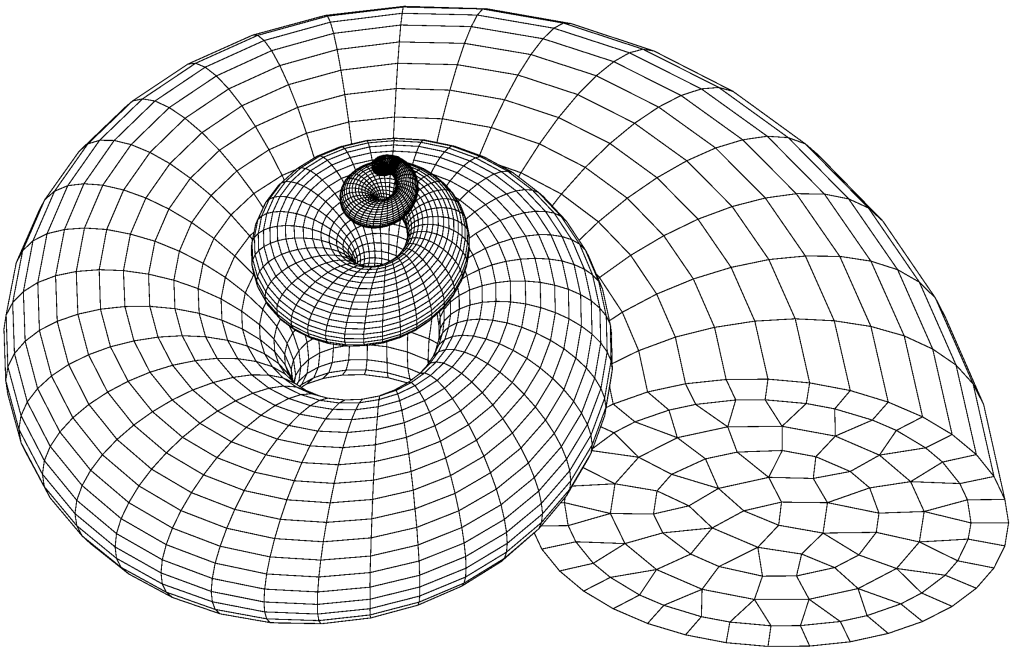


b) Neighboring nodal frequency

Figure 11 - Shock tube problem: logarithm in base 2 of the intrinsic element frequency and of the neighboring nodal frequency in the solution with partitioning



a) Two-whorls cochlea



b) Four-whorls cochlea

Figure 12 - Explosion in an helicoidal tube: initial geometry

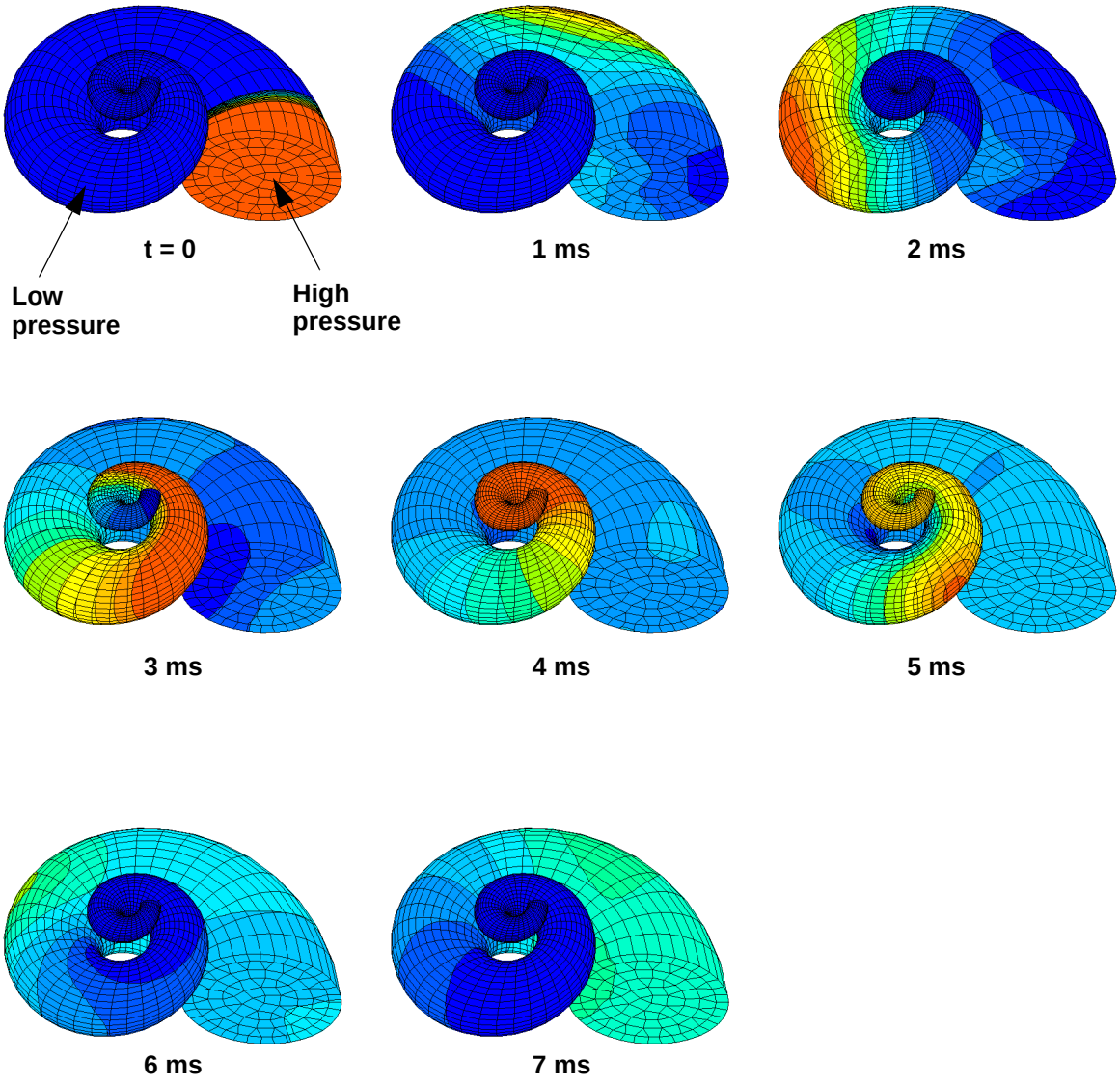
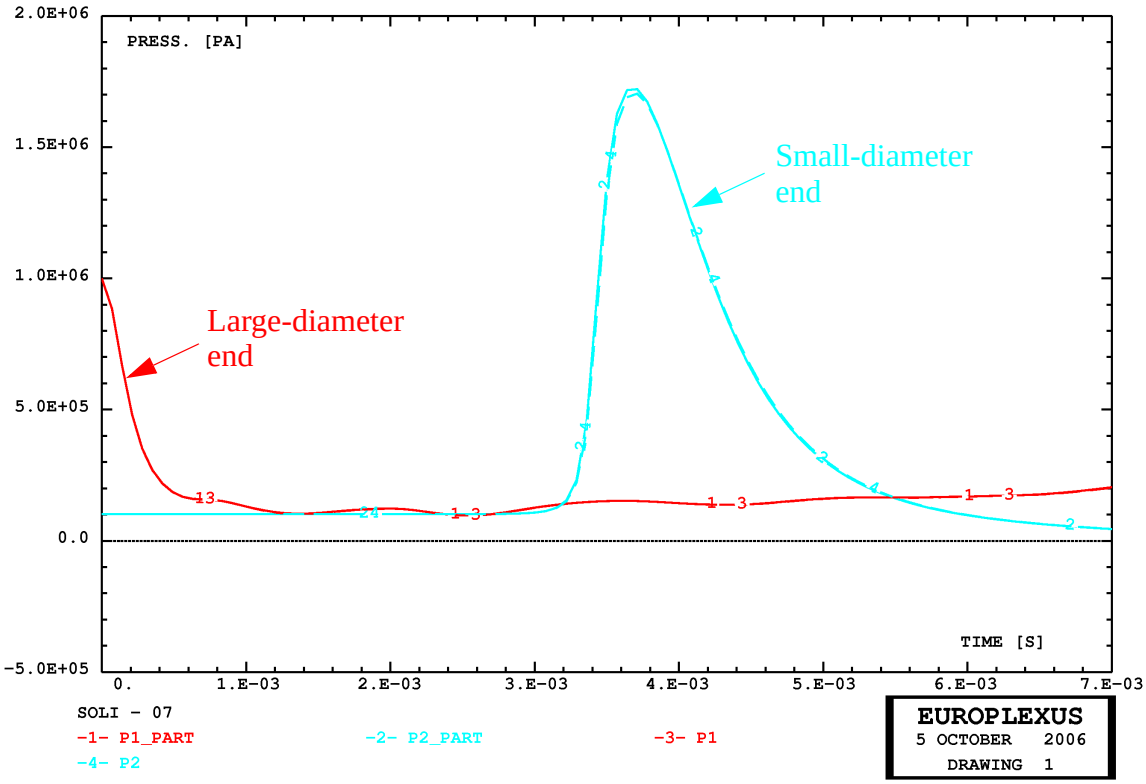
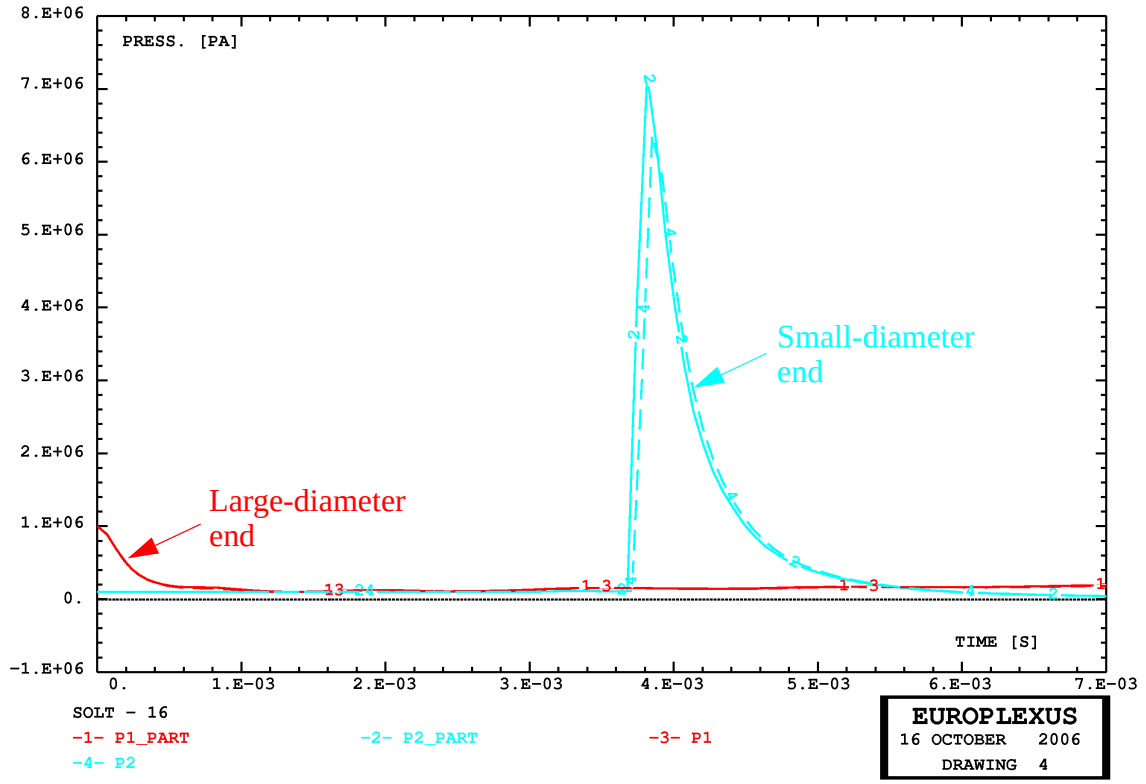


Figure 13 - Explosion in an helicoidal tube: fluid pressure evolution in 2-whorls case with rigid walls

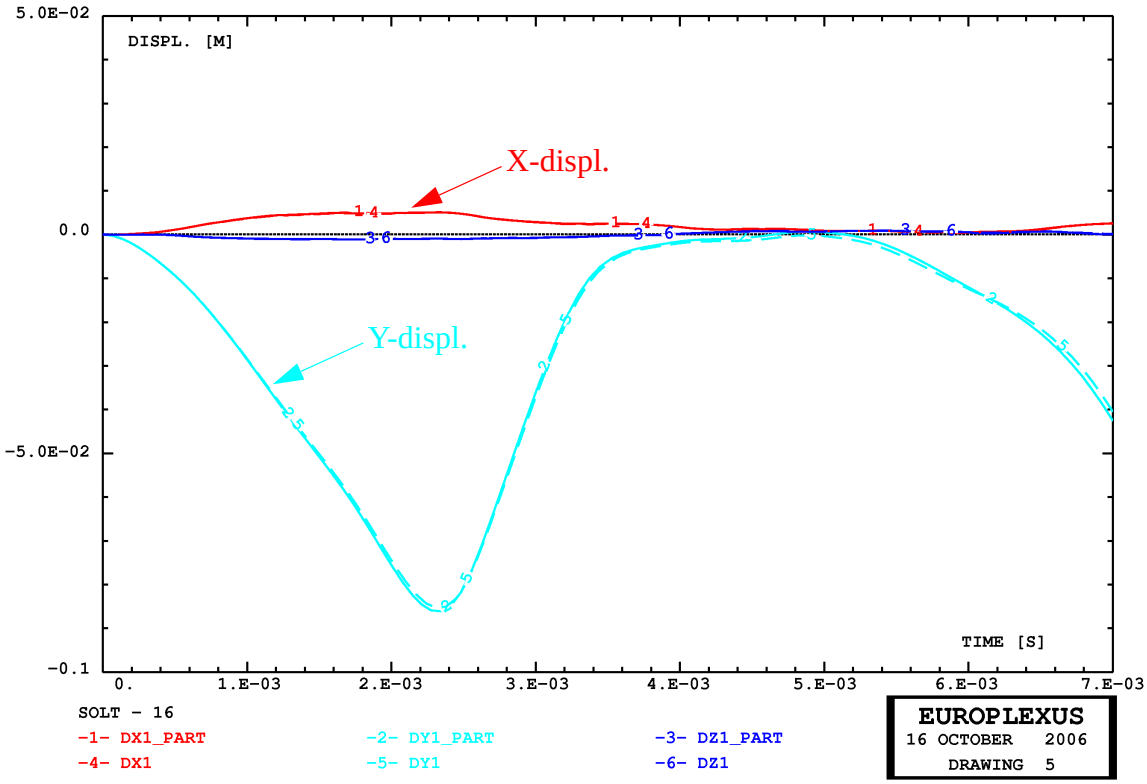


a) two-whorls case with rigid walls

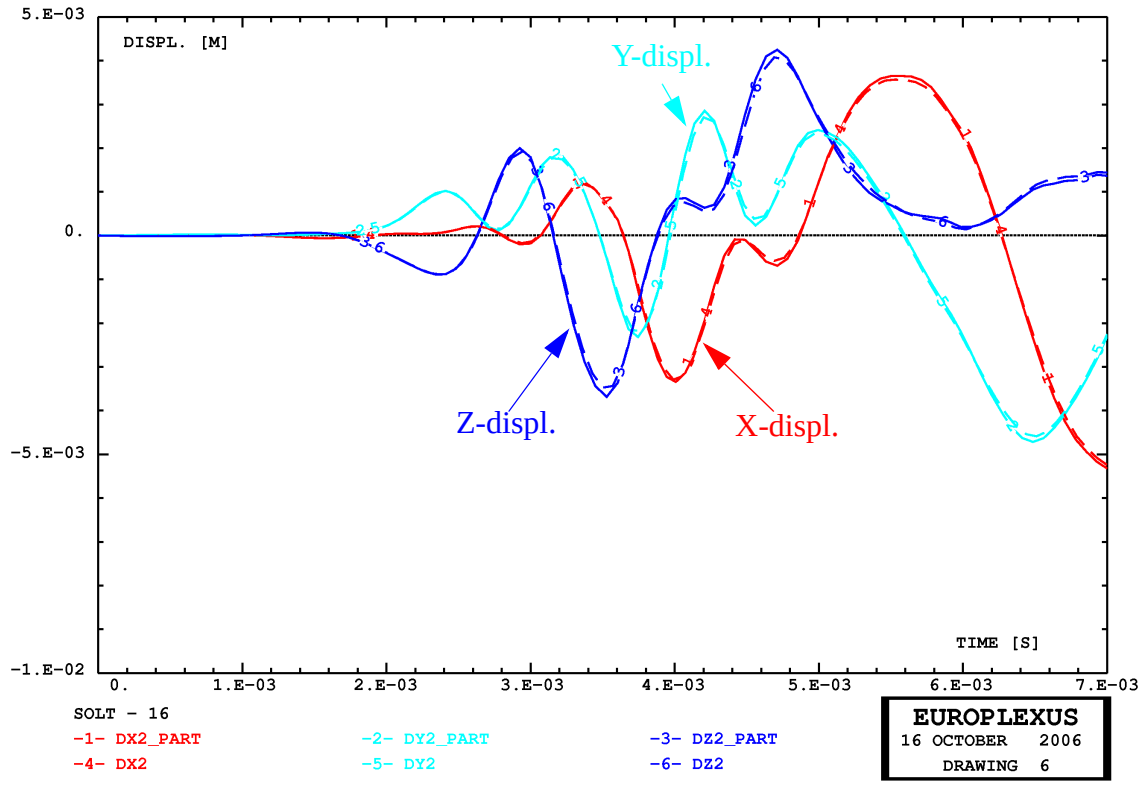


b) four-whorls case with deformable walls

Figure 14 - Explosion in an helicoidal tube: fluid pressures at the two tube ends



a) displacements at large-diameter end



b) displacements at small-diameter end

Figure 15 - Explosion in an helicoidal tube: wall displacements in 4-whorls case with deformable walls

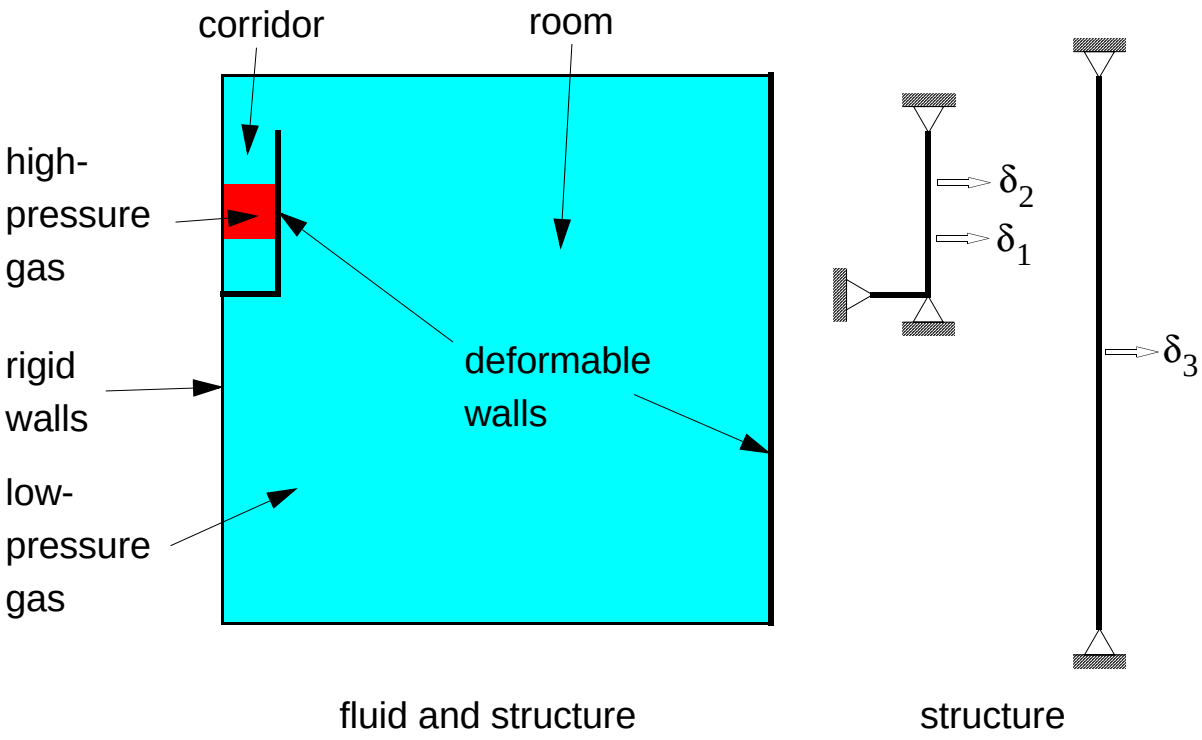
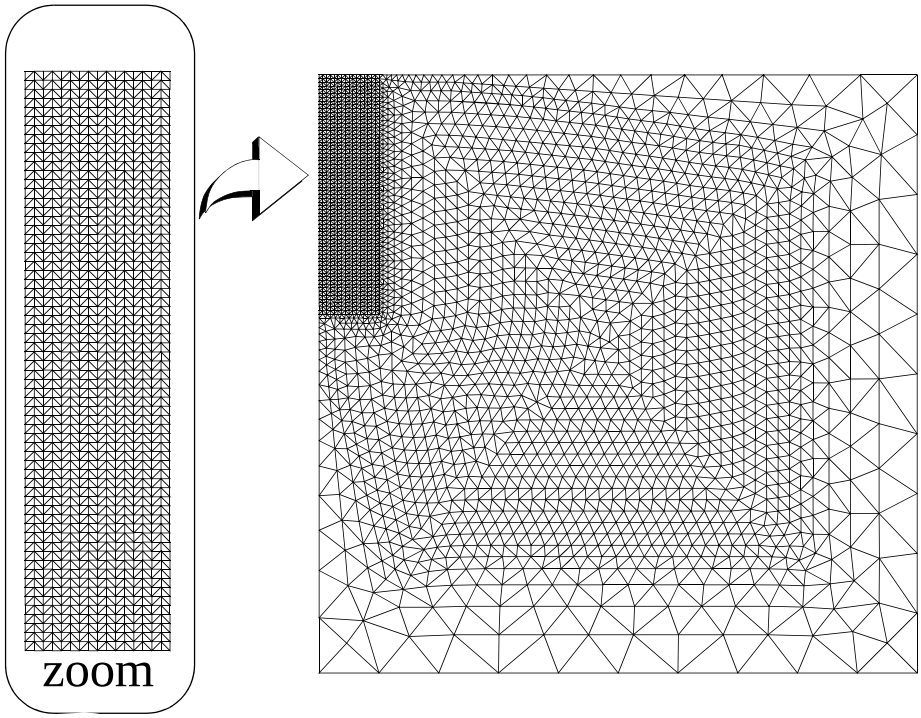
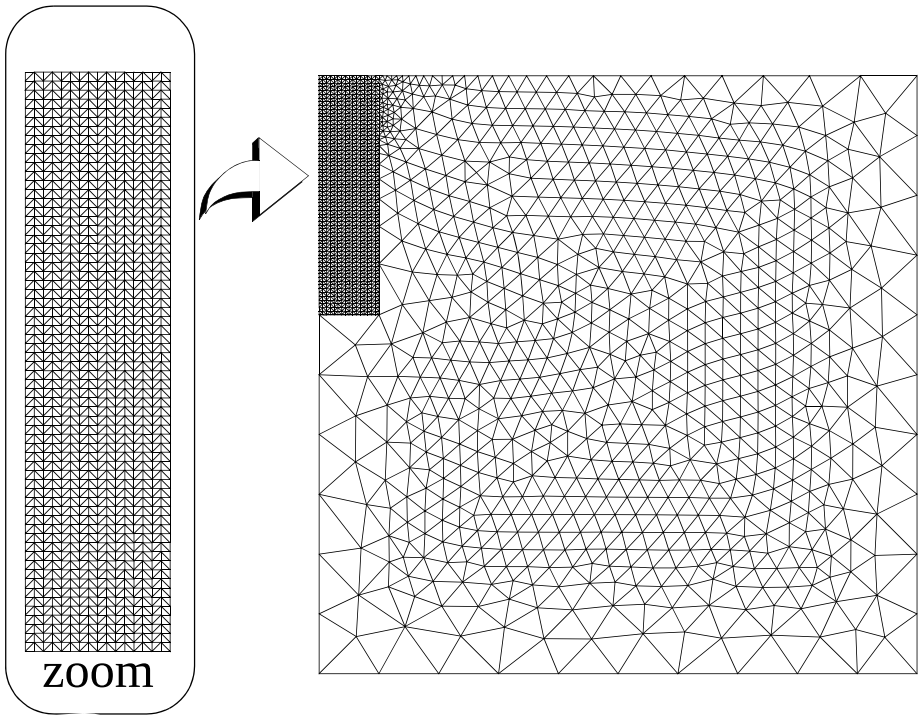


Figure 16 - Explosion in a corridor: geometry

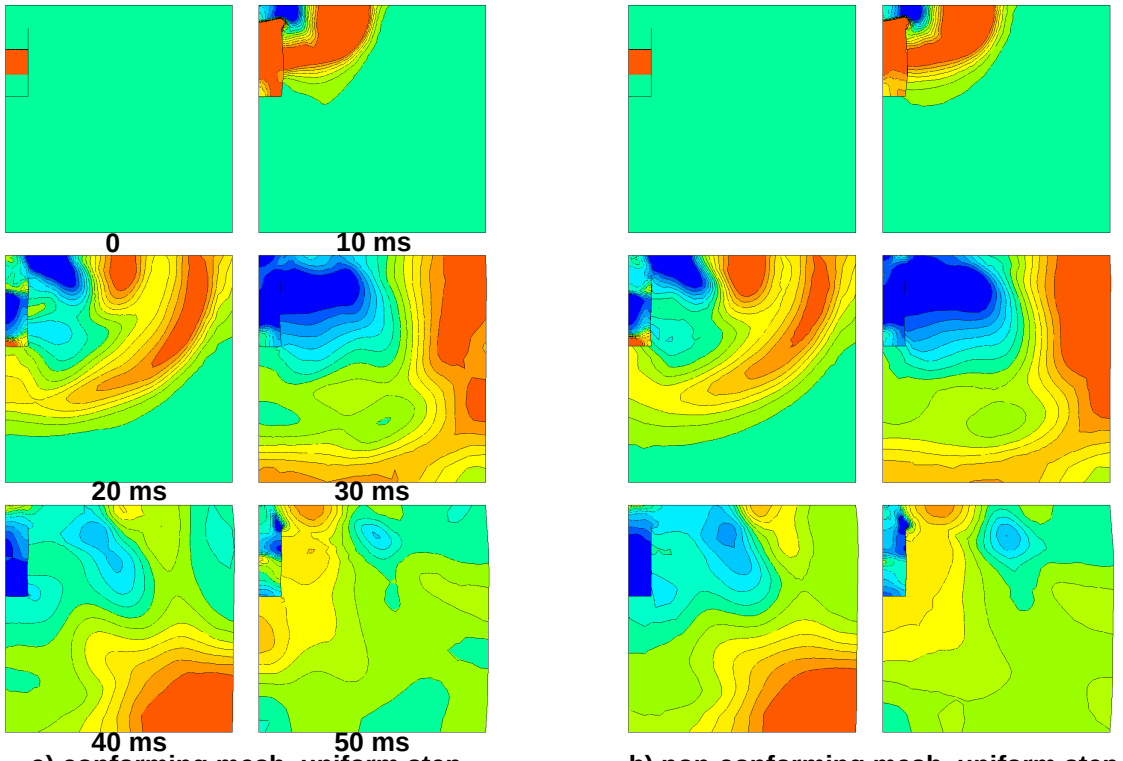


a) conforming mesh



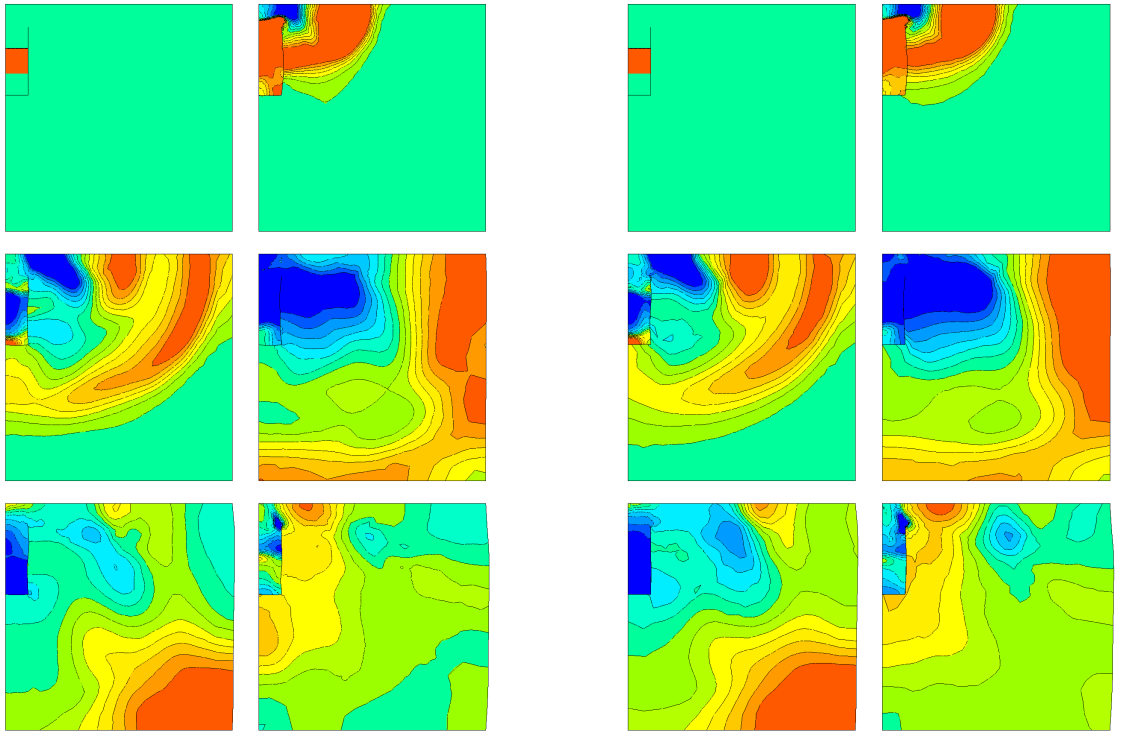
b) non-conforming mesh

Figure 17 - Explosion in a corridor: computational meshes



a) conforming mesh, uniform step

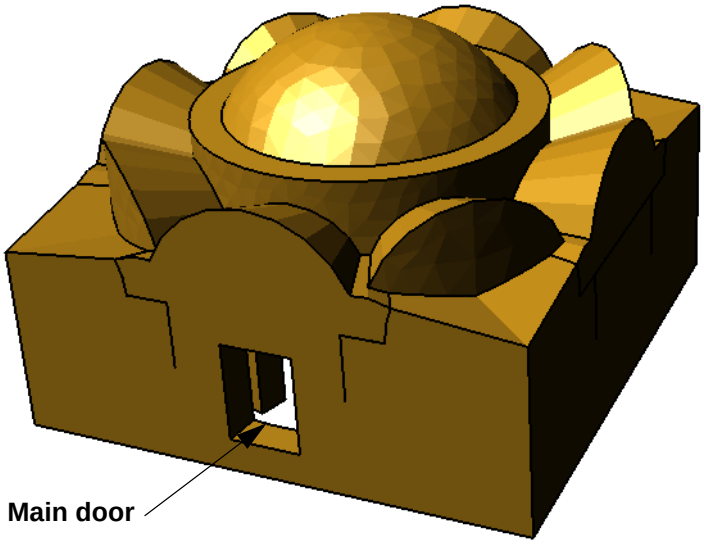
b) non-conforming mesh, uniform step



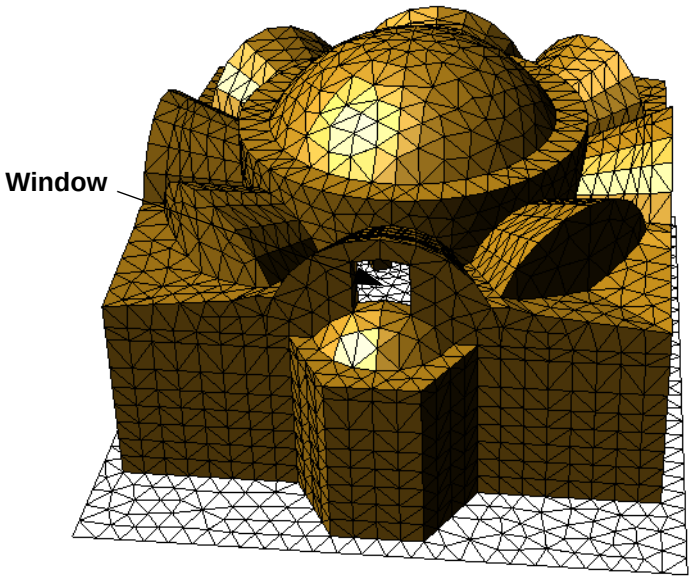
c) conforming mesh, partition

d) non-conforming mesh, partition

Figure 18 - Explosion in a corridor: fluid pressures

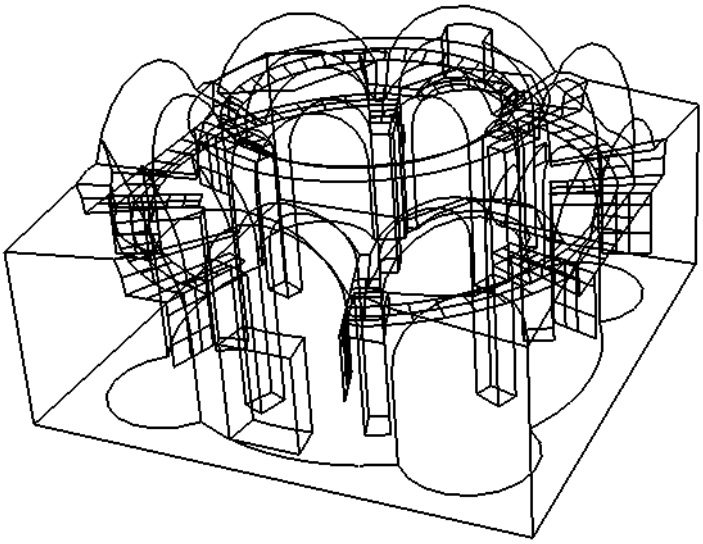


c) Front view

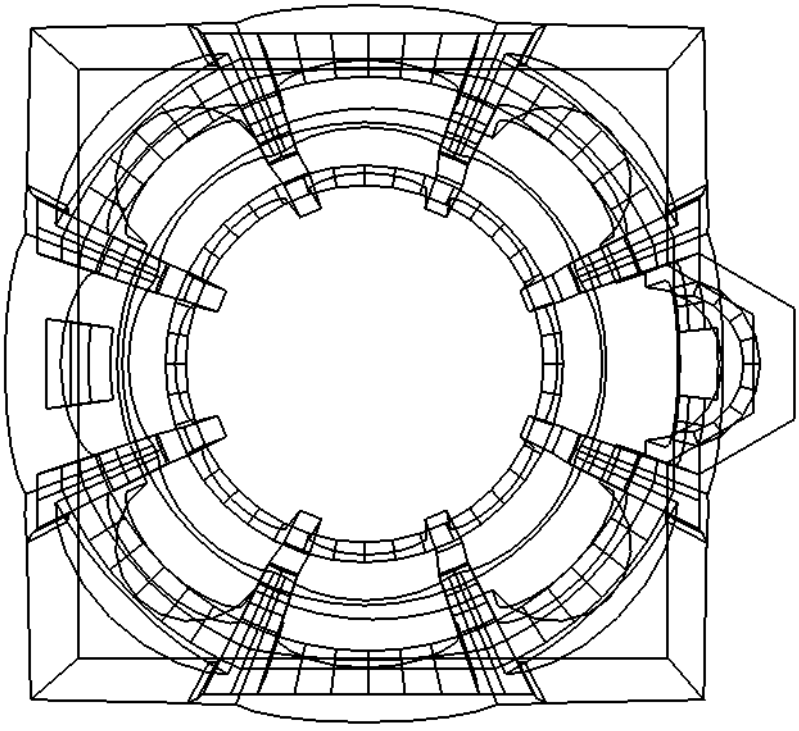


b) Rear view with computational mesh

Figure 19 - Building vulnerability study: geometry

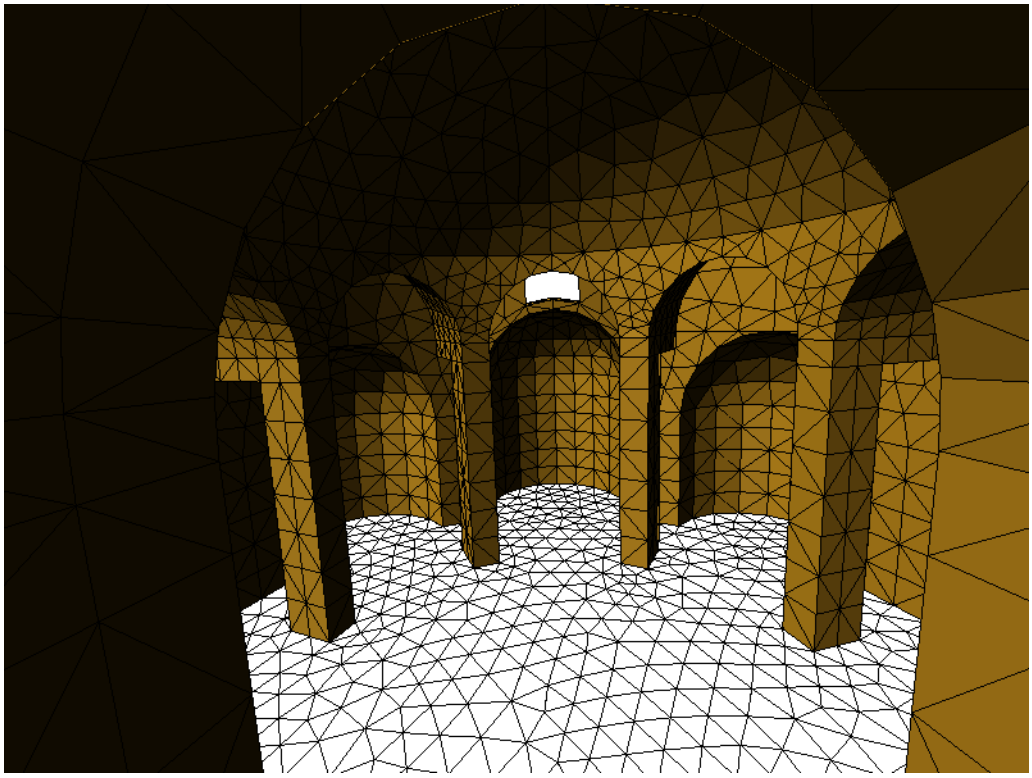


c) Transparency view (from side)

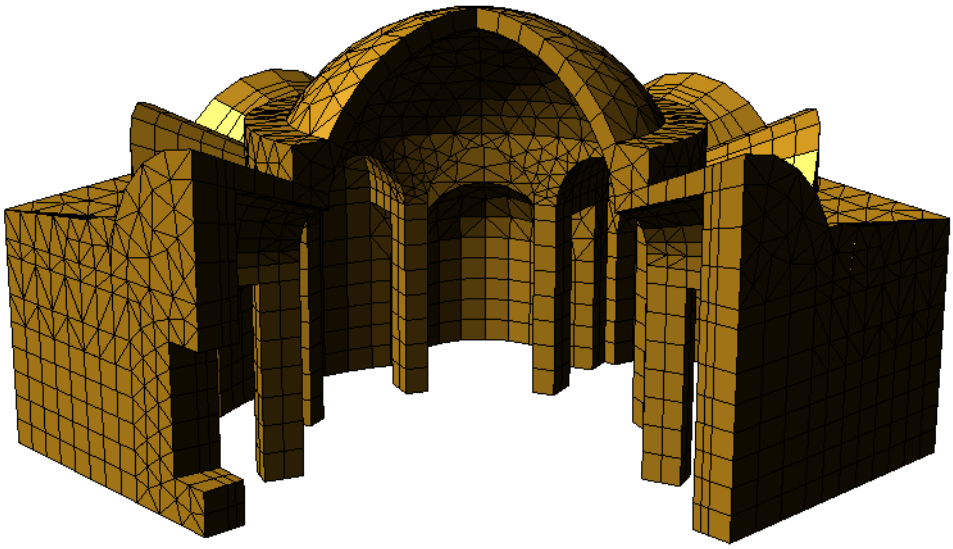


b) Transparency view (from top)

Figure 20 - Building vulnerability study: geometry

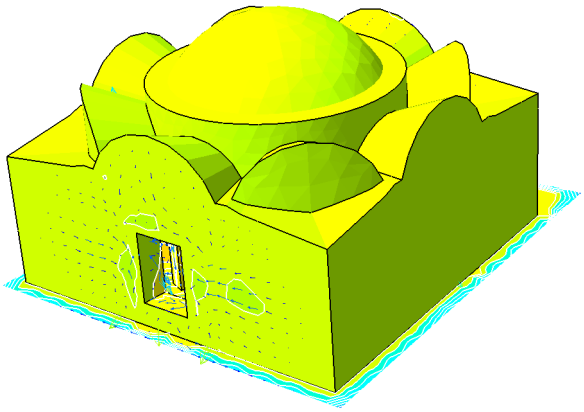


c) Inside view (from main entrance)

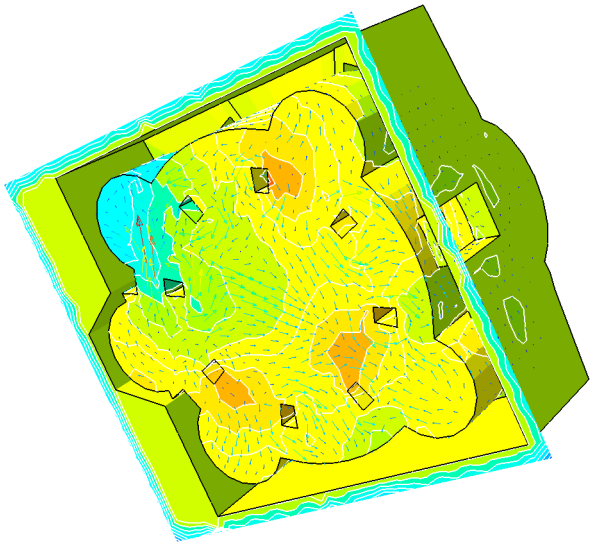


b) Sectioned view

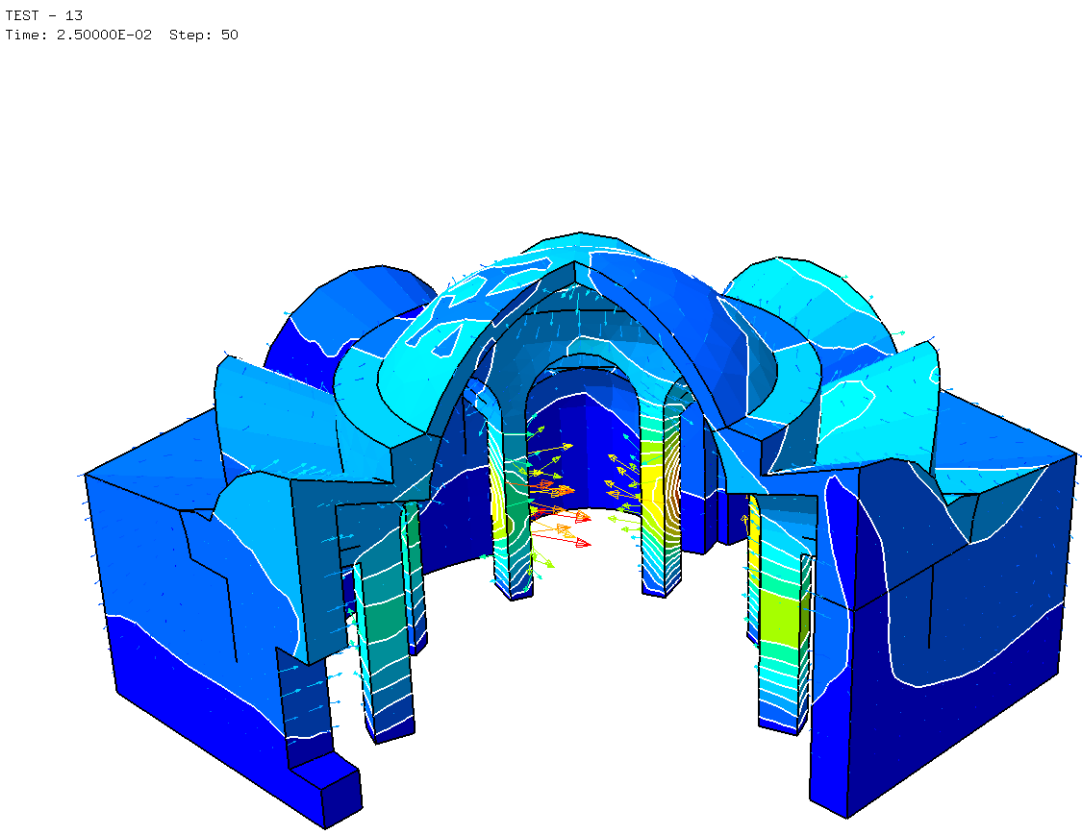
Figure 21 - Building vulnerability study: geometry



a) Outflow from main entrance at 25 ms



b) Fluid flow seen from below at 25 ms



c) Structural displacement and velocities at 25 ms (cross-section view)

Figure 22 - Building vulnerability study: some simulation results

European Commission

EUR 23062 EN – Joint Research Centre – Institute for the Protection and Security of the Citizen
Title: Spatial Time Step Partitioning in Explicit Fast Transient Dynamics
Author(s): F. Casadei, J.P. Halleux
Luxembourg: Office for Official Publications of the European Communities
2008 – 145 pp. – 21.0 x 29.7 cm
EUR – Scientific and Technical Research series – ISSN 1018-5593

Abstract
This report presents a technique for spatial partitioning of the time increment in the explicit central-difference time integration scheme commonly used for finite-element modeling of fast transient dynamic phenomena. The time increment varies not only in time, as is usual to account for mesh distortion and evolution of material properties, but also in space—at the finite element level—following local stability limitations rather than global ones.

This may lead to substantial savings of computer time whenever the material properties that govern wave propagation speed and/or the mesh size are largely non-uniform in the numerical model, as is typical of many large industrial applications, especially in 3D, and even more so in the presence of fluid-structure interactions. The proposed partitioning algorithm, which is completely automatic and does not require any specific input data, may be applied in principle to all types of elements and material models. As shown by several numerical examples, it preserves the outstanding numerical properties—i.e. the renowned accuracy and robustness—of the classical uniform-step explicit time integration scheme, of which it may be considered a powerful generalization.

Once fully implemented and validated in a general explicit computer code, the present technique has the potential for freeing the engineer from the main limitation of explicit analysis, usually related through stability requirements to the size of the smallest element. In fact, the computational mesh may be locally refined virtually at will without the usual prohibitive effects on computational costs. This might open the way to applications which are simply out of reach with the classical algorithms.

The present report is subdivided into two Parts. Part I introduces the basic spatial partitioning technique within a Lagrangian formulation, with some simple academic examples. Part II presents the treatment of boundary conditions, the extension to fluids via an Arbitrary Lagrangian Eulerian formulation and some more realistic applications.

The mission of the JRC is to provide customer-driven scientific and technical support for the conception, development, implementation and monitoring of EU policies. As a service of the European Commission, the JRC functions as a reference centre of science and technology for the Union. Close to the policy-making process, it serves the common interest of the Member States, while being independent of special interests, whether private or national.

JRC
EUROPEAN COMMISSION

Publications Office
Publications.eu.int