ISSN 1831-9424

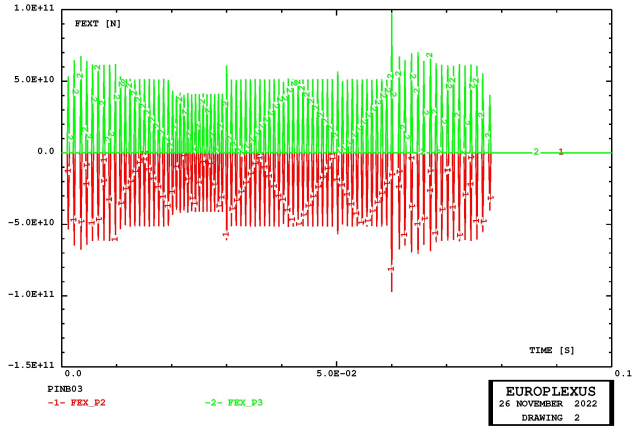
European Commission

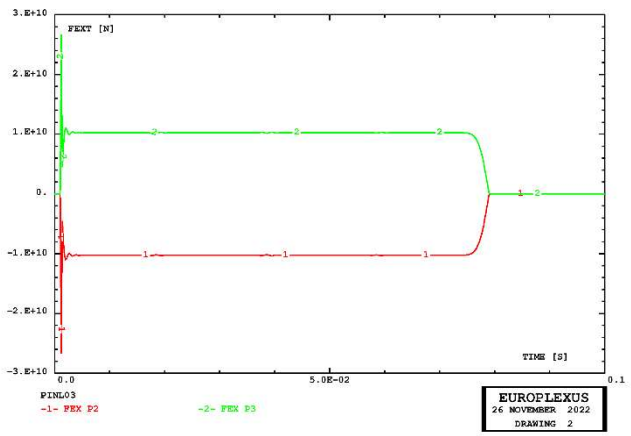
JRC Technical Report

On the parametrization of velocity-based constraints in EUROPLEXUS

Casadei, F., Markovic, D., Larcher M.

2023





Joint Research Centre

EUR 31504 EN

Contact information
Name: Martin Larcher
Address: JRC Ispra
Email: martin.larcher@ec.europa.eu

EU Science Hub
https://joint-research-centre.ec.europa.eu

JRC133012

EUR 31504 EN

PDF ISBN 978-92-68-03227-5 ISSN 1831-9424 doi:10.2760/71283 KJ-NA-31-504-EN-N

Luxembourg: Publications Office of the European Union, 2023

How to cite this report: Casadei, F., Markovic, D., Larcher M., *On the parametrization of velocity-based constraints in EUROPLEXUS*, Publications Office of the European Union, Luxembourg, 2023, doi:10.2760/71283, JRC133012.

On the parametrization of velocity-based constraints in EUROPLEXUS

F. Casadei[1,2], D. Markovic[2], and M. Larcher[2]

[1] *Active Senior*
[2] *European Commission, Joint Research Centre*
Directorate for Space, Security and Migration
Safety and Security of Buildings Unit

April 18, 2023

Contents

List of Tables

List of Figures

Foreword

This report is part of a large series of scientific-technical documents that are meant to provide essential information and documentation to users and developers of the EUROPLEXUS code. EUROPLEXUS (also abbreviated as EPX) is a computer code jointly developed by the French Commissariat à l'Energie Atomique (CEA DMT Saclay) and by EC-Joint Research Centre (JRC Ispra) within the framework of contractual agreements between the two research bodies.

EPX is a mature, general-purpose Finite Element and Finite Volume explicit code under active development since 1999, for the simulation of fast transient dynamic events in complex fluid-structure systems. It is an evolution of its ancestor PLEXIS-3C, which was also jointly developed by CEA and JRC in the 1980s and early '90s.

The code has been traditionally used in safety studies, ranging from nuclear reactors, to energy plants, to chemical and industrial plants, off-shore structures, car and road barrier crashes, among others. More recently it has proven a very useful tool in providing certified and independent numerical solutions in support of EC policies regarding the security of critical infrastructures and public spaces (like buildings, train and metro stations and carriages, etc.), which may be vulnerable to terrorist attacks or to natural disasters.

While being mainly of technical nature, the information contained in this series of reports is an invaluable source of reference for the users (as a complement to the User's manual) but also in particular for the developers of EPX. New models made available in the code are described in detail from the theoretical viewpoint. Several verification and application examples are also usually provided, in order to illustrate the practical use and to verify the correct functioning of the models.

Usually, at the end of each report an Appendix lists the input files that were used to produce the examples presented in the report. This allows users to re-run the test cases with EPX at any time and to use them as a basis for their own numerical simulations.

A complete list of the reports (produced both at JRC and at CEA) in this series can be found in the Bibliography section of the EPX User's manual [1].

Abstract

This report presents some notes on the treatment of constraints by Lagrange Multipliers in EUROPLEXUS [1] (also abbreviated as EPX), a computer code jointly developed by the French Commissariat à l'Energie Atomique (CEA DMT Saclay) and by EC-JRC. The code application domain is the numerical simulation of fast transient phenomena such as explosions, crashes and impacts in complex three-dimensional fluid-structure systems.

Recently, EPX is being used to simulate vehicle crash against obstacles such as road barriers for safety and security studies. These studies involve the treatment of complex contact-impact scenarios, which in EPX are typically modelled by the method of Lagrange Multipliers (LM), although other methods (e.g. penalty-based) are also available in the code.

Recent investigations concerning (elastic) impact tests have shown that the LM-based mid-step velocity constraints strategy (the default one in EPX) leads to smooth solutions (few oscillations), but some energy is lost at the moment of the impact, namely each time some previously free nodes get into contact, so that the total energy of the system is not exactly conserved.

On the contrary, the full-step velocity constraints strategy produces a lot of oscillations (if the material is elastic) but it conserves much better the energy of the system.

Therefore, it has been proposed to implement a *parametrization* of the velocity constraints, allowing to choose an intermediate value of the time at which the constraints are imposed. The goal is to reduce the spurious energy dissipation observed with the mid-step strategy, but hopefully without triggering the large oscillations produced by the full-step strategy.

Keywords: *Constraints, Lagrange Multipliers, Contact-Impact.*

1 Introduction

This report presents some notes on the treatment of constraints by Lagrange Multipliers in EUROPLEXUS.

EUROPLEXUS [1] (also abbreviated as EPX) is a computer code jointly developed by the French Commissariat à l'Energie Atomique (CEA DMT Saclay) and by EC-JRC. The code application domain is the numerical simulation of fast transient phenomena such as explosions, crashes and impacts in complex three-dimensional fluid-structure systems. The Cast3m [2] software from CEA is used as a pre-processor to EPX when it is necessary to generate complex meshes.

Recently, EPX is being used to simulate vehicle crash against obstacles such as road barriers for safety and security studies. The code has proven a very useful tool in providing certified and independent numerical solutions in support of EC policies regarding the security of critical infrastructures and public spaces (like buildings, train and metro stations and carriages, etc.), which may be vulnerable to terrorist attacks or to natural disasters. In this framework, the capabilities of EPX to treat complex contact-impact scenarios in fast transient dynamics are being continuously tested and verified against available experimental data and other computer codes.

The presented work has been conducted in order to improve the performance of the EPX software for the simulation of vehicle crashes. Such simulations are needed for the assessment of security barriers used for the protection of public spaces against hostile vehicle ramming loads. The JRC is also providing generic vehicle models, publicly shared under the EU Public Licence, to foster standardisation in the field of protective barriers [3, 4].

Recent investigations concerning (elastic) impact tests have shown that the LM-based mid-step velocity constraints strategy (the default one in EPX) leads to smooth solutions (few oscillations), but some energy is lost at the moment of the impact, namely each time some previously free nodes get into contact, so that the total energy of the system is not exactly conserved.

Essential boundary conditions (constraints) can be imposed in EPX by a variety of methods. The most important and traditionally more often used of these methods in EPX is the method of Lagrange Multipliers (LM). The method leads to an exact solution of constraints under the form of a linear system of algebraic equations. This introduces an implicit part in the otherwise fully explicit architecture of the code. However, the implicit part involves only a (usually small) subset of the model's degrees of freedom (dofs), and therefore it typically does not penalize too much the efficiency of the numerical solution.

An advantage of the LM method over other methods also available in EPX, such as the penalty method, is that all constraints are satisfied simultaneously and a unique solution is obtained, which does not depend upon any heuristic parameters (which are typical, e.g., of penalty-based methods).

On the contrary, the full-step velocity constraints strategy produces a lot of oscillations (if the material is elastic) but it conserves much better the energy of the system.

Therefore, it has been proposed to implement a *parametrization* of the velocity constraints, allowing to choose an intermediate value of the time at which the constraints are imposed. The goal is that of reducing the spurious energy dissipation observed with the mid-step strategy but hopefully without triggering the large oscillations produced by the full-step strategy.

The report is organized as follows:

- Section 2 starts by briefly presenting the governing equations of transient dynamics, the time integration scheme and the LM method to impose constraints. Then, the theoretical background behind the parametrization of velocity-based constraints is introduced. Finally, the new proposed parametrization of the velocity-based constraints is presented in detail.
- Section 3 contains some implementation notes showing how the new optional parametrization has been programmed in the code.
- Finally, Section 4 contains a series of numerical examples which test and verify the new optional parametrization. All the input files used for the numerical examples are listed in the Appendix.

2 Constrained fast transient dynamics

To solve the governing equations of transient dynamics under some constraints, use is made of the standard method available in EUROPLEXUS, i.e. the method of Lagrange multipliers (see e.g. [5,6]). We start by briefly recalling the governing equations that are at the base of the transient explicit formulation adopted in EUROPLEXUS. Then the time integration scheme is introduced. Finally, the Lagrange multipliers method is outlined.

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 discretization based on Finite Elements, the following set of discrete differential equations in time can be obtained:

$$\boldsymbol{M}\ddot{\boldsymbol{u}} = \boldsymbol{F}_{\text{ext}} - \boldsymbol{F}_{\text{int}} \tag{1}$$

where $\boldsymbol{M}$ is the mass matrix, $\boldsymbol{u}$ are the nodal displacements, $\boldsymbol{F}_{\text{ext}}$ the external forces and $\boldsymbol{F}_{\text{int}}$ the internal forces. The nodal accelerations $\ddot{\boldsymbol{u}}$ are readily obtained from (1) since the mass matrix is lumped.

2.2 Time integration

Time integration of (1) is achieved via a central difference (CD) scheme, which is usually written as:

$$\begin{aligned} \dot{\boldsymbol{u}}^{n+1} &= \dot{\boldsymbol{u}}^{n} + \frac{\Delta t}{2}\left(\ddot{\boldsymbol{u}}^{n} + \ddot{\boldsymbol{u}}^{n+1}\right) \\ \boldsymbol{u}^{n+1} &= \boldsymbol{u}^{n} + \Delta t\left(\dot{\boldsymbol{u}}^{n} + \frac{\Delta t}{2}\ddot{\boldsymbol{u}}^{n}\right) \end{aligned} \tag{2}$$

where $\boldsymbol{u}$ is the vector of nodal displacements, $\dot{\boldsymbol{u}}$ the nodal velocities, $\ddot{\boldsymbol{u}}$ the accelerations, 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 interval used in the discretization process. The integration scheme (2) for the structural domain is implemented as follows in EUROPLEXUS. Assume that a complete solution, i.e. all discretized quantities, are known at time t^n. First, an intermediate (half-step) velocity is introduced:

$$\boldsymbol{v}^{n+1/2} = \dot{\boldsymbol{u}}^{n} + \frac{\Delta t}{2}\ddot{\boldsymbol{u}}^{n} \tag{3}$$

which is denoted $\boldsymbol{v}$ in order to stress the difference with $\dot{\boldsymbol{u}}$. This is the constant velocity that would transform configuration n into configuration $n+1$ over a time interval Δt in the discretization process. From the second of eqs. (3), the new displacements are given by:

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^{n} + \Delta t\boldsymbol{v}^{n+1/2} \tag{4}$$

On this new configuration, the internal forces $\boldsymbol{F}_{\text{int}}^{n+1}$ can now be evaluated by application of the constitutive relations. Then, the new accelerations $\ddot{\boldsymbol{u}}^{n+1}$ can be directly computed via the discretized equilibrium equations (1) and finally the new velocities are obtained from the first of eqs. (2).

It is important to note that in the time integration process the new configuration, induced by the displacements $\boldsymbol{u}^{n+1}$, is obtained first (at the initial time, the configuration is known by definition), then equilibrium is applied on the current configuration, while the velocities $\dot{\boldsymbol{u}}^{n+1}$ corresponding to this new configuration are computed only at the end 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.

2.3 Treatment of constraints by Lagrange multipliers

The method of Lagrange multipliers (LM) is traditionally the main method used in EPX to impose constraints (also called essential boundary conditions), although in recent years also alternative methods based on penalty formulations have been implemented in the code. The main advantages of the LM method are its generality (it combines any type of constraint) and the absence of ad-hoc parameters in the solution. The main drawback is the necessity to solve a linear system, which may be computationally less efficient than the other methods.

The method was first introduced in reference [5] in the framework of the `LIAI` directive of EPX. More recently, it was re-programmed with a modern and flexible data structure based on the Fortran 90 language in the form of the `LINK COUP` directive, see [6]. Finally, it has been generalized to the case of asynchronous constraints, which may occur in models using different time steps not only in time by also in space (spatial partitioning) in the presence of contacts or other non-permanent constraints. A complete description of the resulting method is given in Section 3 of reference [7]. In the following explanation of the method, for simplicity we consider only the case of synchronous constraints [6], since this suffices to discuss the developments (velocity parametrization) introduced in the present work.

Thus, following references [5,6], suppose that a configuration $n+1$ at 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), on the other hand, are known, since they depend only on the current configuration and on time.

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

$$\boldsymbol{m}^{n+1}\boldsymbol{a}^{n+1} = \boldsymbol{f}_{\mathrm{e}}^{n+1} - \boldsymbol{f}_{\mathrm{i}}^{n+1} + \boldsymbol{r}^{n+1} \tag{5}$$

where $\boldsymbol{m}$ is the mass matrix, $\boldsymbol{a}$ the vector of accelerations, $\boldsymbol{f}_{\mathrm{e}}$ and $\boldsymbol{f}_{\mathrm{i}}$ the externally applied loads and internal force vectors, and $\boldsymbol{r}$ indicates the vector of unknown reaction forces produced by the constraints. Note that similar, but distinct, symbols have been used here with respect to eq. (1) to stress the fact that these equations involve only a (usually small) subset of the degrees of freedom of eq. (1).

2.3.1 Velocity constraints at full step

Let us now assume that the imposed essential boundary conditions be expressed by a linear set of constraints on the (full-step) velocities, of the following form, i.e ., considering the bilateral case (equality constraints) for simplicity:

$$\boxed{\boldsymbol{C}^{n+1}\boldsymbol{v}^{n+1} = \boldsymbol{b}^{n+1}} \tag{6}$$

where $\boldsymbol{C}$ is a matrix of known coefficients that may either be constant or variable in time, $\boldsymbol{v}$ is the vector of constrained velocity degrees of freedom, and the right-hand-side $\boldsymbol{b}$ is a known vector. The cases of constraints imposed on accelerations or on displacements rather than on velocities can be treated in a similar way and will be detailed in the following Sections. 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.

In order to introduce the constraint (6) into the equilibrium equation (5), use is made of *Lagrange multipliers*. Without loss of generality, the unknown reactions can be expressed as:

$$\boldsymbol{r} = \boldsymbol{C}^T\boldsymbol{\lambda} \tag{7}$$

where $\boldsymbol{\lambda}$ is the vector of Lagrange multipliers. Substituting into (5) yields:

$$\boldsymbol{m}\boldsymbol{a} = \boldsymbol{f}_{\mathrm{e}} - \boldsymbol{f}_{\mathrm{i}} + \boldsymbol{C}^T\boldsymbol{\lambda} \tag{8}$$

Multiplying both members by $\boldsymbol{C}\boldsymbol{m}^{-1}$ gives:

$$\boldsymbol{C}\boldsymbol{a} = \boldsymbol{C}\boldsymbol{m}^{-1}\left(\boldsymbol{f}_{\mathrm{e}} - \boldsymbol{f}_{\mathrm{i}}\right) + \boldsymbol{C}\boldsymbol{m}^{-1}\boldsymbol{C}^T\boldsymbol{\lambda} \tag{9}$$

and the Lagrange multipliers can be symbolically obtained from:

$$\boldsymbol{C}\boldsymbol{m}^{-1}\boldsymbol{C}^T\boldsymbol{\lambda} = \boldsymbol{C}\boldsymbol{a} - \boldsymbol{C}\boldsymbol{m}^{-1}\left(\boldsymbol{f}_\mathrm{e} - \boldsymbol{f}_\mathrm{i}\right) \tag{10}$$

In order to actually obtain $\boldsymbol{\lambda}$, the term $\boldsymbol{C}\boldsymbol{a}$ has to be determined. To this end, the constraint (6) can be used together with the time integration scheme. The central difference integration scheme for the velocity, see eqs. (2) and (3), can be written as:

$$\boldsymbol{v}^{n+1} = \boldsymbol{v}^{n+1/2} + \frac{\Delta t}{2}\boldsymbol{a}^{n+1} \tag{11}$$

where, as usual, superscript $n+1/2$ is used to indicate the (known) value at the middle of the current step Δt, that spans from the previous configuration n to the current configuration $n+1$.

Substituting (11) into (6) yields:

$$\boldsymbol{C}\boldsymbol{v}^{n+1/2} + \frac{\Delta t}{2}\boldsymbol{C}\boldsymbol{a} = \boldsymbol{b} \qquad \text{and hence} \qquad \boldsymbol{C}\boldsymbol{a} = \frac{2}{\Delta t}\left(\boldsymbol{b} - \boldsymbol{C}\boldsymbol{v}^{n+1/2}\right) \tag{12}$$

In conclusion, the second of expressions (12) substituted into (10) allows to find the Lagrange multipliers $\boldsymbol{\lambda}$, from which the reactions are finally obtained with expression (7).

The method outlined above is implemented as follows. Let us pose:

$$\boxed{\boldsymbol{B}^* = \boldsymbol{C}\boldsymbol{m}^{-1}\boldsymbol{C}^T \qquad \boldsymbol{S} = \frac{2}{\Delta t}\left(\boldsymbol{b} - \boldsymbol{C}\boldsymbol{v}^{n+1/2}\right) \qquad \boldsymbol{W} = \boldsymbol{S} - \boldsymbol{C}\boldsymbol{m}^{-1}\left(\boldsymbol{f}_\mathrm{e} - \boldsymbol{f}_\mathrm{i}\right)} \tag{13}$$

Eq. (10), using (12)-(13), becomes:

$$\boldsymbol{B}^*\boldsymbol{\lambda} = \boldsymbol{W} \tag{14}$$

where $\boldsymbol{B}^*$ is a matrix, called the *matrix of connections*, and $\boldsymbol{W}$ a vector. Both $\boldsymbol{B}^*$ and $\boldsymbol{W}$ are known, as appears from the definition (13).

The code computes both $\boldsymbol{B}^*$ and $\boldsymbol{W}$ at each step, since the coefficients 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 (14) in order to find the Lagrange multipliers, symbolically:

$$\boldsymbol{\lambda} = \left(\boldsymbol{B}^*\right)^{-1}\boldsymbol{W} \tag{15}$$

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 $\boldsymbol{B}^*$ is a square, symmetric matrix. This results from the definition of $\boldsymbol{B}^*$, the first of eqs. (13), and from the fact that $\boldsymbol{m}$ is a square, non-singular, symmetric matrix, hence $\boldsymbol{m}^{-1}$ is also symmetric.

This completes the description of the Lagrange multipliers method in the form that was implemented in PLEXIS-3C, an ancestor of EUROPLEXUS, see [5, 6, 8]. By default, EUROPLEXUS uses a slightly different algorithm, which is described in the next Section. The PLEXIS-3C algorithm, where the velocity-based constraints are imposed on the full-step values by eq. (6), can still be used also in EUROPLEXUS by activating the `OPTI LIAJ` input option. However, before the present work the option was implemented only for the old `LIAI` model of constraints, and had no effect on the more modern `LINK` model, which is now used almost exclusively for new applications.

2.3.2 Velocity constraints at mid-step

An alternative approach to the one detailed in the previous Section consists in expressing the constraint on the new *mid-step velocity* $\boldsymbol{v}^{n+3/2}$ rather than on the new full-step velocity $\dot{\boldsymbol{u}}^{n+1}$ (or $\boldsymbol{v}^{n+1}$). In other words, in place of eq. (6) for the constraints, one can use:

$$\boldsymbol{C}^{n+3/2}\boldsymbol{v}^{n+3/2} = \boldsymbol{b}^{n+3/2} \tag{16}$$

This expression seems preferable to (6) because the mid-step velocity $\boldsymbol{v}$, defined by eq. (3), and not the full-step velocity $\dot{\boldsymbol{u}}$, is the fundamental quantity for the time marching algorithm. Note,

however, that although $\boldsymbol{C}$ and $\boldsymbol{b}$ have different indexes in (16) with respect to (6), they are in reality the same quantities. In fact, for practical reasons they are computed based upon known quantities on the current configuration $n+1$. Therefore, the actually used constraint reads:

$$\boxed{\boldsymbol{C}^{n+1}\boldsymbol{v}^{n+3/2} = \boldsymbol{b}^{n+1}} \tag{17}$$

By using (17) instead of (6), eqs. (7) to (10) remain unchanged, but instead of eq. (11) we need now an expression for $\boldsymbol{v}^{n+3/2}$, given by:

$$\boldsymbol{v}^{n+3/2} = \boldsymbol{v}^{n+1/2} + \frac{\Delta t^n + \Delta t^{n+1}}{2}\boldsymbol{a}^{n+1} \tag{18}$$

Here we indicate by Δt^n the time step that has lead to the current configuration (previously denoted by Δt), i.e. $\Delta t^n = t^{n+1} - t^n$, while Δt^{n+1} is the next time step: $\Delta t^{n+1} = t^{n+2} - t^{n+1}$. Note, incidentally, that the size of the next time step Δt^{n+1} is already known when reaction forces are computed on the current configuration, because it is estimated during the loop over elements used to compute the internal forces, see [5, 6]. By substituting (18) into (17) we get, in place of eqs. (12) and (13):

$$\begin{gathered}\boldsymbol{C}\boldsymbol{v}^{n+1/2} + \frac{\Delta t^n + \Delta t^{n+1}}{2}\boldsymbol{C}\boldsymbol{a} = \boldsymbol{b} \qquad \text{and hence} \qquad \boldsymbol{C}\boldsymbol{a} = \frac{2}{\Delta t^n + \Delta t^{n+1}}\left(\boldsymbol{b} - \boldsymbol{C}\boldsymbol{v}^{n+1/2}\right) \\ \boxed{\boldsymbol{B}^* = \boldsymbol{C}\boldsymbol{m}^{-1}\boldsymbol{C}^T \qquad \boldsymbol{S} = \frac{2}{\Delta t^n + \Delta t^{n+1}}\left(\boldsymbol{b} - \boldsymbol{C}\boldsymbol{v}^{n+1/2}\right) \qquad \boldsymbol{W} = \boldsymbol{S} - \boldsymbol{C}\boldsymbol{m}^{-1}\left(\boldsymbol{f}_{\mathrm{e}} - \boldsymbol{f}_{\mathrm{i}}\right)}\end{gathered} \tag{19}$$

As shown in [9, 10] for a simple academic 1D bar impact test case, this second algorithm based upon eq. (17) gives much smoother numerical results than eq. (6) in the presence of pinball (`PINB`) contact conditions, and is therefore preferable to the other approach. For the other (permanent) boundary condition types, the two methods appear substantially equivalent. For this reason, the second algorithm (17)-(19), where velocity-based constraints are imposed on the mid-step rather than on the full-step velocity, is now used by default in EUROPLEXUS for all types of constraints.

The first algorithm, where the velocity-based constraints are imposed on the full-step values by eq. (6), is still usable in EUROPLEXUS and can be activated by the `OPTI LIAJ` input option. However, before the present work that option was implemented only for the old `LIAI` model of constraints, and had no effect on the more modern `LINK` model which is now used almost exclusively for new applications.

2.3.3 Algorithm behavior at the initial time

Yet another reason for preferring the expression (17) involving mid-step velocities rather than expression (6) is related to the behavior of the algorithm at start-up, i.e. at the initial time t^0 of the computation, corresponding to step 0.

Suppose that a constraint on velocities is imposed already at t^0. Since initial velocities are independently prescribed by the user in the input file, there are two possibilities: (*i*) either these velocities satisfy the constraint or (*ii*) they do not satisfy the constraint. A problem may therefore arise with the first method because it tries to enforce a constraint on velocity values at t^0, which are already set and may not be altered. Thus in case (*ii*) it may simply be impossible to satisfy the constraint at t^0.

The second method, however, is sounder because the velocities at $t^{0+1/2}$ are not set in the input file and may therefore be constrained. In fact, the EUROPLEXUS implementation is such that, when the first method (`OPTI LIAJ`) is chosen, a special treatment is performed at step 0 which consists in using the second method (only for this initial step), in order to avoid the above mentioned difficulty. In other words, it is like if the `OPTI LIAJ` option would be active only starting from step 1.

Note that at the initial time t^0 of a computation (step 0, corresponding to $n = -1$) eqs. (17-18) should be replaced by:

$$\boldsymbol{C}^0\boldsymbol{v}^{1/2} = \boldsymbol{b}^0 \tag{20}$$

$$\boldsymbol{v}^{1/2} = \boldsymbol{v}^0 + \frac{\Delta t^0}{2}\boldsymbol{a}^0 \tag{21}$$

where $\boldsymbol{v}^0$ is the initial, known velocity imposed by the user and $\Delta t^0 = t^1 - t^0$. Then instead of eqs. (19) one gets:

$$\boldsymbol{C}\boldsymbol{v}^0 + \frac{\Delta t^0}{2}\boldsymbol{C}\boldsymbol{a} = \boldsymbol{b} \qquad \text{and hence} \qquad \boldsymbol{C}\boldsymbol{a} = \frac{2}{\Delta t^0}\left(\boldsymbol{b} - \boldsymbol{C}\boldsymbol{v}^0\right)$$

$$\boxed{\boldsymbol{B}^* = \boldsymbol{C}\boldsymbol{m}^{-1}\boldsymbol{C}^T \qquad \boldsymbol{S} = \frac{2}{\Delta t^0}\left(\boldsymbol{b} - \boldsymbol{C}\boldsymbol{v}^0\right) \qquad \boldsymbol{W} = \boldsymbol{S} - \boldsymbol{C}\boldsymbol{m}^{-1}\left(\boldsymbol{f}_{\mathrm{e}} - \boldsymbol{f}_{\mathrm{i}}\right)} \tag{22}$$

2.3.4 Acceleration constraints

We consider also the case in which constraints are expressed on the accelerations rather than on the velocities. In other words, in place of (17) we would have:

$$\boldsymbol{C}^{n+1}\boldsymbol{a}^{n+1} = \boldsymbol{b}^{n+1} \qquad \text{or simply} \qquad \boldsymbol{C}\boldsymbol{a} = \boldsymbol{b} \tag{23}$$

This formulation is worth being introduced because in some special cases the constraints seem to be more naturally imposed on the accelerations rather than on the velocities. An example is a constraint of constant distance (`DIST`) between points, see [6]. Lagrange multipliers may be introduced in exactly the same manner as before and eqs. (7) to (10) continue to hold without any modifications. However, in place of eqs. (18-19) we may in this case replace directly the (known) expression (23) of $\boldsymbol{C}\boldsymbol{a}$, i.e. the right-hand side term $\boldsymbol{b}$, into (10), which becomes:

$$\boldsymbol{C}\boldsymbol{m}^{-1}\boldsymbol{C}^T\boldsymbol{\lambda} = \boldsymbol{b} - \boldsymbol{C}\boldsymbol{m}^{-1}\left(\boldsymbol{f}_\mathrm{e} - \boldsymbol{f}_\mathrm{i}\right) \tag{24}$$

Therefore, in place of the second of eqs. (19) we get in this case the simpler expressions:

$$\boxed{\boldsymbol{B}^* = \boldsymbol{C}\boldsymbol{m}^{-1}\boldsymbol{C}^T \qquad \boldsymbol{S} = \boldsymbol{b} \qquad \boldsymbol{W} = \boldsymbol{S} - \boldsymbol{C}\boldsymbol{m}^{-1}\left(\boldsymbol{f}_\mathrm{e} - \boldsymbol{f}_\mathrm{i}\right)} \tag{25}$$

The solution of the system (14) or (15) proceeds exactly as above, and so does the calculation of the reactions (7).

2.3.5 Displacement constraints

Finally, let us consider also the case of constraints imposed on the displacements, i.e. of the form:

$$\boldsymbol{C}\boldsymbol{d} = \boldsymbol{b} \tag{26}$$

However, note that in this case the current displacements $\boldsymbol{d} = \boldsymbol{d}^{n+1}$ (current configuration) are already known at the moment of imposing the constraints and may not be altered. Therefore, we can only write displacement constraints on the *next* step, i.e. at time $n+2$ in the formalism adopted so far:

$$\boldsymbol{C}^{n+1}\boldsymbol{d}^{n+2} = \boldsymbol{b}^{n+1} \tag{27}$$

Note also that, in analogy with the comments to eq. (17) in Section 2.3.2, the coefficients appearing in both $\boldsymbol{C}$ and $\boldsymbol{b}$ may only be computed (explicitly) on the current configuration and therefore we use a superscript $n+1$ for these quantities in eq. (27).

Lagrange multipliers may be introduced in exactly the same manner as before and eqs. (7) to (10) continue to hold without any modifications. In order to actually obtain $\boldsymbol{\lambda}$ from (10), the term $\boldsymbol{C}\boldsymbol{a}$ has to be determined. To this end, the constraint (27) can be used together with the time integration scheme.

The central difference integration scheme for the displacement and the velocity, see eqs. (2) and (3), can be written as:

$$\begin{aligned} \boldsymbol{d}^{n+2} &= \boldsymbol{d} + \Delta t^{n+1}\boldsymbol{v}^{n+3/2} \\ \boldsymbol{v}^{n+3/2} &= \boldsymbol{v}^{n+1/2} + \frac{\Delta t^n + \Delta t^{n+1}}{2}\boldsymbol{a} \end{aligned} \tag{28}$$

where Δt^n and Δt^{n+1} have the same meaning as before (see definitions in Section 2.3.2). Substituting (28) into (27) yields:

$$\boldsymbol{C}\boldsymbol{d}^{n+2} = \boldsymbol{C}\left(\boldsymbol{d} + \Delta t^{n+1}\boldsymbol{v}^{n+3/2}\right) = \boldsymbol{C}\left[\boldsymbol{d} + \Delta t^{n+1}\left(\boldsymbol{v}^{n+1/2} + \frac{\Delta t^n + \Delta t^{n+1}}{2}\boldsymbol{a}\right)\right] = \boldsymbol{b} \tag{29}$$

and hence:

$$\boldsymbol{C}\boldsymbol{a} = \frac{2}{\Delta t^{n+1}\left(\Delta t^n + \Delta t^{n+1}\right)}\left(\boldsymbol{b} - \boldsymbol{C}\boldsymbol{d} - \Delta t^{n+1}\boldsymbol{C}\boldsymbol{v}^{n+1/2}\right) \tag{30}$$

Therefore, in place of the second of eqs. (19) we get in this case the expressions:

$$\boxed{\begin{aligned} \boldsymbol{B}^* &= \boldsymbol{C}\boldsymbol{m}^{-1}\boldsymbol{C}^T \\ \boldsymbol{S} &= \frac{2}{\Delta t^{n+1}\left(\Delta t^n + \Delta t^{n+1}\right)}\left(\boldsymbol{b} - \boldsymbol{C}\boldsymbol{d} - \Delta t^{n+1}\boldsymbol{C}\boldsymbol{v}^{n+1/2}\right) \\ \boldsymbol{W} &= \boldsymbol{S} - \boldsymbol{C}\boldsymbol{m}^{-1}\left(\boldsymbol{f}_{\mathrm{e}} - \boldsymbol{f}_{\mathrm{i}}\right) \end{aligned}} \tag{31}$$

The solution of the system (14) or (15) proceeds exactly as above, and so does the calculation of reactions (7).

2.3.6 Constraints transformation before system solution

In Section 2.3.2 it has been pointed out that the standard way of imposing links in EUROPLEXUS is to write constraints on the velocities. This is indeed the case in the majority of link types that are available in the system. However, there are cases where it may appear more "natural" to write conditions on the accelerations. This has been outlined in Section 2.3.4, and a practical example of such a constraint is the case of imposed constant distance (`DIST`) between two nodes, see [6].

When several (non-independent) links are imposed together, forming a set (linear system) of links, it is obviously essential that all of them be expressed in the same type of quantities, i.e. either all on velocities or all on accelerations.

The strategy adopted in EUROPLEXUS within the implementation of the `LIAI` and `LINK COUP` directives is as follows. Each constraint is written in the more "natural" form, i.e. either on velocities or on accelerations, depending upon the specific type. Then the system of links is assembled but, immediately prior to solution, any velocity-related constraints are transformed into the equivalent acceleration-related form. Those constraints which are already in the form of accelerations are just skipped.

The transformation procedure is implemented in subroutine SMVCAL (called from FLIAIS for the relevant types of constraints) for the `LIAI` model, while for the `LINK COUP` model it is implemented in subroutine SMVCALCUL (see Section 3).

To perform the transformation of velocity-based into equivalent acceleration-based constraints we observe the expressions of the term $\boldsymbol{C}\boldsymbol{a}$ (i.e. of the acceleration-based constraint) in the various cases:

- For an acceleration-based constraint one has, trivially, eq. (23):

$$\boldsymbol{C}\boldsymbol{a} = \boldsymbol{b} \tag{32}$$

  i.e. the original constraint, so that no transformation is needed.
- For a constraint based on mid-step velocity, eq. (17):

$$\boldsymbol{C}\boldsymbol{v}^{n+3/2} = \boldsymbol{b} \tag{33}$$

  one has the second of eqs. (19):

$$\boldsymbol{C}\boldsymbol{a} = \frac{2}{\Delta t^n + \Delta t^{n+1}}\left(\boldsymbol{b} - \boldsymbol{C}\boldsymbol{v}^{n+1/2}\right) \tag{34}$$

The expression (34) can be viewed as the equivalent of (33) expressed on the accelerations rather than on the velocities. In other words, the velocity-based constraint (33) can be replaced, without any effect on results, by the acceleration-based constraint (34).

Therefore, in order to implement the transformation of a velocity-based constraint in the code, it is sufficient to "correct" (replace) the right-hand side computed "on the velocities" ($\boldsymbol{b}$) as follows:

$$\boldsymbol{b} \leftarrow \frac{2}{\Delta t^n + \Delta t^{n+1}}\left(\boldsymbol{b} - \boldsymbol{C}\boldsymbol{v}^{n+1/2}\right) \tag{35}$$

In the implementation of the new `LINK` directive presented in Section 4 of [6], the strategy described in Section 3 of the same reference for the old `LIAI` directive has been maintained, as concerns the solution of a set of coupled links. In the case of a single independent link, however, a direct solution is performed, therefore each type of constraint (either on the velocities or on the accelerations) is solved without any transformation, as described in detail in Sections 4.10 and 4.10.1 of reference [6].

2.3.7 Parametrization of the velocity constraints

Before the present development, EUROPLEXUS offered two possibilities to impose velocity-based constraints:

- *Mid-step* constraints, where the conditions are imposed on the new mid-step velocity $\boldsymbol{v}^{n+3/2}$, i.e. at time $t^{n+1} + \frac{\Delta t^{n+1}}{2}$. This is the default procedure in EUROPLEXUS and is described in Section 2.3.2.
- *Full-step* constraints, where the conditions are imposed on the new (current) full-step velocity $\boldsymbol{v}^{n+1}$, i.e. at time t^{n+1}. In EUROPLEXUS this procedure is activated by the `OPTI LIAJ` input command, but prior to the present work it only had effect on the old `LIAI` constraint model, and not on the `LINK COUP` constraint model. It is described in Section 2.3.1. This was the default procedure in PLEXIS-3C, one of the ancestors of EUROPLEXUS.

The motivations for choosing the mid-step velocity constraints as a default in EUROPLEXUS were given in [6, 9, 10] and can be summarized as follows:

- The mid-step velocity is the fundamental quantity in the central difference time integration scheme. In EUROPLEXUS the full-step velocity is only computed for post-processing (and to evaluate the kinetic energy used in the energy balance check) and does not appear directly in the time stepping scheme.
- The start-up of the time stepping procedure may be inconsistent (depending on the initial velocities chosen by the user) with the full-step velocity procedure, as detailed in Section 2.3.3. Therefore the mid-step procedure is used at step 0 even when the full-step procedure is chosen (EUROPLEXUS with `OPTI LIAJ` or PLEXIS-3C).
- Simple test cases involving elastic impacts with the pinball contact model (`PINB`) using `LIAI` in EUROPLEXUS showed large oscillations (intermittent contact) with the full-step procedure, while the results with the mid-step procedure were very smooth (continuous contact).

Recent investigations concerning (elastic) impact tests have shown that the mid-step velocity constraints strategy leads to smooth solutions (few oscillations), but some energy is lost at the moment of the impact, namely each time some previously free nodes get into contact, so that the total energy of the system is not exactly conserved.

On the contrary, the full-step velocity constraints strategy produces a lot of oscillations (if the material is elastic) but it conserves much better the energy of the system.

Therefore, it has been proposed to implement a *parametrization* of the velocity constraints, allowing to choose an intermediate value of the time at which the constraints are imposed. The goal is that of reducing the spurious energy dissipation observed in contact-impact applications with the mid-step strategy but hopefully without triggering the large oscillations produced by the full-step strategy. The theoretical background for the parametrization of contact constraints based on the velocities was inspired by reference [11] and will be presented in Section 2.3.8 below.

The parametrization may be formulated as follows. We introduce a scalar parameter α, with $0 \leq \alpha \leq 1$, and impose the velocity-based constraints at time:

$$t^{\alpha} = t^{n+1} + \alpha \frac{\Delta t^{n+1}}{2} \qquad (0 \leq \alpha \leq 1) \tag{36}$$

Thus, $\alpha = 0$ produces full-step velocity constraints, $\alpha = 1$ produces mid-step velocity constraints and intermediate values of α produce velocity constrains at intermediate times between these two extremes:

$$\begin{aligned} t^{\alpha=0} &= t^{n+1} && \text{full step} \\ t^{\alpha=1} &= t^{n+1} + \frac{\Delta t^{n+1}}{2} && \text{mid step} \end{aligned} \tag{37}$$

Instead of using eqs. (6) or (17) the velocity constraint is written as:

$$\boxed{\boldsymbol{C}^{n+1} \boldsymbol{v}^{n+1+\alpha/2} = \boldsymbol{b}^{n+1}} \tag{38}$$

where in place of (11) or (18) the velocity at the parametrized time is given by:

$$\boldsymbol{v}^{n+1+\alpha/2} = \boldsymbol{v}^{n+1/2} + \frac{\Delta t^n + \alpha \Delta t^{n+1}}{2} \boldsymbol{a}^{n+1} \tag{39}$$

By substituting (39) into (38) we get, in place of eqs. (12-13) or (19):

$$\begin{gathered} \boldsymbol{C}\boldsymbol{v}^{n+1/2} + \frac{\Delta t^n + \alpha \Delta t^{n+1}}{2} \boldsymbol{C}\boldsymbol{a} = \boldsymbol{b} \qquad \text{and hence} \qquad \boldsymbol{C}\boldsymbol{a} = \frac{2}{\Delta t^n + \alpha \Delta t^{n+1}} \left(\boldsymbol{b} - \boldsymbol{C}\boldsymbol{v}^{n+1/2} \right) \\ \boxed{\boldsymbol{B}^* = \boldsymbol{C}\boldsymbol{m}^{-1}\boldsymbol{C}^T \qquad \boldsymbol{S} = \frac{2}{\Delta t^n + \alpha \Delta t^{n+1}} \left(\boldsymbol{b} - \boldsymbol{C}\boldsymbol{v}^{n+1/2} \right) \qquad \boldsymbol{W} = \boldsymbol{S} - \boldsymbol{C}\boldsymbol{m}^{-1} \left(\boldsymbol{f}_\mathrm{e} - \boldsymbol{f}_\mathrm{i} \right)} \end{gathered} \tag{40}$$

2.3.8 Theoretical background for the parametrization of velocity constraints

We now briefly present the theoretical background behind the proposed parametrization of velocity-based constraints and the expected effects of the parameter on energy conservation in contact-impact applications.

EPX considers a contact condition as a kinematic constraint applied when a penetration (which is a non-physical condition) is detected. In the contact treatment strategy used in EPX, through the kinematic constraint the penetration value is maintained constant between two successive discrete time instants t^{n+1} and t^{n+2}. In other words, when detected, the penetration is considered *a priori* as acceptably small and does not need to be reduced, because the critical time step Δt_crit is in general very small (of the order of 10^{-6} s).

Since in the CD time scheme of EPX (cfr. Section 2.2) the displacement $\boldsymbol{d}^{n+2}$ at time t^{n+2} is completely determined by the mid-step velocity $\boldsymbol{v}^{n+3/2}$, the kinematic condition corresponding to maintaining the relative displacement (i.e. the penetration) constant can be expressed as:

$$\boldsymbol{C}\boldsymbol{v}^{n+3/2} = \boldsymbol{0} \tag{41}$$

where $\boldsymbol{C}$ is the link matrix corresponding to the kinematic constraints related to the contact conditions.

However, the inconvenience of representing a contact condition with such a kinematic constraint is that it leads to a spurious energy dissipation. This algorithmic energy loss can be very high in some situations. For instance, in case of an impact between rigid bodies, this approach is equivalent to an inelastic shock, where the bodies remain glued together after the impact, without any rebound, and where possibly all the initial energy could be dissipated (e.g. for two rigid bodies with the same mass and opposite velocities or for a body impacting a fixed rigid obstacle). In many practical cases the energy dissipation is quite low, but in some situations it could be non-negligible.

The article by Fetecau *et al.* [11], from which this Section is inspired, presents a rigorous variational framework to model collisions between rigid, i.e. non-deformable, bodies of arbitrary shape. In that work the contact conditions are not represented by the Lagrange multiplier method as they are in EPX, but are deduced instead from the Lagrangian mechanics formalism involving the "principle of minimum action". In that approach, impacts between bodies are considered instantaneous, as they are non-deformable, and the energy conservation is imposed explicitly, since this condition follows directly from the variational principle together with the standard Euler-Lagrange equations for multi-body dynamics.

Since impacts must also respect the non-penetrability conditions, in addition to energy conservation, it is assumed that a kinematic constraint similar to the one in eq. (41) is applied. However, this alternative form of the constraint is not imposed on the displacement at the end of the next time step t^{n+2}, but at some intermediate time between the instants t^{n+1} and t^{n+2}. The alternative form of the constraint is then:

$$\boldsymbol{C}\boldsymbol{v}^{n+1+\alpha/2} = \boldsymbol{0} \tag{42}$$

where a new parameter α is introduced. It is important to note that only when $\alpha = 1$, is the condition in eq. (42) identical to the condition in eq. (41). As it will be shown next, the new parameter α can be chosen such that the total energy is conserved, i.e. so that the sum of the internal and kinetic energies is the same before and after the contact.

To this end, it is convenient to express the difference in the kinetic energy between two successive mid-step instants $t^{n+1/2}$ and $t^{n+3/2}$ as follows:

$$\Delta E = \frac{1}{2}\left(\boldsymbol{\Gamma}^{n+3/2}\right)^T \boldsymbol{M}^{-1}\boldsymbol{\Gamma}^{n+3/2} - \frac{1}{2}\left(\boldsymbol{\Gamma}^{n+1/2}\right)^T \boldsymbol{M}^{-1}\boldsymbol{\Gamma}^{n+1/2} \tag{43}$$

where $\boldsymbol{M}$ is the mass matrix, assumed here to be symmetric so that $\boldsymbol{M}^T = \boldsymbol{M}$ (actually, in EPX the mass matrix is not only symmetric but typically diagonal, or lumped, as is customary in explicit codes), and $\boldsymbol{\Gamma}^{n+1/2}$ and $\boldsymbol{\Gamma}^{n+3/2}$ are the values of momentum at the two different time instants:

$$\begin{aligned} \boldsymbol{\Gamma}^{n+1/2} &= \boldsymbol{M}\boldsymbol{v}^{n+1/2} \\ \boldsymbol{\Gamma}^{n+3/2} &= \boldsymbol{M}\boldsymbol{v}^{n+3/2} \end{aligned} \tag{44}$$

To check that the expression (43) actually represents the kinetic energy, we expand the first term by using the definition (44) of the momentum obtaining:

$$\begin{aligned} \frac{1}{2}\left(\boldsymbol{\Gamma}^{n+3/2}\right)^T \boldsymbol{M}^{-1}\boldsymbol{\Gamma}^{n+3/2} &= \frac{1}{2}\left(\boldsymbol{M}\boldsymbol{v}^{n+3/2}\right)^T \boldsymbol{M}^{-1}\boldsymbol{M}\boldsymbol{v}^{n+3/2} \\ &= \frac{1}{2}\left(\boldsymbol{v}^{n+3/2}\right)^T \boldsymbol{M}^T\boldsymbol{M}^{-1}\boldsymbol{M}\boldsymbol{v}^{n+3/2} \\ &= \frac{1}{2}\left(\boldsymbol{v}^{n+3/2}\right)^T \boldsymbol{M}\boldsymbol{v}^{n+3/2} \ \blacksquare \end{aligned} \tag{45}$$

where in the second passage we have used the fact that $\boldsymbol{M}$ is symmetric.

By introducing the momentum difference as:

$$\Delta\boldsymbol{\Gamma} = \boldsymbol{\Gamma}^{n+3/2} - \boldsymbol{\Gamma}^{n+1/2} \tag{46}$$

the expression in eq. (43) becomes:

$$\Delta E = \left(\boldsymbol{\Gamma}^{n+1/2}\right)^T \boldsymbol{M}^{-1}\Delta\boldsymbol{\Gamma} + \frac{1}{2}\left(\Delta\boldsymbol{\Gamma}\right)^T \boldsymbol{M}^{-1}\Delta\boldsymbol{\Gamma} \tag{47}$$

To show that (47) is equivalent to (43) we proceed as follows. Replacing the definition of $\Delta\boldsymbol{\Gamma}$ eq. (46) into (47) gives:

$$\begin{aligned} \Delta E = &\left(\boldsymbol{\Gamma}^{n+1/2}\right)^T \boldsymbol{M}^{-1}\boldsymbol{\Gamma}^{n+3/2} - \left(\boldsymbol{\Gamma}^{n+1/2}\right)^T \boldsymbol{M}^{-1}\boldsymbol{\Gamma}^{n+1/2} + \\ &\frac{1}{2}\left(\boldsymbol{\Gamma}^{n+3/2} - \boldsymbol{\Gamma}^{n+1/2}\right)^T \boldsymbol{M}^{-1}\left(\boldsymbol{\Gamma}^{n+3/2} - \boldsymbol{\Gamma}^{n+1/2}\right) \\ = &\frac{1}{2}\left(\boldsymbol{\Gamma}^{n+3/2}\right)^T \boldsymbol{M}^{-1}\left(\boldsymbol{\Gamma}^{n+3/2} - \boldsymbol{\Gamma}^{n+1/2}\right) - \frac{1}{2}\left(\boldsymbol{\Gamma}^{n+1/2}\right)^T \boldsymbol{M}^{-1}\left(\boldsymbol{\Gamma}^{n+3/2} - \boldsymbol{\Gamma}^{n+1/2}\right) + \\ &\left(\boldsymbol{\Gamma}^{n+1/2}\right)^T \boldsymbol{M}^{-1}\boldsymbol{\Gamma}^{n+3/2} - \left(\boldsymbol{\Gamma}^{n+1/2}\right)^T \boldsymbol{M}^{-1}\boldsymbol{\Gamma}^{n+1/2} \\ = &\frac{1}{2}\left(\boldsymbol{\Gamma}^{n+3/2}\right)^T \boldsymbol{M}^{-1}\boldsymbol{\Gamma}^{n+3/2} - \frac{1}{2}\left(\boldsymbol{\Gamma}^{n+3/2}\right)^T \boldsymbol{M}^{-1}\boldsymbol{\Gamma}^{n+1/2} - \\ &\frac{1}{2}\left(\boldsymbol{\Gamma}^{n+1/2}\right)^T \boldsymbol{M}^{-1}\boldsymbol{\Gamma}^{n+3/2} + \frac{1}{2}\left(\boldsymbol{\Gamma}^{n+1/2}\right)^T \boldsymbol{M}^{-1}\boldsymbol{\Gamma}^{n+1/2} + \\ &\left(\boldsymbol{\Gamma}^{n+1/2}\right)^T \boldsymbol{M}^{-1}\boldsymbol{\Gamma}^{n+3/2} - \left(\boldsymbol{\Gamma}^{n+1/2}\right)^T \boldsymbol{M}^{-1}\boldsymbol{\Gamma}^{n+1/2} \\ = &\frac{1}{2}\left(\boldsymbol{\Gamma}^{n+3/2}\right)^T \boldsymbol{M}^{-1}\boldsymbol{\Gamma}^{n+3/2} - \frac{1}{2}\left(\boldsymbol{\Gamma}^{n+3/2}\right)^T \boldsymbol{M}^{-1}\boldsymbol{\Gamma}^{n+1/2} + \\ &\frac{1}{2}\left(\boldsymbol{\Gamma}^{n+1/2}\right)^T \boldsymbol{M}^{-1}\boldsymbol{\Gamma}^{n+3/2} - \frac{1}{2}\left(\boldsymbol{\Gamma}^{n+1/2}\right)^T \boldsymbol{M}^{-1}\boldsymbol{\Gamma}^{n+1/2} \end{aligned} \tag{48}$$

The last expression (48) of ΔE coincides with (43) if the second and third terms cancel out, i.e. provided:

$$\left(\boldsymbol{\Gamma}^{n+3/2}\right)^T \boldsymbol{M}^{-1}\boldsymbol{\Gamma}^{n+1/2} \stackrel{?}{=} \left(\boldsymbol{\Gamma}^{n+1/2}\right)^T \boldsymbol{M}^{-1}\boldsymbol{\Gamma}^{n+3/2} \tag{49}$$

Now, we know that, if $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ are three matrices with compatible dimensions for multiplication, their product and triple product satisfy:

$$\boxed{(\boldsymbol{AB})^T = \boldsymbol{B}^T\boldsymbol{A}^T} \qquad \boxed{(\boldsymbol{ABC})^T = \boldsymbol{C}^T\boldsymbol{B}^T\boldsymbol{A}^T} \tag{50}$$

So, by taking the transpose of the first member of (49) we get:

$$\begin{aligned}\left[\left(\boldsymbol{\Gamma}^{n+3/2}\right)^T \boldsymbol{M}^{-1}\boldsymbol{\Gamma}^{n+1/2}\right]^T &= \left(\boldsymbol{\Gamma}^{n+1/2}\right)^T \left(\boldsymbol{M}^{-1}\right)^T \boldsymbol{\Gamma}^{n+3/2} \\ &= \left(\boldsymbol{\Gamma}^{n+1/2}\right)^T \boldsymbol{M}^{-1}\boldsymbol{\Gamma}^{n+3/2}\end{aligned} \tag{51}$$

where the last passage has exploited the fact that, since $\boldsymbol{M}$ is symmetric, its inverse $\boldsymbol{M}^{-1}$ is also symmetric, and the transpose of a symmetric matrix is the matrix itself, i.e. $\left(\boldsymbol{M}^{-1}\right)^T = \boldsymbol{M}^{-1}$.

Eq. (51) tells us that the right hand side of (49) is the transpose of its left hand side, so the equality in (49) would not hold in general. However, in the present particular case we know that each side of (49) represents a scalar s, because the $\boldsymbol{\Gamma}$ are one-dimensional $[n \times 1]$ (i.e. column) vectors. Then, since the transpose of a scalar is the scalar itself ($s^T = s$), we can conclude that the equality in (49) does indeed hold, and therefore eq. (47) is equivalent to (43).■

Thus, in general, if $\boldsymbol{a}$ and $\boldsymbol{c}$ are $[n \times 1]$ (i.e., column) vectors and $\boldsymbol{B}$ is a symmetric $[n \times n]$ matrix, so that $\boldsymbol{B}^T = \boldsymbol{B}$, then their triple product is a scalar s:

$$\boldsymbol{a}^T\boldsymbol{B}\boldsymbol{c} = s \tag{52}$$

and:

$$\boxed{\boldsymbol{c}^T\boldsymbol{B}\boldsymbol{a} = \boldsymbol{a}^T\boldsymbol{B}\boldsymbol{c}} \qquad \boldsymbol{B} \text{ symmetric matrix;} \quad \boldsymbol{a}, \boldsymbol{c} \text{ column vectors} \tag{53}$$

In fact, by taking the transpose of both members in eq. (52) we get:

$$\left(\boldsymbol{a}^T\boldsymbol{B}\boldsymbol{c}\right)^T = s^T \tag{54}$$

which, by using the expression of the triple product in (50), the above mentioned property $\boldsymbol{B}^T = \boldsymbol{B}$, the identity $s^T = s$ and the eq. (52), becomes:

$$\boldsymbol{c}^T\boldsymbol{B}^T\left(\boldsymbol{a}^T\right)^T = \boldsymbol{c}^T\boldsymbol{B}\boldsymbol{a} = s^T = s = \boldsymbol{a}^T\boldsymbol{B}\boldsymbol{c} \tag{55}$$

thus proving (53).■

The identity (53) will be used in some of the subsequent developments. Moreover, the momentum change must be equal to the impulse of the total force $\boldsymbol{f}$:

$$\Delta\boldsymbol{\Gamma} = \Delta t\, \boldsymbol{f} \tag{56}$$

which, replaced into (47), leads to the following expression of the change in kinetic energy:

$$\begin{aligned}\Delta E &= \left(\boldsymbol{\Gamma}^{n+1/2}\right)^T \boldsymbol{M}^{-1}\Delta\boldsymbol{\Gamma} + \frac{1}{2}\left(\Delta\boldsymbol{\Gamma}\right)^T \boldsymbol{M}^{-1}\Delta\boldsymbol{\Gamma} \\ &= \Delta t \left(\boldsymbol{\Gamma}^{n+1/2}\right)^T \boldsymbol{M}^{-1}\boldsymbol{f} + \frac{1}{2}\Delta t^2 \boldsymbol{f}^T\boldsymbol{M}^{-1}\boldsymbol{f} \\ &= \Delta t \left(\boldsymbol{M}\boldsymbol{v}^{n+1/2}\right)^T \boldsymbol{M}^{-1}\boldsymbol{f} + \frac{1}{2}\Delta t^2 \boldsymbol{f}^T\boldsymbol{M}^{-1}\boldsymbol{f} \\ &= \Delta t \left(\boldsymbol{v}^{n+1/2}\right)^T \boldsymbol{M}^T\boldsymbol{M}^{-1}\boldsymbol{f} + \frac{1}{2}\Delta t^2 \boldsymbol{f}^T\boldsymbol{M}^{-1}\boldsymbol{f} \\ &= \Delta t \left(\boldsymbol{v}^{n+1/2}\right)^T \boldsymbol{f} + \frac{1}{2}\Delta t^2 \boldsymbol{f}^T\boldsymbol{M}^{-1}\boldsymbol{f}\end{aligned} \tag{57}$$

where the definition (44) of $\boldsymbol{\Gamma}$ has been used in the second passage, the first of (50) in the third passage while the last passage has exploited the fact that $\boldsymbol{M}$ is symmetric, so that $\boldsymbol{M}^T\boldsymbol{M}^{-1} = \boldsymbol{M}\boldsymbol{M}^{-1} = \boldsymbol{I}$, where $\boldsymbol{I}$ is the identity matrix.

In this context, the total forces have three components: external, internal and contact interaction forces:

$$\boldsymbol{f} = \boldsymbol{f}_e - \boldsymbol{f}_i + \boldsymbol{f}_c = \boldsymbol{f}_{\text{free}} + \boldsymbol{f}_c \tag{58}$$

The external forces are due to the applied loading conditions (e.g. gravity), the internal forces are due to stresses caused by the deformation, and the contact interaction forces are here the main unknowns. Namely, they need to be determined such that the condition in eq. (42) is satisfied when contact is detected. In (58) it has been set:

$$\boldsymbol{f}_{\text{free}} = \boldsymbol{f}_e - \boldsymbol{f}_i \tag{59}$$

to simplify the subsequent derivations. These are the "free" forces, i.e. the forces in the absence of contact. It is important to stress that all types of forces are considered at the (current) time t^{n+1}, so the index $n+1$ can be omitted.

In addition to the expression for the kinetic energy change eq. (57), the work of forces between two consecutive (mid-step) discrete instants $t^{n+1/2}$ and $t^{n+3/2} = t^{n+1/2} + \Delta t$ in the CD time integration scheme can be expressed as:

$$\Delta W = \Delta t \left(\boldsymbol{v}^{n+1}\right)^T \boldsymbol{f} = \Delta t \left(\boldsymbol{v}^{n+1}\right)^T \boldsymbol{f}_{\text{free}} \tag{60}$$

The last passage is justified by the fact that to each contact force there corresponds an equal and opposite reaction, so that the net work performed by such forces vanishes.

It is easily verified that ΔE and ΔW computed in this way (eqs. 57 and 60, respectively) are always equal in the absence of contact ($\boldsymbol{f_c} = \boldsymbol{0}$), i.e. $\Delta E_{\text{free}} = \Delta W$, which confirms the energy consistency of the CD integration scheme used in EPX. In fact, from the last expression of (57) we obtain:

$$\begin{aligned}
\Delta E_{\text{free}} &= \Delta t \left(\boldsymbol{v}^{n+1/2}\right)^T \boldsymbol{f}_{\text{free}} + \frac{1}{2}\Delta t^2 \boldsymbol{f}_{\text{free}}^T \boldsymbol{M}^{-1} \boldsymbol{f}_{\text{free}} \\
&= \Delta t \left[\boldsymbol{v}^{n+1/2} + \frac{1}{2}\Delta t \left(\boldsymbol{M}^{-1}\right)^T \boldsymbol{f}_{\text{free}}\right]^T \boldsymbol{f}_{\text{free}} \\
&= \Delta t \left(\boldsymbol{v}^{n+1/2} + \frac{1}{2}\Delta t\, \boldsymbol{M}^{-1} \boldsymbol{f}_{\text{free}}\right)^T \boldsymbol{f}_{\text{free}} \\
&= \Delta t \left(\boldsymbol{v}^{n+1/2} + \frac{1}{2}\Delta t\, \boldsymbol{a}^{n+1}\right)^T \boldsymbol{f}_{\text{free}} \\
&= \Delta t \left(\boldsymbol{v}^{n+1}\right)^T \boldsymbol{f}_{\text{free}} \\
&= \Delta W \ \blacksquare
\end{aligned} \tag{61}$$

where the second passage has exploited the property $\left(\boldsymbol{M}^{-1}\right)^T = \boldsymbol{M}^{-1}$ already noted above for the diagonal mass matrix $\boldsymbol{M}$, the third passage has used the equilibrium equation $\boldsymbol{a}^{n+1} = \boldsymbol{M}^{-1}\boldsymbol{f}$ and the fourth passage has used the CD time integration scheme detailed in Section 2.2.

In order to conserve the total energy also in the presence of contact ($\boldsymbol{f_c} \neq \boldsymbol{0}$) in the current context of the central difference scheme, the kinetic energy change (including the contribution of contact forces) must be equal to the work of the forces, thus ensuring the conservation of the total energy:

$$\Delta E - \Delta W = 0 \tag{62}$$

The above conservation equation will lead, after expansion of the terms, to an equation containing the velocity discretization parameter α. Then, a suitable choice of α will allow to satisfy the equation and thus to conserve the total energy.

From the last of (57), by using the decomposition (58) of the total force, we obtain for ΔE:

$$\begin{aligned}
\Delta E &= \Delta t \left(\boldsymbol{v}^{n+1/2}\right)^T \boldsymbol{f} + \frac{1}{2}\Delta t^2 \boldsymbol{f}^T \boldsymbol{M}^{-1} \boldsymbol{f} \\
&= \Delta t \left(\boldsymbol{v}^{n+1/2}\right)^T \boldsymbol{f}_{\text{free}} + \Delta t \left(\boldsymbol{v}^{n+1/2}\right)^T \boldsymbol{f}_c + \frac{1}{2}\Delta t^2 \left(\boldsymbol{f}_{\text{free}} + \boldsymbol{f}_c\right)^T \boldsymbol{M}^{-1} \left(\boldsymbol{f}_{\text{free}} + \boldsymbol{f}_c\right) \\
&= \Delta t \left(\boldsymbol{v}^{n+1/2}\right)^T \boldsymbol{f}_{\text{free}} + \Delta t \left(\boldsymbol{v}^{n+1/2}\right)^T \boldsymbol{f}_c + \\
&\quad \frac{1}{2}\Delta t^2 \boldsymbol{f}_{\text{free}}^T \boldsymbol{M}^{-1} \boldsymbol{f}_{\text{free}} + \frac{1}{2}\Delta t^2 \boldsymbol{f}_c^T \boldsymbol{M}^{-1} \boldsymbol{f}_c + \frac{1}{2}\Delta t^2 \boldsymbol{f}_{\text{free}}^T \boldsymbol{M}^{-1} \boldsymbol{f}_c + \frac{1}{2}\Delta t^2 \boldsymbol{f}_c^T \boldsymbol{M}^{-1} \boldsymbol{f}_{\text{free}} \\
&= \Delta t \left(\boldsymbol{v}^{n+1/2}\right)^T \boldsymbol{f}_{\text{free}} + \Delta t \left(\boldsymbol{v}^{n+1/2}\right)^T \boldsymbol{f}_c + \\
&\quad \frac{1}{2}\Delta t^2 \boldsymbol{f}_{\text{free}}^T \boldsymbol{M}^{-1} \boldsymbol{f}_{\text{free}} + \frac{1}{2}\Delta t^2 \boldsymbol{f}_c^T \boldsymbol{M}^{-1} \boldsymbol{f}_c + \Delta t^2 \boldsymbol{f}_c^T \boldsymbol{M}^{-1} \boldsymbol{f}_{\text{free}}
\end{aligned} \tag{63}$$

where the identity (53) has been used in the last passage. Furthermore, from (60) we obtain:

$$\begin{aligned}
\Delta W &= \Delta t \left(\boldsymbol{v}^{n+1}\right)^T \boldsymbol{f}_{\text{free}} \\
&= \Delta t \left(\boldsymbol{v}^{n+1/2} + \frac{1}{2}\Delta t\, \boldsymbol{a}^{n+1}\right)^T \boldsymbol{f}_{\text{free}} \\
&= \Delta t \left(\boldsymbol{v}^{n+1/2} + \frac{1}{2}\Delta t\, \boldsymbol{M}^{-1} \boldsymbol{f}\right)^T \boldsymbol{f}_{\text{free}} \\
&= \Delta t \left(\boldsymbol{v}^{n+1/2} + \frac{1}{2}\Delta t\, \boldsymbol{M}^{-1} \boldsymbol{f}_{\text{free}} + \frac{1}{2}\Delta t\, \boldsymbol{M}^{-1} \boldsymbol{f}_c\right)^T \boldsymbol{f}_{\text{free}} \\
&= \Delta t \left(\boldsymbol{v}^{n+1/2}\right)^T \boldsymbol{f}_{\text{free}} + \frac{1}{2}\Delta t^2 \left(\boldsymbol{M}^{-1} \boldsymbol{f}_{\text{free}}\right)^T \boldsymbol{f}_{\text{free}} + \frac{1}{2}\Delta t^2 \left(\boldsymbol{M}^{-1} \boldsymbol{f}_c\right)^T \boldsymbol{f}_{\text{free}} \\
&= \Delta t \left(\boldsymbol{v}^{n+1/2}\right)^T \boldsymbol{f}_{\text{free}} + \frac{1}{2}\Delta t^2 \boldsymbol{f}_{\text{free}}^T \left(\boldsymbol{M}^{-1}\right)^T \boldsymbol{f}_{\text{free}} + \frac{1}{2}\Delta t^2 \boldsymbol{f}_c^T \left(\boldsymbol{M}^{-1}\right)^T \boldsymbol{f}_{\text{free}} \\
&= \Delta t \left(\boldsymbol{v}^{n+1/2}\right)^T \boldsymbol{f}_{\text{free}} + \frac{1}{2}\Delta t^2 \boldsymbol{f}_{\text{free}}^T \boldsymbol{M}^{-1} \boldsymbol{f}_{\text{free}} + \frac{1}{2}\Delta t^2 \boldsymbol{f}_c^T \boldsymbol{M}^{-1} \boldsymbol{f}_{\text{free}}
\end{aligned} \tag{64}$$

where in the expression for ΔW the following CD scheme relation and the equilibrium equation:

$$\begin{aligned}
\boldsymbol{v}^{n+1} &= \boldsymbol{v}^{n+1/2} + \frac{1}{2}\Delta t\, \boldsymbol{a}^{n+1} \\
\boldsymbol{a}^{n+1} &= \boldsymbol{M}^{-1} \boldsymbol{f}
\end{aligned} \tag{65}$$

have been used in the first two passages, eq. (58) in the third passage, the first of (50) in the fifth passage and the property $\left(\boldsymbol{M}^{-1}\right)^T = \boldsymbol{M}^{-1}$ in the last passage.

Therefore, the total variation of the energy becomes, by using again (53) for the last term:

$$\Delta E - \Delta W = \Delta t \left(\boldsymbol{v}^{n+1/2}\right)^T \boldsymbol{f}_c + \frac{1}{2}\Delta t^2 \boldsymbol{f}_c^T \boldsymbol{M}^{-1} \boldsymbol{f}_c + \frac{1}{2}\Delta t^2 \boldsymbol{f}_{\text{free}}^T \boldsymbol{M}^{-1} \boldsymbol{f}_c \tag{66}$$

Beside (preferably) ensuring the conservation of the energy, the contact forces $\boldsymbol{f}_c$ must primarily respect the impenetrability condition, expressed via the velocity constraint in eq. (42) which is parametrized in terms of α. However, it is more practical for further developments to introduce a new temporary parameter β:

$$\beta = \frac{1}{2}\left(\alpha + 1\right) \tag{67}$$

so that when α goes from 0 to 1, correspondingly β goes from 1/2 to 1. This transforms the parametrized constraint (42) into:

$$\boldsymbol{C}\boldsymbol{v}^{n+1+\alpha/2} = \boldsymbol{C}\boldsymbol{v}^{n+1/2+\beta} = \boldsymbol{0} \tag{68}$$

By expressing the parametrized velocity via the acceleration $\boldsymbol{a}^{n+1}$ and the equilibrium equation (second of 65):

$$\begin{aligned}\boldsymbol{v}^{n+1/2+\beta} &= \boldsymbol{v}^{n+1/2} + \beta\Delta t\, \boldsymbol{a}^{n+1} \\ &= \boldsymbol{v}^{n+1/2} + \beta\Delta t\, \boldsymbol{M}^{-1}\boldsymbol{f}\end{aligned} \tag{69}$$

the constraint (68) becomes:

$$\boldsymbol{C}\left(\boldsymbol{v}^{n+1/2} + \beta\Delta t\, \boldsymbol{M}^{-1}\boldsymbol{f}\right) = \boldsymbol{0} \tag{70}$$

We now apply the method of Lagrange Multipliers to solve the equilibrium equation under the constraint (70), by closely following the procedure detailed in Section 2.3.1. Since the constraint parametrization can be applied to any velocity-based constraint (i.e., not only to contacts), in place of (70) we consider a more general version of the parametrized constraint, where the RHS is a known vector $\boldsymbol{b}$, not necessarily $\boldsymbol{0}$:

$$\boxed{\boldsymbol{C}\left(\boldsymbol{v}^{n+1/2} + \beta\Delta t\, \boldsymbol{M}^{-1}\boldsymbol{f}\right) = \boldsymbol{b}} \tag{71}$$

The unknown reactions $\boldsymbol{r}$ are expressed via the unknown Lagrange Multipliers $\boldsymbol{\lambda}$, cfr. eq. (7):

$$\boldsymbol{r} = \boldsymbol{C}^T\boldsymbol{\lambda} \tag{72}$$

By letting the reactions appear explicitly, the equilibrium equation becomes:

$$\boldsymbol{M}\boldsymbol{a} = \boldsymbol{f}_{\text{free}} + \boldsymbol{r} = \boldsymbol{f}_{\text{free}} + \boldsymbol{C}^T\boldsymbol{\lambda} \tag{73}$$

Multiplying both members by $\boldsymbol{C}\boldsymbol{M}^{-1}$ and simplifying:

$$\boldsymbol{C}\boldsymbol{a} = \boldsymbol{C}\boldsymbol{M}^{-1}\boldsymbol{f}_{\text{free}} + \boldsymbol{C}\boldsymbol{M}^{-1}\boldsymbol{C}^T\boldsymbol{\lambda} \tag{74}$$

From this the Lagrange Multipliers are symbolically obtained as:

$$\boldsymbol{C}\boldsymbol{M}^{-1}\boldsymbol{C}^T\boldsymbol{\lambda} = \boldsymbol{C}\boldsymbol{a} - \boldsymbol{C}\boldsymbol{M}^{-1}\boldsymbol{f}_{\text{free}} \tag{75}$$

which, by letting:

$$\boldsymbol{B} = \boldsymbol{C}\boldsymbol{M}^{-1}\boldsymbol{C}^T \tag{76}$$

becomes:

$$\boldsymbol{B}\boldsymbol{\lambda} = \boldsymbol{C}\boldsymbol{a} - \boldsymbol{C}\boldsymbol{M}^{-1}\boldsymbol{f}_{\text{free}} \tag{77}$$

In order to actually find $\boldsymbol{\lambda}$, the term $\boldsymbol{C}\boldsymbol{a}$ must be explicited. To this end, we use the time integration scheme and the constraint. The parametrized velocity is (see the first of 69):

$$\boldsymbol{v}^{n+1/2+\beta} = \boldsymbol{v}^{n+1/2} + \beta\Delta t\, \boldsymbol{a}^{n+1} \tag{78}$$

which, replaced in the form (68) of the constraint (with $\boldsymbol{b}$ at the RHS), gives:

$$\boldsymbol{C}\boldsymbol{v}^{n+1/2+\beta} = \boldsymbol{C}\boldsymbol{v}^{n+1/2} + \beta\Delta t\, \boldsymbol{C}\boldsymbol{a} = \boldsymbol{b} \tag{79}$$

and from this we obtain:

$$\boldsymbol{C}\boldsymbol{a} = \frac{1}{\beta\Delta t}\left(\boldsymbol{b} - \boldsymbol{C}\boldsymbol{v}^{n+1/2}\right) \tag{80}$$

By replacing (80) into (77) we obtain then:

$$\boldsymbol{B}\boldsymbol{\lambda} = \frac{1}{\beta\Delta t}\left(\boldsymbol{b} - \boldsymbol{C}\boldsymbol{v}^{n+1/2}\right) - \boldsymbol{C}\boldsymbol{M}^{-1}\boldsymbol{f}_{\text{free}} \tag{81}$$

and from this we explicit $\boldsymbol{\lambda}$ by multiplying both members by $\boldsymbol{B}^{-1}$:

$$\boldsymbol{\lambda} = \frac{1}{\beta\Delta t}\boldsymbol{B}^{-1}\left(\boldsymbol{b} - \boldsymbol{C}\boldsymbol{v}^{n+1/2}\right) - \boldsymbol{B}^{-1}\boldsymbol{C}\boldsymbol{M}^{-1}\boldsymbol{f}_{\text{free}} \tag{82}$$

By replacing (82) into the expression of the reactions (72) we get:

$$\boldsymbol{r} = \boldsymbol{C}^T\boldsymbol{\lambda} = \frac{1}{\beta\Delta t}\boldsymbol{C}^T\boldsymbol{B}^{-1}\left(\boldsymbol{b} - \boldsymbol{C}\boldsymbol{v}^{n+1/2}\right) - \boldsymbol{C}^T\boldsymbol{B}^{-1}\boldsymbol{C}\boldsymbol{M}^{-1}\boldsymbol{f}_{\text{free}} \tag{83}$$

By particularizing (83) for the case of contact, i.e. by posing $\boldsymbol{b} = 0$ and $\boldsymbol{r} = \boldsymbol{f}_c$, we obtain finally:

$$\boxed{\boldsymbol{f}_c = -\frac{1}{\beta\Delta t}\boldsymbol{C}^T\boldsymbol{B}^{-1}\boldsymbol{C}\boldsymbol{v}^{n+1/2} - \boldsymbol{C}^T\boldsymbol{B}^{-1}\boldsymbol{C}\boldsymbol{M}^{-1}\boldsymbol{f}_{\text{free}}} \tag{84}$$

Now, it is convenient to introduce a new matrix $\boldsymbol{H}$ and the vector $\boldsymbol{a}_{\text{free}}$ which corresponds to the acceleration if there were no contact forces:

$$\begin{aligned} \boldsymbol{H} &= \boldsymbol{C}^T\boldsymbol{B}^{-1}\boldsymbol{C} \\ \boldsymbol{a}_{\text{free}} &= \boldsymbol{M}^{-1}\boldsymbol{f}_{\text{free}} \end{aligned} \tag{85}$$

in order to express $\boldsymbol{f}_c$ from (84) as follows:

$$\boldsymbol{f}_c = -\frac{1}{\beta\Delta t}\boldsymbol{H}\boldsymbol{v}^{n+1/2} - \boldsymbol{H}\boldsymbol{a}_{\text{free}} \tag{86}$$

We note, incidentally, that the matrices $\boldsymbol{B}$ and $\boldsymbol{H}$ are symmetric, and this important property will be used in the following.

To prove that $\boldsymbol{B}$ is symmetric (i.e. that $\boldsymbol{B}^T = \boldsymbol{B}$), we take the transpose of both members in its definition, eq. (76):

$$\begin{aligned} \boldsymbol{B}^T &= \left(\boldsymbol{C}^T\right)^T\left(\boldsymbol{M}^{-1}\right)^T\boldsymbol{C}^T \\ &= \boldsymbol{C}\left(\boldsymbol{M}^{-1}\right)^T\boldsymbol{C}^T \\ &= \boldsymbol{C}\boldsymbol{M}^{-1}\boldsymbol{C}^T \\ &= \boldsymbol{B}\ \blacksquare \end{aligned} \tag{87}$$

where in the second passage we have exploited the property (proven shortly below) that, since $\boldsymbol{M}$ is symmetric ($\boldsymbol{M}^T = \boldsymbol{M}$), then $\boldsymbol{M}^{-1}$ is also symmetric ($(\boldsymbol{M}^{-1})^T = \boldsymbol{M}^{-1}$).

Similarly, to prove that $\boldsymbol{H}$ is symmetric (i.e. that $\boldsymbol{H}^T = \boldsymbol{H}$), we take the transpose of both members in its definition, the first of eqs. (85):

$$\begin{aligned} \boldsymbol{H}^T &= \boldsymbol{C}^T\left(\boldsymbol{B}^{-1}\right)^T\left(\boldsymbol{C}^T\right)^T \\ &= \boldsymbol{C}^T\left(\boldsymbol{B}^{-1}\right)^T\boldsymbol{C} \\ &= \boldsymbol{C}^T\boldsymbol{B}^{-1}\boldsymbol{C} \\ &= \boldsymbol{H}\ \blacksquare \end{aligned} \tag{88}$$

where in the second passage we have exploited the fact that, since (87) tells us that $\boldsymbol{B}$ is symmetric, then its inverse $\boldsymbol{B}^{-1}$ is also symmetric, i.e. $\left(\boldsymbol{B}^{-1}\right)^T = \boldsymbol{B}^{-1}$, as demonstrated hereafter.

Let us now prove the property, used above, that the inverse of a (non-singular) symmetric matrix $\boldsymbol{A}$ is also symmetric. In other words, we assume that $\boldsymbol{A}^T = \boldsymbol{A}$ and that $\boldsymbol{A}^{-1}$ exists. Then we want to prove that $(\boldsymbol{A}^{-1})^T = \boldsymbol{A}^{-1}$. By definition of inverse:

$$\boldsymbol{A}\boldsymbol{A}^{-1} = \boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{I} \tag{89}$$

where $\boldsymbol{I}$ is the identity matrix, which is symmetric:

$$\boldsymbol{I} = \boldsymbol{I}^T \tag{90}$$

From the last two relations it descends that:

$$\begin{aligned} \boldsymbol{A}\boldsymbol{A}^{-1} &= \left(\boldsymbol{A}\boldsymbol{A}^{-1}\right)^T \\ &= \left(\boldsymbol{A}^{-1}\right)^T\boldsymbol{A}^T \\ &= \left(\boldsymbol{A}^{-1}\right)^T\boldsymbol{A} \end{aligned} \tag{91}$$

where in the first passage we have used the formula of the transpose of a product (50) and in the second one the symmetry of $\boldsymbol{A}$. But, because of (89) we may also write:

$$\boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{A}\boldsymbol{A}^{-1} = \left(\boldsymbol{A}^{-1}\right)^T \boldsymbol{A} \tag{92}$$

Next, we right-multiply both sides of (92) by $\boldsymbol{A}^{-1}$, obtaining:

$$\boldsymbol{A}^{-1}\boldsymbol{A}\boldsymbol{A}^{-1} = \left(\boldsymbol{A}^{-1}\right)^T \boldsymbol{A}\boldsymbol{A}^{-1} \tag{93}$$

which is equivalent to:

$$\boldsymbol{A}^{-1} = \left(\boldsymbol{A}^{-1}\right)^T \blacksquare \tag{94}$$

We now introduce the following scalars h_v, h_a and h_{va}:

$$h_v = \left(\boldsymbol{v}^{n+1/2}\right)^T \boldsymbol{H}\boldsymbol{v}^{n+1/2} \qquad h_a = \boldsymbol{a}_{\text{free}}^T \boldsymbol{H}\boldsymbol{a}_{\text{free}} \qquad h_{va} = \left(\boldsymbol{v}^{n+1/2}\right)^T \boldsymbol{H}\boldsymbol{a}_{\text{free}} \tag{95}$$

and use them to obtain equivalent expressions of the three terms appearing in the formula of the energy jump $\Delta E - \Delta W$, eq. (66).

The first term is, by using the expression (86) of $\boldsymbol{f}_c$:

$$\begin{aligned}\left(\boldsymbol{v}^{n+1/2}\right)^T \boldsymbol{f}_c &= -\frac{1}{\beta\Delta t}\left(\boldsymbol{v}^{n+1/2}\right)^T \boldsymbol{H}\boldsymbol{v}^{n+1/2} - \left(\boldsymbol{v}^{n+1/2}\right)^T \boldsymbol{H}\boldsymbol{a}_{\text{free}} \\ &= -\frac{1}{\beta\Delta t}h_v - h_{va}\end{aligned} \tag{96}$$

The second term, by using the expression (86) of $\boldsymbol{f}_c$, becomes:

$$\begin{aligned}\boldsymbol{f}_c^T \boldsymbol{M}^{-1}\boldsymbol{f}_c &= \left(-\frac{1}{\beta\Delta t}\boldsymbol{H}\boldsymbol{v}^{n+1/2} - \boldsymbol{H}\boldsymbol{a}_{\text{free}}\right)^T \boldsymbol{M}^{-1}\left(-\frac{1}{\beta\Delta t}\boldsymbol{H}\boldsymbol{v}^{n+1/2} - \boldsymbol{H}\boldsymbol{a}_{\text{free}}\right) \\ &= \left[-\frac{1}{\beta\Delta t}\left(\boldsymbol{v}^{n+1/2}\right)^T \boldsymbol{H}^T - \boldsymbol{a}_{\text{free}}^T \boldsymbol{H}^T\right]\boldsymbol{M}^{-1}\left(-\frac{1}{\beta\Delta t}\boldsymbol{H}\boldsymbol{v}^{n+1/2} - \boldsymbol{H}\boldsymbol{a}_{\text{free}}\right) \\ &= \frac{1}{\beta^2\Delta t^2}\left(\boldsymbol{v}^{n+1/2}\right)^T \boldsymbol{H}^T\boldsymbol{M}^{-1}\boldsymbol{H}\boldsymbol{v}^{n+1/2} + \frac{1}{\beta\Delta t}\left(\boldsymbol{v}^{n+1/2}\right)^T \boldsymbol{H}^T\boldsymbol{M}^{-1}\boldsymbol{H}\boldsymbol{a}_{\text{free}} + \\ &\quad \frac{1}{\beta\Delta t}\boldsymbol{a}_{\text{free}}^T \boldsymbol{H}^T\boldsymbol{M}^{-1}\boldsymbol{H}\boldsymbol{v}^{n+1/2} + \boldsymbol{a}_{\text{free}}^T \boldsymbol{H}^T\boldsymbol{M}^{-1}\boldsymbol{H}\boldsymbol{a}_{\text{free}} \\ &= \frac{1}{\beta^2\Delta t^2}\left(\boldsymbol{v}^{n+1/2}\right)^T \boldsymbol{H}\boldsymbol{v}^{n+1/2} + \frac{1}{\beta\Delta t}\left(\boldsymbol{v}^{n+1/2}\right)^T \boldsymbol{H}\boldsymbol{a}_{\text{free}} + \\ &\quad \frac{1}{\beta\Delta t}\boldsymbol{a}_{\text{free}}^T \boldsymbol{H}\boldsymbol{v}^{n+1/2} + \boldsymbol{a}_{\text{free}}^T \boldsymbol{H}\boldsymbol{a}_{\text{free}} \\ &= \frac{1}{\beta^2\Delta t^2}\left(\boldsymbol{v}^{n+1/2}\right)^T \boldsymbol{H}\boldsymbol{v}^{n+1/2} + \frac{1}{\beta\Delta t}\left(\boldsymbol{v}^{n+1/2}\right)^T \boldsymbol{H}\boldsymbol{a}_{\text{free}} + \\ &\quad \frac{1}{\beta\Delta t}\left(\boldsymbol{v}^{n+1/2}\right)^T \boldsymbol{H}\boldsymbol{a}_{\text{free}} + \boldsymbol{a}_{\text{free}}^T \boldsymbol{H}\boldsymbol{a}_{\text{free}} \\ &= \frac{1}{\beta^2\Delta t^2}h_v + h_a + \frac{2}{\beta\Delta t}h_{va}\end{aligned} \tag{97}$$

where in the first passage we have used the transpose of a product formula (first of eqs. 50), in the third passage we have used the identity:

$$\begin{aligned}\boldsymbol{H}\boldsymbol{M}^{-1}\boldsymbol{H} &= \boldsymbol{C}^T\boldsymbol{B}^{-1}\boldsymbol{C}\boldsymbol{M}^{-1}\boldsymbol{C}^T\boldsymbol{B}^{-1}\boldsymbol{C} \\ &= \boldsymbol{C}^T\boldsymbol{B}^{-1}\boldsymbol{B}\boldsymbol{B}^{-1}\boldsymbol{C} \\ &= \boldsymbol{H}\end{aligned} \tag{98}$$

and in the fourth passage we have used the identity $\boldsymbol{a}_{\text{free}}^T \boldsymbol{H}\boldsymbol{v}^{n+1/2} = \left(\boldsymbol{v}^{n+1/2}\right)^T \boldsymbol{H}\boldsymbol{a}_{\text{free}}$ which holds thanks to eq. (53), since $\boldsymbol{v}^{n+1/2}$ and $\boldsymbol{a}_{\text{free}}$ are column vectors and matrix $\boldsymbol{H}$ is symmetric, see (88).

The third term, by using the definition of the free acceleration from the second of (85) and the expression (86) of $\boldsymbol{f}_c$, becomes:

$$\begin{aligned}
\boldsymbol{f}_{\text{free}}^T \boldsymbol{M}^{-1} \boldsymbol{f}_c &= (\boldsymbol{M}\boldsymbol{a}_{\text{free}})^T \boldsymbol{M}^{-1} \left(-\frac{1}{\beta \Delta t} \boldsymbol{H}\boldsymbol{v}^{n+1/2} - \boldsymbol{H}\boldsymbol{a}_{\text{free}} \right) \\
&= \boldsymbol{a}_{\text{free}}^T \boldsymbol{M}^T \boldsymbol{M}^{-1} \left(-\frac{1}{\beta \Delta t} \boldsymbol{H}\boldsymbol{v}^{n+1/2} - \boldsymbol{H}\boldsymbol{a}_{\text{free}} \right) \\
&= -\frac{1}{\beta \Delta t} \boldsymbol{a}_{\text{free}}^T \boldsymbol{M}^T \boldsymbol{M}^{-1} \boldsymbol{H}\boldsymbol{v}^{n+1/2} - \boldsymbol{a}_{\text{free}}^T \boldsymbol{M}^T \boldsymbol{M}^{-1} \boldsymbol{H}\boldsymbol{a}_{\text{free}} \\
&= -\frac{1}{\beta \Delta t} \boldsymbol{a}_{\text{free}}^T \boldsymbol{H}\boldsymbol{v}^{n+1/2} - \boldsymbol{a}_{\text{free}}^T \boldsymbol{H}\boldsymbol{a}_{\text{free}} \\
&= -\frac{1}{\beta \Delta t} \left(\boldsymbol{v}^{n+1/2} \right)^T \boldsymbol{H}\boldsymbol{a}_{\text{free}} - \boldsymbol{a}_{\text{free}}^T \boldsymbol{H}\boldsymbol{a}_{\text{free}} \\
&= -\frac{1}{\beta \Delta t} h_{va} - h_a
\end{aligned} \tag{99}$$

where in the third passage we have exploited the fact that, since $\boldsymbol{M}$ is symmetric, $\boldsymbol{M}^T = \boldsymbol{M}$, so that $\boldsymbol{M}^T \boldsymbol{M}^{-1} = \boldsymbol{M}\boldsymbol{M}^{-1} = \boldsymbol{I}$. Furthermore, in the fourth passage we have used the identity $\boldsymbol{a}_{\text{free}}^T \boldsymbol{H}\boldsymbol{v}^{n+1/2} = \left(\boldsymbol{v}^{n+1/2}\right)^T \boldsymbol{H}\boldsymbol{a}_{\text{free}}$, which holds thanks to eq. (53), since $\boldsymbol{v}^{n+1/2}$ and $\boldsymbol{a}_{\text{free}}$ are column vectors and $\boldsymbol{H}$ is a symmetric matrix, see (88).

Finally, by replacing the three terms (96), (97) and (99) the expression of the total energy jump (66) becomes:

$$\begin{aligned}
\Delta E - \Delta W &= \left(\frac{1}{2\beta} - 1 \right) \frac{1}{\beta} h_v - \left(1 - \frac{1}{\beta} + \frac{1}{2\beta} \right) \Delta t \, h_{va} \\
&= \left(\frac{1}{2\beta} - 1 \right) \frac{1}{\beta} h_v - \left(1 - \frac{1}{2\beta} \right) \Delta t \, h_{va} \\
&= -\left(1 - \frac{1}{2\beta} \right) \frac{1}{\beta} h_v - \left(1 - \frac{1}{2\beta} \right) \Delta t \, h_{va} \\
&= \left(\frac{1}{2\beta} - 1 \right) \left(\frac{1}{\beta} h_v + \Delta t \, h_{va} \right)
\end{aligned} \tag{100}$$

It is worth stressing that, in the above equation, the sole unknown is the parameter β, whereas h_v and h_{va} are (known) scalars depending on the intermediate velocity $\boldsymbol{v}^{n+1/2}$ and on the free acceleration $\boldsymbol{a}_{\text{free}}$, i.e. on the internal and external forces at the current time instant $\boldsymbol{f}_e$ and $\boldsymbol{f}_i$. By replacing the β parameter with the initially introduced α parameter through eq. (67), the energy jump expression becomes:

$$\boxed{\Delta E - \Delta W = \left(\frac{1}{\alpha + 1} - 1 \right) \left(\frac{2}{\alpha + 1} h_v + \Delta t \, h_{va} \right)} \tag{101}$$

It is important to note that, according to this equation, the zero energy jump, i.e. total energy conservation, can be achieved simply by setting to zero the first factor in (101), i.e. by choosing $\alpha = 0$ (full-step constraint), and this independently from the values assumed by h_v and h_{va}.

Since the eq. (101) is quadratic, indeed a second value of α exists which causes $\Delta E - \Delta W = 0$. This is obtained by setting to zero the second factor, which gives:

$$\alpha = -\left(1 - \frac{2 h_v}{\Delta t \, h_{va}} \right) \tag{102}$$

However, recall that only the values between 0 and 1 are allowed for the parameter ($0 \leq \alpha \leq 1$). Substituting this constraint into (102) we obtain the following inequality:

$$-2 \leq \frac{2 h_v}{\Delta t \, h_{va}} \leq -1 \tag{103}$$

But, since it is not possible to determine a general relation between the scalars h_v, h_{va} and the time step Δt, the second energy-conserving solution does not seem to be usable in actual practice.

Instead, by taking $\alpha = 1$ the energy jump from (101) is:

$$\Delta E - \Delta W = -\frac{1}{2}\left(h_v + \Delta t\, h_{va}\right) \tag{104}$$

So, with $\alpha = 1$ the energy jump is negative ($\Delta E - \Delta W < 0$), if $h_v + \Delta t\, h_{va} > 0$. It is expected that the latter condition be almost always satisfied, implying that the external and internal forces cannot create accelerations that could produce velocities of a magnitude close to the one of the initial impact velocity within a single time step increment. In the (unlikely) case that $h_v + \Delta t\, h_{va} < 0$, the energy jump would be positive, and this could eventually lead to stability issues.

It is important to note that the the choice of the parameter $\alpha = 0$ *vs.* $\alpha = 1$, induces a big difference with respect to the management of the separation of the surfaces in contact. With $\alpha = 1$ (half-step constraint), the contact is maintained until the link forces become negative, whereby contact is broken by a suitable rebound algorithm, since continuing it would be contrary to physics for non-cohesive interactions.

Instead, with $\alpha = 0$ (full-step constraint) the contact is maintained only until the next discrete time instant, i.e. the degrees of freedom in interaction are always immediately separated. This means that in a typical situation where the contact is (physically) supposed to be maintained for a certain period of time, the algorithm with $\alpha = 0$ will be continuously switching between contact and non-contact conditions, causing strong oscillations in the contact force. On the other hand, the algorithm with $\alpha = 0$ always conserves the momentum and the total energy of the system, which is not the case for the algorithm with $\alpha \neq 0$. In principle, α could be equal to any value between 0 and 1, but the physical meaning is only clear for $\alpha = 0$ and $\alpha = 1$.

The algorithm with $\alpha = 0$ (full-step constraint) could also be seen as a penalty method with an infinite contact stiffness, but without penalizing the critical time step.

2.3.9 Summary of the obtained expressions

For all the strategies that have been outlined in the previous Sections, the application of the Lagrange multipliers method involves the following common steps:

1. First, the matrix of connections is computed:

$$\boxed{\boldsymbol{B}^* = \boldsymbol{C}\boldsymbol{m}^{-1}\boldsymbol{C}^T} \tag{105}$$

2. Then, the right-hand side term is evaluated by using the appropriate expression for $\boldsymbol{S}$ (see below):

$$\boxed{\boldsymbol{W} = \boldsymbol{S} - \boldsymbol{C}\boldsymbol{m}^{-1}\left(\boldsymbol{f}_{\mathrm{e}} - \boldsymbol{f}_{\mathrm{i}}\right)} \tag{106}$$

3. Next, the linear system is solved for the Lagrange multipliers:

$$\boxed{\boldsymbol{\lambda} = \left(\boldsymbol{B}^*\right)^{-1}\boldsymbol{W}} \tag{107}$$

4. Finally, the reactions are computed:

$$\boxed{\boldsymbol{r} = \boldsymbol{C}^T\boldsymbol{\lambda}} \tag{108}$$

The only expression that varies in the different formulations is that of the $\boldsymbol{S}$ vector used in (106) to build the right-hand side $\boldsymbol{W}$. It is summarized in the following Table for the various cases considered in the previous Sections.

Constraint on	Constraint equation	Expression of $\boldsymbol{S}$
Acceleration	$\boldsymbol{C}^{n+1}\boldsymbol{a}^{n+1} = \boldsymbol{b}^{n+1}$	$\boxed{\boldsymbol{S} = \boldsymbol{b}}$
Full-step velocity	$\boldsymbol{C}^{n+1}\boldsymbol{v}^{n+1} = \boldsymbol{b}^{n+1}$	$\boxed{\boldsymbol{S} = \frac{2}{\Delta t}\left(\boldsymbol{b} - \boldsymbol{C}\boldsymbol{v}^{n+1/2}\right)}$
Mid-step velocity	$\boldsymbol{C}^{n+1}\boldsymbol{v}^{n+3/2} = \boldsymbol{b}^{n+1}$	$\boxed{\boldsymbol{S} = \frac{2}{\Delta t^n + \Delta t^{n+1}}\left(\boldsymbol{b} - \boldsymbol{C}\boldsymbol{v}^{n+1/2}\right)}$
Parametrized velocity	$\boldsymbol{C}^{n+1}\boldsymbol{v}^{n+1+\alpha/2} = \boldsymbol{b}^{n+1}$	$\boxed{\boldsymbol{S} = \frac{2}{\Delta t^n + \alpha\Delta t^{n+1}}\left(\boldsymbol{b} - \boldsymbol{C}\boldsymbol{v}^{n+1/2}\right)}$
Displacement	$\boldsymbol{C}^{n+1}\boldsymbol{d}^{n+2} = \boldsymbol{b}^{n+1}$	$\boxed{\boldsymbol{S} = \frac{2}{\Delta t^{n+1}\left(\Delta t^n + \Delta t^{n+1}\right)}\left(\boldsymbol{b} - \boldsymbol{C}\boldsymbol{d} - \Delta t^{n+1}\boldsymbol{C}\boldsymbol{v}^{n+1/2}\right)}$

Table 1: Summary of analytical expressions.

3 Implementation notes

First, the input syntax of the LIAJ option has been modified by adding the new optional parameter α introduced in Section 2.3.7. Then, we briefly present the code flowcharts and the most important changes to implement the developments described in the previous Sections.

3.1 New syntax of the LIAJ option

The option LIAJ has been modified as follows (additions are shown in red):

```
OPTI ...
< LIAJ <ALPH alph> >
```

with the following descriptioon:

> This option causes all constraints (imposed either via LIAI on via LINK COUP) on velocities to be expressed on the velocity at time $n+1+\alpha/2$, rather than at time $n+3/2$, which is the default in EUROPLEXUS (note that in this notation the current configuration is indicated by $n+1$). If the ALPH sub-keyword is omitted, the code assumes $\alpha = 1.0$ and so the constraints are expressed at the next mid-step (time $n+3/2$). If α is specified, it must be $0.0 \leq \alpha \leq 1.0$. The value $\alpha = 0.0$, corresponding to full-step velocity constraints (at time $n+1$), was used for example in PLEXIS-3C. Therefore, this option is mainly useful in order to perform fine-grain comparisons between results of PLEXIS-3C and EUROPLEXUS, for debugging purposes. Specifying intermediate values of α may be useful to fine-tune the properties of some types of constraints, e.g. contacts or impacts, while the value of α may be irrelevant for other types of constraints (e.g. blockages).

3.2 The LIAI directive

The code flowchart for the LIAI directive (old "liaisons" model) is presented in Figure 1. The COMPUTE_STEP routine contains the logic to perform one time step of the simulation. Its sibling routine COMPUTE_STEP_PART routine is used instead of COMPUTE_STEP when a model with spatial partitioning (PART) is used, i.e. a model in which the time step varies in space over the model, besides varying in time. The partitioning model is described in reference [7].

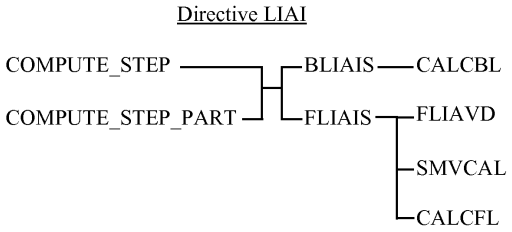


Figure 1: Code flowchart for the LIAI directive.

Both these routines call the two main routines dealing with the constraints: BLIAIS builds the $\boldsymbol{B}^*$ matrix (matrix of connections) by calling an ancillary routine CALCBL, while FLIAIS computes the reactions, by relying upon three ancillary routines. The FLIAVD routine is used for the imposed motion constraints, i.e. imposed time-dependent displacements (DEPL), velocities (VITE) or accelerations (ACCE). The SMVCAL routine transforms velocity-based constraints into their acceleration-based form, according to Section 2.3.6.

It should be noted that at the moment of this writing the LIAI model is only compatible with the spatial partitioning if all the linked nodes are placed in the lowest level of the partition (which is only possible for *permanent* constraints). In other words, the PLIN option [1] which allows using different time increments for the linked nodes cannot be activated with LIAI and can only be used with the more modern LINK COUP model to be described next.

The changes in the source code are detailed in the next paragraphs.

3.2.1 Include COPT.INC

The real VOPTIS previously free slot is re-used to store the ALPHA_LIAJ parameter, corresponding to α as specified in the `OPTI LIAJ <ALPH alph>` directive:

```
 . . .
      COMMON /COPTIS/ ZBAR,FQ95,TIONOR,angfsa,holeax,tolcno,dtdrop,
     >                tfrom,tupto,alpha_liaj,
     >                kfscr,KDPMA,KDPGR,IOSTEP,IEDSS,largcd,lsgdio,
     >                kpxlog,k2fibe,k2mesh,kfcube,kmtran,kdtml,
     >                lscoor,lselem,lsgibi,lsgril,lsepai,lsnorm,
     >                kvcont,liajrc,k2chel,render_safe,render_dump,
     >                new_fl34,kdtbe,render_stat,manfsr,flsc8,
     >                prgr
C
      REAL(8) :: ZBAR,FQ95,TIONOR,angfsa,holeax,tolcno,dtdrop,tfrom,
     &           tupto,alpha_liaj
 . . .
C           alpha_liaj: il faut que 0. <= alpha_liaj <= 1.
C                       alpha_liaj = 0. --> liaisons en vitesse a (n+1)
C                                           (OPTI LIAJ sans ALPH)
C                       alpha_liaj = 1. --> liaisons en vitesse a (n+3/2)
C                                           (default, donc sans OPTI LIAJ)
C                       donc pour alpha quelconque (OPTI LIAJ ALPH alph),
C                       liaisons en vitesse a (n+1+alpha/2)
 . . .
```

3.2.2 Subroutine RAZOPT

In the subroutine RAZOPT we add the initialization of ALPHA_LIAJ at its default value ($\alpha = 1.0$):

```
 . . .
*
*---     commun  coptis*
      CALL DRAZ(10,ZBAR)
      ALPHA_LIAJ = 1.D0 ! Liaisons en vitesse a (n+3/2) par defaut
      CALL IRAZ(30,KFSCR)
      CALL DESTROY_LINKS_FS
 . . .
```

3.2.3 Subroutine OPTION

In the subroutine OPTION we add the reading of the new optional `ALPH` sub-keyword of the `LIAJ` option:

```
 . . .
      INTEGER, PARAMETER :: NMOT=155,  NDECE=3,  NIMPO=2,   NMADDF=11,
 . . .
     >                      NMDOMD=5,  NMRZEL=1, NMRIGB=4,  NMLIAJ=1
 . . .
      CHARACTER(4) :: CDECE(NDECE),   CIMPO(NIMPO),   MADDF(NMADDF),
 . . .
     >                MORIGB(NMRIGB), MOLIAJ(NMLIAJ)
 . . .
      DATA MOLIAJ /'ALPH'/
 . . .
*---- "liaj" : liaisons concern velocities at (n+1+alpha/2), not at (n+3/2)
      CASE (111)
        LIAJRC = 1 ! On a lu l'option LIAJ (eventuellement suivie par ALPH)
        ALPHA_LIAJ = 0.D0 ! Si LIAJ sans ALPH (ancienne syntaxe), ALPHA=0
                          ! donc liaisons en vitesse a (n+1)
  111   CONTINUE
```

```
      CALL LIRMOT (MOLIAJ, NMLIAJ, KOP) ! On cherche a lire le mot-cle ALPH
      SELECT CASE (KOP)
      CASE (1) ! ALPH
        CALL LIRE (3) ! On cherche a lire la valeur de alpha
        ALPHA_LIAJ = DPREC
        IF (ALPHA_LIAJ < 0.D0 .OR. ALPHA_LIAJ > 1.D0) THEN
          CALL ERRMSS ('OPTION',
     >      'INVALID ALPH VALUE FOR LIAJ (MUST BE 0<=ALPH<=1)')
          CALL EPX_STOP
     >    ('OPTION :INVALID ALPH VALUE FOR LIAJ (MUST BE 0<=ALPH<=1)')
        ENDIF
      CASE DEFAULT
        GOTO 1
      END SELECT
      GO TO 111
 . . .
```

3.2.4 Subroutine IMPOPT

In the subroutine IMPOPT we improve the printout of the `LIAJ` option and add the printout of the coefficient ALPHA_LIAJ:

```
 . . .
*----- options for liaisons
      IF (LIAJRC.EQ.1) THEN ! Mot-cle LIAJ, eventuellement suivi par ALPH
        WRITE(TAPOUT,1512) ALPHA_LIAJ
      ENDIF
 . . .
 1512 FORMAT(/,T10,
     >      'LIAI/LINK EN VITESSES A (N+1+ALPHA/2), PAS A (N+3/2)',/,
     >      'ALPHA=',1PD12.5)
 . . .
 1512 FORMAT(/,T10,
     >      'LIAI/LINK ON VELOCITIES AT (N+1+ALPHA/2), NOT (N+3/2)',/,
     >      'ALPHA=',1PD12.5)
 . . .
```

3.2.5 Subroutine FLIAIS

The old and new relevant parts of the subroutine FLIAIS are shown side-by-side to highlight the changes:

Old code

```
 . . .
      REAL(8) :: AUX, SMDTV
 . . .
      IF (LIAJRC == 1) THEN
* la liaison porte sur les vitesses au pas (n+1)
        SMDTV = 2.D0 / DT1
      ELSE
* par defaut la liaison porte sur les vitesses au pas (n+3/2)
        IF (IFLAG2 == 1) THEN
* cas de la collision et des chocs entre structures articulees composees
* de corps rigides : il y a choc
          SMDTV = 1.D0
        ELSE
* cas normal
          IF (NPAS == 0) THEN
            SMDTV = 2.D0 / DT2
          ELSE
            SMDTV = 2.D0 / (DT1 + DT2)
          ENDIF
        ENDIF
      ENDIF
 . . .
```

New code

```
 . . .
      REAL(8) :: AUX, SMDTV, ALPH
 . . .
      IF (NPAS == 0) THEN
        ALPH = 1.D0 ! Au pas 0, on ecrit les liaison en vitesse
                    ! toujours sur (n+3/2) quelque soit la valeur
                    ! ALPHA_LIAJ dans l'input
      ELSE
        ALPH = ALPHA_LIAJ
      ENDIF
*
      IF (IFLAG2 == 1) THEN ! On ne tient pas compte de OPTI LIAJ eventuel
* cas de la collision et des chocs entre structures articulees composees
* de corps rigides : il y a choc
        SMDTV = 1.D0
      ELSE ! Cas normal (il n'y a pas de collisions)
!fc on n'a pas besoin de tester LIAJRC, il suffit d'utiliser ALPHA_LIAJ
!fc la liaison porte sur les vitesses au pas (n+1+alpha/2):
!fc    si OPTI LIAJ n'a pas ete specifie, par defaut ALPHA_LIAJ=1.0,
!fc    ce qui correspond a ecrire les liaisons en vitesse a (n+3/2)
        IF (NPAS == 0) THEN
          SMDTV = 2.D0 / (ALPH*DT2)
        ELSE
          SMDTV = 2.D0 / (DT1 + ALPH*DT2)
        ENDIF
      ENDIF
 . . .
```

First, a local value of α (ALPH) is computed. This is equal to the α value given in input in the `LIAJ` option (ALPHA_LIAJ) except at step 0, where we take $\alpha = 1.0$. The effect of this is that, as already mentioned, at step 0 ($n = -1$) any velocity-based constraints are always expressed at time station $n + 1 + 1/2 = 1/2$, corresponding to time $t^{1/2} = t^0 + \Delta t^1/2$ (mid-step), irrespective of the α value set in input, instead of time station $n + 1 + \alpha/2$ or time $t^0 + \alpha\Delta t^1/2$.

This is, firstly, to avoid a pathological situation (division by zero) if one chooses $\alpha = 0$ (`LIAJ` without `ALPH`). This is a consequence of the fact that the initial velocity $\boldsymbol{v}^0$ is set in the input and must not be modified. The second reason is to avoid potentially very large accelerations due to constraints at the zero step, which could arise for very small (albeit not zero) values of α. So, as already mentioned, the actual value of α given in input starts being used only from step 1 on.

The test on IFLAG2 has been moved outside the test on LIAJRC. In the old code, the case of IFLAG2 equal to 1 was only treated when the LIAJ option was not specified, while now it is always treated and in any case leads to setting SMDTV to 1. This case is for an old model of collisions between rigid articulated bodies which is not documented and in practice never used in EPX and it is kept only to let old test cases to run.

In the normal case (IFLAG2 equal 0), the definition of SMDTV is modified by replacing DT2 with ALPH*DT2. This causes velocity-based links to be expressed at the parametrized time $t^{n+1} + \alpha\Delta_t^{n+1}/2$.

The value of SMDTV is eventually passed to subroutine SMVCAL which transforms any velocity-based constraint into its equivalent acceleration-based form before forming and solving the linear system:

```
 . . .
        CALL SMVCAL (SNDMBR(IR), NBPLI(1,IR), LPAIR(IAD), COEFL(IAD),
     >               V, SMDTV)
 . . .
```

A notable exception is the case of imposed velocities (`VITE`), which are not treated in SMVCAL but rather in subroutine FLIAVD, described in the next paragraph:

```
 . . .
      IF (NBIMPO > 0) THEN
* cas des accelerations, vitesses et deplacements imposes
        CALL FLIAVD (SNDMBR, G, V, DFX)
      ENDIF
 . . .
```

Note that FLIAVD treats all imposed motions, i.e. not only imposed velocities, but also imposed displacements (`DEPL`) and accelerations (`ACCE`). However, the velocity parametrization only concerns the imposed velocities. The imposed accelerations are always expressed at the current time station $n + 1$ while the imposed displacements are always expressed at the next time station $n + 2$.

3.2.6 Subroutine FLIAVD

The subroutine FLIAVD treats imposed motion constraints. The subroutine text is listed below. The formulation used is rather obscure, therefore for the moment this routine is not modified in order to take into account the possible presence of the `ALPH` keyword in the input.

```
Old code
      SUBROUTINE FLIAVD(SM2,G,V,DFX)
*
*     ----------------------------------------------------------------
*
*         calcul du 2eme membre pour les acce.,vit.,depl. imposes
*                                                        h. bung 01-86
*     ----------------------------------------------------------------
*
*
      USE M_OLIAISONS_DATA


      INCLUDE 'NONE.INC'
*
      INCLUDE 'COPT.INC'
      INCLUDE 'LIESON.INC'
      INCLUDE 'TEMPS.INC'
      INCLUDE 'TEMPX.INC'
*
*     organisation de <fcoef>  : dimension  fcoef(3,mxfcoe)
*
*        fcoef(1,ifco) : valeur du coefficient (valco)
*        fcoef(2,ifco) : ifonc  no de la fonction f(t)
*        fcoef(3,ifco) : valeur pour t = 0  (val0)
*
*     val = val0 + valco*fonction(t)
*
*     wp1 = valeur de la fonction ifonc au temps t
*     wp2 = valeur de la fonction ifonc au temps t+dt2
*
*     ity = 10    : acceleration imposee
*     ity = 11    : vitesse      imposee
*     ity = 12    : deplacement  imposee
*     ity = 13    : element d.r.i.t.   fait dans blsoli
*
*
      REAL(8), INTENT(IN) :: G(*),V(*),DFX(*)
      REAL(8), INTENT(OUT) :: SM2(*)
*
      INTEGER :: IR,ITY,KAD,IPOS,IFONC,IFCO,ITY1
      REAL(8) :: FVAL1,FVAL2,FVALE,SMDTV,SMDTD,AUX,WP1,WP2
*
*
*-----      le 2 eme membre des relations
*
      IF (LIAJRC.EQ.1) THEN
*
*-----   la liaison porte sur les vitesses au pas (n+1)
        SMDTV = 2D0 / DT1
        AUX   = DT2*(DT1+DT2)
        SMDTD = 2D0 /  AUX
      ELSE
*
*-----   par defaut la liaison porte sur les vitesses au pas (n+3/2)
        IF(NPAS.NE.0) THEN
          SMDTV = 2D0 / DT1
          AUX=DT2*(DT1+DT2)
          SMDTD = 2D0 / AUX
        ELSE
          SMDTV = 2.D0 / DT2
          SMDTD = 1.D0 / (DT2*DT2)
        ENDIF
      ENDIF
*
*
      DO IR=1,NLIE
        ITY=NPLIE(1,IR)
*
        SELECT CASE (ITY)
*
        CASE (10:13)
          IFCO = NPLIE(4,IR)
          IFONC = NINT(FCOEF(2,IFCO))
*-----       calcul des fonctions au temps t et t+dt2
          CALL CALC_FONCTION(IFONC,T    ,WP1)
          CALL CALC_FONCTION(IFONC,T+DT2,WP2)
*
          FVAL1 = WP1*FCOEF(1,IFCO) + FCOEF(3,IFCO)
          FVAL2 = WP2*FCOEF(1,IFCO) + FCOEF(3,IFCO)
*cccccccc      dfsdt = (fval2 - fval1)*smdtv*0.5d0
*
          KAD=NBPLI(2,IR)
          IPOS=LPAIR(KAD)
!---        Traitement spÃ©cifiques du 2 membre poutr ity=10,11,12
          ITY1=ITY
        SELECT CASE(ITY1)

        CASE (10)
*-----        acceleration  impose
            SM2(IR) = FVAL1

        CASE (11)
*-----        vitesse       imposee
            FVALE = FVAL1
            IF(NPAS == 0) FVALE = 0.5D0*(FVAL1+FVAL2)
            SM2(IR) = (FVALE - V(IPOS)) * SMDTV

        CASE (12)
*-----        deplacement   impose
            IF(NPAS /= 0) THEN
              SM2(IR) = (FVAL2 - DFX(IPOS) - V(IPOS)*DT2) * SMDTD
            ELSE
              SM2(IR) = 2.D0*SMDTD*FVAL2  - V(IPOS)*SMDTV
            ENDIF
          END SELECT
*
*   sinon on ne fait rien
*   ---------------------
      CASE DEFAULT
        CYCLE
      END SELECT
*
*
      END DO
*
      END SUBROUTINE FLIAVD
```

3.3 The `LINK COUP` directive

The code flowchart for the `LINK COUP` directive (new coupled "links" model) is presented in Figure 2. The CALCFL routine computes the right-hand side of the constraints. The flowchart is similar to that of the `LIAI` case but slightly more complicated due to the data structure and data organization used in the `LINK COUP` model. The COMPUTE_STEP and COMPUTE_STEP_PART routines already described for the `LIAI` case call either FE_TRUE_LIAIS or FE_LINKS (in the case of COMPUTE_STEP through an intermediate routine CALCUL_LINK contained in the CALCUL source file), both in the M_LINKS module, depending upon the organization of the links.

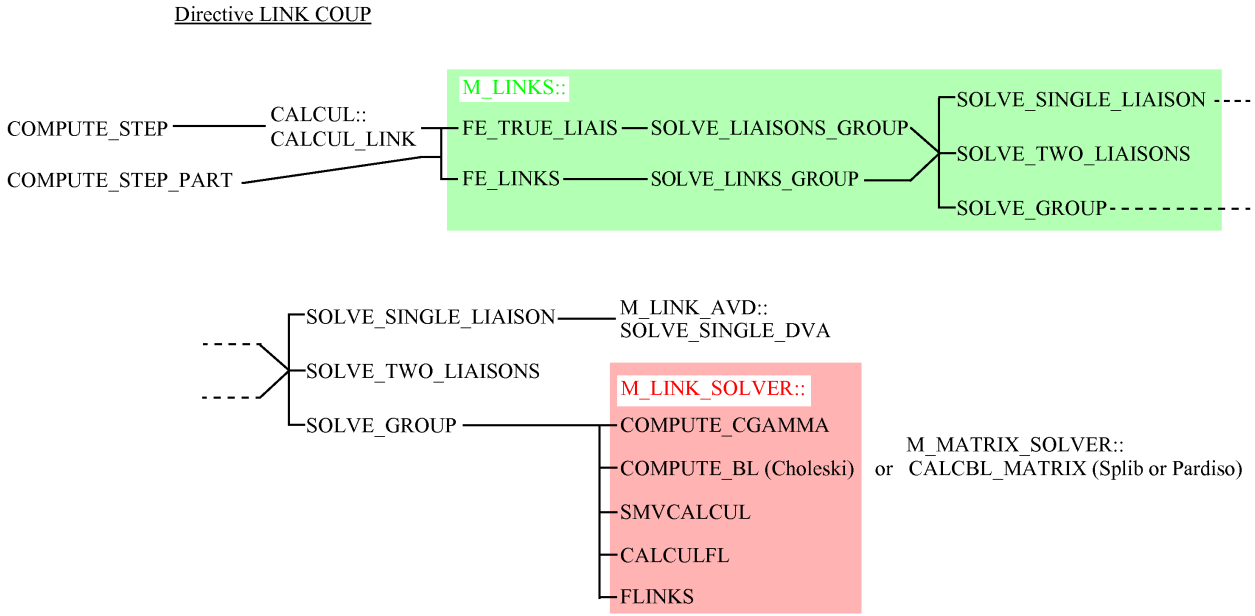


Figure 2: Code flowchart for the `LINK COUP` directive.

The first routine (FE_TRUE_LIAIS) is called if the constraints are stored in a (doubly-)linked list form, while the second (FE_LINKS) is called when the constraints are stored in a dynamic array. The first type of data structure is extremely powerful since it allows almost arbitrary manipulations of the constraints within the list, but it is computationally expensive to use, since addressing a particular link requires traversing the list. The second organization is less flexible as concerns the manipulation of constraints but much more efficient computationally, since any constraint in the array can be addressed directly simply by its index.

By default, the set of links is split into a number of mutually independent sub-sets (or "groups" of links) (unless the `SPLT NONE` keyword is specified after `LINK COUP`). The degrees of freedom (dofs) in each group are disjoint, i.e. they appear only in that group. Then, instead of solving a monolithic linear system (like in the case of the `LIAI` model), each group is solved separately. This is accomplished by calling three possible routines: SOLVE_SINGLE_LIAISON deals with a group containing a single link, which is solved analytically (no need to form a matrix and to invert it). The SOLVE_TWO_LIAISONS routine deals with groups containing two links, and again the solution is analytical (although much more involved than for Finally, SOLVE_GROUP is involved if the group contains more than two links and in that case a numerical solution similar to that of the `LIAI` model is carried out.

However, it should be noted that in order to activate the detection of two-constraint groups and their solution through SOLVE_TWO_LIAISONS, the user must activate an optional keyword `SOL2`, see [1]. In fact, by default, EPX uses SOLVE_GROUP also for groups containing only two links. All these routines are contained in the M_LINKS module.

The numerical solution is obtained in SOLVE_GROUP by calling lower-level routines (from module M_LINK_SOLVER) which are roughly similar to those of the `LIAI` model shown in Figure 1: COMPUTE_BL is similar to CALCBL, SMVCALCUL is similar to SMVCAL, CALCULFL is similar to CALCFL and FLINKS is similar to the last part of FLIAIS.

An additional routine COMPUTE_CGAMMA is called at the beginning of SOLVE_GROUP. Roughly, this routine computes the term $\boldsymbol{C}\gamma$, where $\boldsymbol{C}$ is the matrix of constraint coefficients (left-hand sides) and γ is typically a scalar related to the time step, see [7]. However, in the case of *asynchronous* constraints, which may occur in models using time partitioning (`PART`) if one activates the `PLIN` option, the scalar γ is replaced by a matrix $\boldsymbol{\Gamma}$, since in that case the time step can vary from one node to another [7].

3.3.1 Subroutine SOLVE_SINGLE_LIAISON

The old and new relevant parts of the subroutine SOLVE_SINGLE_LIAISON are shown side-by-side to highlight the changes:

Old code

```
      SUBROUTINE SOLVE_SINGLE_LIAISON (L, XM, FI, V, FE, KFIFS, FIFS,
     &                                 MVGRIL, CBALE, INDOX, NUMN, X,
     &                                 DFX, POSP, IDOM, POSECR, ECR,
     &                                 POSIG, SIG, RO, WG,
     &                                 FSVECT,DU,THEREF)
 . . .
      INCLUDE 'COPT.INC'
 . . .
*
* calcul de dtsum
      IF (NPAS == 0) THEN
        DTSUM = DT2
      ELSE
        DTSUM = DT1 + DT2
      ENDIF
 . . .
         CALL SOLVE_SINGLE_DVA (L, XM, FI, V, FE, DFX, DTSUM, K)
 . . .
```

New code

```
      SUBROUTINE SOLVE_SINGLE_LIAISON (L, XM, FI, V, FE, KFIFS, FIFS,
     &                                 MVGRIL, CBALE, INDOX, NUMN, X,
     &                                 DFX, POSP, IDOM, POSECR, ECR,
     &                                 POSIG, SIG, RO, WG,
     &                                 FSVECT,DU,THEREF)
 . . .
      INCLUDE 'COPT.INC'
      INCLUDE 'CSCONT.INC' ! For IFLAG2
 . . .
*
      IF (NPAS == 0) THEN
        ALPH = 1.D0 ! Au pas 0, on ecrit les liaisons en vitesse
                    ! toujours sur (n+3/2) quelque soit la valeur
                    ! de ALPHA_LIAJ dans l'input
      ELSE
        ALPH = ALPHA_LIAJ
      ENDIF
*
* calcul de dtsum
      IF (IFLAG2 == 1) THEN ! On ne tient pas compte de OPTI LIAJ eventuel
* cas de la collision et des chocs entre structures articulees composees
* de corps rigides : il y a choc
        DTSUM = 1.D0
      ELSE ! Cas normal (il n'y a pas de collisions)
!fc on n'a pas besoin de tester LIAJRC, il suffit d'utiliser ALPHA_LIAJ
!fc la liaison porte sur les vitesses au pas (n+1+alpha/2):
!fc    si OPTI LIAJ n'a pas ete specifie, par defaut ALPHA_LIAJ=1.0,
!fc    ce qui correspond a ecrire les liaisons en vitesse a (n+3/2)
        IF (NPAS == 0) THEN
          DTSUM = ALPH*DT2
        ELSE
          DTSUM = DT1 + ALPH*DT2
        ENDIF
      ENDIF
 . . .
         CALL SOLVE_SINGLE_DVA (L, XM, FI, V, FE, DFX, ALPH, DTSUM, K)
 . . .
```

First, a local value of α (ALPH) is computed, exactly like in subroutine FLIAIS as explained in Section 3.2.5. Then, the definition of DTSUM is modified by replacing DT2 with ALPH*DT2. This causes velocity-based links to be expressed at the parametrized time $t^{n+1} + \alpha\Delta_t^{n+1}/2$.

Finally, the value of ALPH is passed to subroutine SOLVE_SINGLE_DVA to take into account the parametrization in the case of an imposed velocity constraint (`VITE`) with variable coefficients (link of class 1 or 3), as shown in the next paragraph.

3.3.2 Subroutine SOLVE_SINGLE_DVA

The old and new relevant parts of the subroutine SOLVE_SINGLE_DVA are shown side-by-side to highlight the changes:

Old code

```
      SUBROUTINE SOLVE_SINGLE_DVA (L, XM, FI, V, FE, DFX, DTSUM, K)
 . . .
      REAL(8), INTENT(IN) :: XM(*), FI(*), V(*), DFX(*), DTSUM
 . . .
      CASE (11) ! V(T^N+3/2)
        SELECT CASE (LCLA)
        CASE (0,2)
          VAL = FAC
        CASE (1,3)
          FUN = L%LDATA(1)
          VAL = TIME_VALUE (FUN, T + 0.5D0*DT2) * FAC ! IMPOSED V
          CALL SET_LIAI_BTERM (L, VAL)    ! ONLY FOR VERI
        CASE DEFAULT
          CALL ERRMSS ('SOLVE_SINGLE_DVA','INVALID LCLA')
          CALL EPX_STOP('SOLVE_SINGLE_DVA')
        END SELECT
        FE(K) = FI(K) + 2.D0*XM(K)*(VAL-V(K))/DTSUM
 . . .
```

New code

```
      SUBROUTINE SOLVE_SINGLE_DVA (L, XM, FI, V, FE, DFX, ALPH, DTSUM,
     >                             K)
 . . .
      REAL(8), INTENT(IN) :: XM(*), FI(*), V(*), DFX(*), ALPH, DTSUM
 . . .
      CASE (11) ! V(T^N+1+ALPH/2)
        SELECT CASE (LCLA)
        CASE (0,2)
          VAL = FAC
        CASE (1,3)
          FUN = L%LDATA(1)
          VAL = TIME_VALUE (FUN, T + 0.5D0*ALPH*DT2) * FAC ! IMPOSED V
          CALL SET_LIAI_BTERM (L, VAL)    ! ONLY FOR VERI
        CASE DEFAULT
          CALL ERRMSS ('SOLVE_SINGLE_DVA','INVALID LCLA')
          CALL EPX_STOP('SOLVE_SINGLE_DVA')
        END SELECT
        FE(K) = FI(K) + 2.D0*XM(K)*(VAL-V(K))/DTSUM
 . . .
```

The value of ALPH received from the caller routine (SOLVE_SINGLE_LIAISON) is used to take into account the parametrization in the case of an imposed velocity constraint (`VITE`) with variable coefficients (link of class 1 or 3).

3.3.3 Subroutine SOLVE_TWO_LIAISONS

The old and new relevant parts of the subroutine SOLVE_TWO_LIAISONS are shown side-by-side to highlight the changes:

Old code

```
      SUBROUTINE SOLVE_TWO_LIAISONS (LL, XM, FI, V, FE, KFIFS, FIFS,
     &                               MVGRIL, CBALE, INDOX, NUMN, X,
     &                               POSECR, ECR, POSP, POSIG, SIG,
     &                               DFX, RO, WG, FSVECT, IDOM)
 . . .
      INCLUDE 'CSCONT.INC'
 . . .
      REAL(8) :: DTSUM, AUX, FAC, VAL, LAMBDA1, LAMBDA2, A11, A12, A22,
     >           Z1, Z2, CI, DI, CX, DELTA, GAMMAI, DTO, DTN, AMX
 . . .
* calcul de dtsum
      IF (NPAS == 0) THEN
        DTSUM = DT2
      ELSE
        DTSUM = DT1 + DT2
      ENDIF
 . . .
            CASE (11)         ! VITE
              LCLA = L%LCLASS
              FAC = L%RDATA(1)
              SELECT CASE (LCLA)
              CASE (0)
                VAL = FAC
              CASE (1)
                FUN = L%LDATA(1)
                VAL = TIME_VALUE (FUN, T + 0.5D0*DT2) * FAC
              CASE DEFAULT
                CALL ERRMSS ('SOLVE_TWO_LIAISONS','INVALID LCLA')
                CALL EPX_STOP('')
              END SELECT
              CALL SET_LIAI_BTERM (L, VAL)
 . . .
          IF (IPARTI /= 0 .AND. PLINKS == 1) THEN
* (potentially) asynchronous case : <cgamma> = <c> * <gamma>
            DO I = 1, LD1
 . . .
              IF (NPAS == 0) THEN
                DTO = 0.D0
              ELSE
                DTO = DELTNO(NOD)        ! OLD DT OF NODE
              ENDIF
              NDTN = LEVFAC_NODE(NOD)
              DTN = DELMIN*MAXSUB / NDTN ! NEW DT OF NODE
              GAMMAI = 0.5D0*(DTO + DTN)
              CGAMMA1(I) = CGAMMA1(I)*GAMMAI
            END DO
          ELSE
* normal case (synchronous) : <cgamma> = <c> * gamma
            GAMMAI = 0.5D0*DTSUM
            DO I = 1, LD1
              CGAMMA1(I) = CGAMMA1(I)*GAMMAI
            END DO
          ENDIF
 . . .
          IF (IPARTI /= 0 .AND. PLINKS == 1) THEN
* (potentially) asynchronous case : <cgamma> = <c> * <gamma>
            DO I = 1, LD2
 . . .
              IF (NPAS == 0) THEN
                DTO = 0.D0
              ELSE
                DTO = DELTNO(NOD)        ! OLD DT OF NODE
              ENDIF
              NDTN = LEVFAC_NODE(NOD)
              DTN = DELMIN*MAXSUB / NDTN ! NEW DT OF NODE
              GAMMAI = 0.5D0*(DTO + DTN)
              CGAMMA2(I) = CGAMMA2(I)*GAMMAI
            END DO
          ELSE
* normal case (synchronous) : <cgamma> = <c> * gamma
            GAMMAI = 0.5D0*DTSUM
            DO I = 1, LD2
              CGAMMA2(I) = CGAMMA2(I)*GAMMAI
            END DO
          ENDIF
 . . .
```

New code

```
      SUBROUTINE SOLVE_TWO_LIAISONS (LL, XM, FI, V, FE, KFIFS, FIFS,
     &                               MVGRIL, CBALE, INDOX, NUMN, X,
     &                               POSECR, ECR, POSP, POSIG, SIG,
     &                               DFX, RO, WG, FSVECT, IDOM)
 . . .
      INCLUDE 'CSCONT.INC' ! For IFLAG2
 . . .
      REAL(8) :: DTSUM, AUX, FAC, VAL, LAMBDA1, LAMBDA2, A11, A12, A22,
     >           Z1, Z2, CI, DI, CX, DELTA, GAMMAI, DTO, DTN, AMX, ALPH
 . . .
      IF (NPAS == 0) THEN
        ALPH = 1.D0 ! Au pas 0, on ecrit les liaisons en vitesse
                    ! toujours sur (n+3/2) quelque soit la valeur
                    ! de ALPHA_LIAJ dans l'input
      ELSE
        ALPH = ALPHA_LIAJ
      ENDIF
*
* calcul de dtsum
      IF (IFLAG2 == 1) THEN ! On ne tient pas compte de OPTI LIAJ eventuel
* cas de la collision et des chocs entre structures articulees composees
* de corps rigides : il y a choc
        DTSUM = 1.D0
      ELSE ! Cas normal (il n'y a pas de collisions)
!fc on n'a pas besoin de tester LIAJRC, il suffit d'utiliser ALPHA_LIAJ
!fc la liaison porte sur les vitesses au pas (n+1+alpha/2):
!fc    si OPTI LIAJ n'a pas ete specifie, par defaut ALPHA_LIAJ=1.0,
!fc    ce qui correspond a ecrire les liaisons en vitesse a (n+3/2)
        IF (NPAS == 0) THEN
          DTSUM = ALPH*DT2
        ELSE
          DTSUM = DT1 + ALPH*DT2
        ENDIF
      ENDIF
 . . .
            CASE (11)         ! VITE
              LCLA = L%LCLASS
              FAC = L%RDATA(1)
              SELECT CASE (LCLA)
              CASE (0)
                VAL = FAC
              CASE (1)
                FUN = L%LDATA(1)
                VAL = TIME_VALUE (FUN, T + 0.5D0*ALPH*DT2) * FAC
              CASE DEFAULT
                CALL ERRMSS ('SOLVE_TWO_LIAISONS','INVALID LCLA')
                CALL EPX_STOP('')
              END SELECT
              CALL SET_LIAI_BTERM (L, VAL)
 . . .
          IF (IPARTI /= 0 .AND. PLINKS == 1) THEN
* (potentially) asynchronous case : <cgamma> = <c> * <gamma>
            DO I = 1, LD1
 . . .
              IF (NPAS == 0) THEN
                DTO = 0.D0
              ELSE
                DTO = DELTNO(NOD)        ! OLD DT OF NODE
              ENDIF
              NDTN = LEVFAC_NODE(NOD)
              DTN = DELMIN*MAXSUB / NDTN ! NEW DT OF NODE
              GAMMAI = 0.5D0*(DTO + ALPH*DTN)
              CGAMMA1(I) = CGAMMA1(I)*GAMMAI
            END DO
          ELSE
* normal case (synchronous) : <cgamma> = <c> * gamma
            GAMMAI = 0.5D0*DTSUM
            DO I = 1, LD1
              CGAMMA1(I) = CGAMMA1(I)*GAMMAI
            END DO
          ENDIF
 . . .
          IF (IPARTI /= 0 .AND. PLINKS == 1) THEN
* (potentially) asynchronous case : <cgamma> = <c> * <gamma>
            DO I = 1, LD2
 . . .
              IF (NPAS == 0) THEN
                DTO = 0.D0
              ELSE
                DTO = DELTNO(NOD)        ! OLD DT OF NODE
              ENDIF
              NDTN = LEVFAC_NODE(NOD)
              DTN = DELMIN*MAXSUB / NDTN ! NEW DT OF NODE
              GAMMAI = 0.5D0*(DTO + ALPH*DTN)
              CGAMMA2(I) = CGAMMA2(I)*GAMMAI
            END DO
          ELSE
* normal case (synchronous) : <cgamma> = <c> * gamma
            GAMMAI = 0.5D0*DTSUM
            DO I = 1, LD2
              CGAMMA2(I) = CGAMMA2(I)*GAMMAI
            END DO
          ENDIF
 . . .
```

First, a local value of α (ALPH) is computed, exactly like in subroutine FLIAIS as explained in Section 3.2.5. Then, the definition of DTSUM is modified by replacing DT2 with ALPH*DT2. This causes velocity-based links to be expressed at the parametrized time $t^{n+1} + \alpha\Delta_t^{n+1}/2$.

Finally, the value of ALPH is used also to take into account the parametrization in the case of an imposed velocity constraint (`VITE`) and in the case of potentially asynchronous constraints with partitioning (`PART`), similarly to what is done in subroutine COMPUTE_CGAMMA (see Section 3.3.5 below).

3.3.4 Subroutine SOLVE_GROUP

The old and new relevant parts of the subroutine SOLVE_GROUP are shown side-by-side to highlight the changes:

Old code

```
      SUBROUTINE SOLVE_GROUP (NLIE, NBC, LL_ARRAY, LS,
     &                        HAS_VARIABLE_COEFS,
     &                        HAS_NON_PERMANENT_LINKS,
     &                        HAS_REMOTE_DOFS,
     &                        XM, FI, V, FE, KFIFS, FIFS,
     &                        MVGRIL, CBALE, INDOX, NUMN, X,
     &                        POSECR, ECR, POSP, POSIG, SIG,
     &                        DFX, RO, WG, FSVECT, IDOM, DU, THEREF)
 . . .
      REAL(8) :: DTSUM, AUX, FAC, VAL, VNORM(3), FNORM, AMU, FLC(2),
     &           DTSUM_OLD, RBID(1), FTNOR, FTTAN, MU, REDU, FLC2,
     >           SHA1, SHA2
 . . .
* calcul de dtsum
      IF (NPAS == 0) THEN
        DTSUM = DT2
      ELSE
        DTSUM = DT1 + DT2
      ENDIF
 . . .
*
* update the coefficients and the right hand side of the liaisons,
* if needed, **before** building up ls%coefl(:) and ls%sm(:)
*
        ILIES = 0
    2   ILIES = ILIES + 1
        IF (ILIES <= NLIE) THEN
          L => LL_ARRAY(ILIES)%LIAISON
          IF (L%LCLASS /= 0 .AND. L%LCLASS /= 2) THEN
            ITYP = L%LTYPE
            SELECT CASE (ITYP)
 . . .
            CASE (11)         ! VITE
              LCLA = L%LCLASS
              FAC = L%RDATA(1)
              SELECT CASE (LCLA)
              CASE (0, 2)
                VAL = FAC
              CASE (1, 3)
                FUN = L%LDATA(1)
                VAL = TIME_VALUE (FUN, T + 0.5D0*DT2) * FAC
              CASE DEFAULT
                CALL ERRMSS ('SOLVE_GROUP','INVALID LCLA')
                CALL EPX_STOP('')
              END SELECT
              CALL SET_LIAI_BTERM (L, VAL)
 . . .
```

New code

```
      SUBROUTINE SOLVE_GROUP (NLIE, NBC, LL_ARRAY, LS,
     &                        HAS_VARIABLE_COEFS,
     &                        HAS_NON_PERMANENT_LINKS,
     &                        HAS_REMOTE_DOFS,
     &                        XM, FI, V, FE, KFIFS, FIFS,
     &                        MVGRIL, CBALE, INDOX, NUMN, X,
     &                        POSECR, ECR, POSP, POSIG, SIG,
     &                        DFX, RO, WG, FSVECT, IDOM, DU, THEREF)
 . . .
      REAL(8) :: DTSUM, AUX, FAC, VAL, VNORM(3), FNORM, AMU, FLC(2),
     &           DTSUM_OLD, RBID(1), FTNOR, FTTAN, MU, REDU, FLC2,
     >           SHA1, SHA2, ALPH
 . . .
      IF (NPAS == 0) THEN
        ALPH = 1.D0 ! Au pas 0, on ecrit les liaison en vitesse
                    ! toujours sur (n+3/2) quelque soit la valeur
                    ! ALPHA_LIAJ dans l'input
      ELSE
        ALPH = ALPHA_LIAJ
      ENDIF
*
* calcul de dtsum
      IF (IFLAG2 == 1) THEN ! On ne tient pas compte de OPTI LIAJ eventuel
* cas de la collision et des chocs entre structures articulees composees
* de corps rigides : il y a choc
        DTSUM = 1.D0
      ELSE ! Cas normal (il n'y a pas de collisions)
!fc on n'a pas besoin de tester LIAJRC, il suffit d'utiliser ALPHA_LIAJ
!fc la liaison porte sur les vitesses au pas (n+1+alpha/2):
!fc    si OPTI LIAJ n'a pas ete specifie, par defaut ALPHA_LIAJ=1.0,
!fc    ce qui correspond a ecrire les liaisons en vitesse a (n+3/2)
        IF (NPAS == 0) THEN
          DTSUM = ALPH*DT2
        ELSE
          DTSUM = DT1 + ALPH*DT2
        ENDIF
      ENDIF
 . . .
*
* update the coefficients and the right hand side of the liaisons,
* if needed, **before** building up ls%coefl(:) and ls%sm(:)
*
        ILIES = 0
    2   ILIES = ILIES + 1
        IF (ILIES <= NLIE) THEN
          L => LL_ARRAY(ILIES)%LIAISON
          IF (L%LCLASS /= 0 .AND. L%LCLASS /= 2) THEN
            ITYP = L%LTYPE
            SELECT CASE (ITYP)
 . . .
            CASE (11)         ! VITE
              LCLA = L%LCLASS
              FAC = L%RDATA(1)
              SELECT CASE (LCLA)
              CASE (0, 2)
                VAL = FAC
              CASE (1, 3)
                FUN = L%LDATA(1)
                VAL = TIME_VALUE (FUN, T + 0.5D0*ALPH*DT2) * FAC
              CASE DEFAULT
                CALL ERRMSS ('SOLVE_GROUP','INVALID LCLA')
                CALL EPX_STOP('')
              END SELECT
              CALL SET_LIAI_BTERM (L, VAL)
 . . .
```

First, a local value of α (ALPH) is computed, exactly like in subroutine FLIAIS as explained in Section 3.2.5. Then, the definition of DTSUM is modified by replacing DT2 with ALPH*DT2. This causes velocity-based links to be expressed at the parametrized time $t^{n+1} + \alpha\Delta_t^{n+1}/2$.

Finally, in the case of an imposed velocity constraint (`VITE`) with variable coefficients (link of class 1 or 3), the time function is evaluated at the parametrized time, again by replacing DT2 with ALPH*DT2.

3.3.5 Subroutine COMPUTE_CGAMMA

The old and new relevant parts of the subroutine COMPUTE_CGAMMA are shown side-by-side to highlight the changes:

Old code

```
      SUBROUTINE COMPUTE_CGAMMA (LL_ARRAY, NLIE, COEFL, DTSUM, LPAIR,
     >                           IFLAG2, CGAMMA)
*
*fc 9 march 2006: compute cgamma:
* for a link on velocities:     cgamma = coefl * gamma
* for a link on accelerations:  cgamma = coefl
*
      USE M_PARTITION_DATA
*
      INCLUDE 'TEMPX.INC'
 . . .
*
* locals
      TYPE(LIAISON), POINTER :: L
      INTEGER :: K, ILIES, NDL, ITYP, I, IDOF, NOD, NDTN
      REAL(8) :: DTO, DTN, GAMMAI
      LOGICAL :: ON_VELOCITIES
*
      K = 0
      DO ILIES = 1, NLIE
 . . .
*
       IF (ON_VELOCITIES) THEN
* link on velocities
         IF (IPARTI /= 0 .AND. PLINKS == 1) THEN
* (potentially) asynchronous case : <cgamma> = <c> * <gamma>
           DO I = 1, NDL
             K = K + 1
             IDOF = LPAIR(K)            ! CONCERNED DOF
             NOD = DOF_NODE(IDOF)       ! CORRESPONDING NODE
             IF (NPAS == 0) THEN
               DTO = 0.D0
             ELSE
               DTO = DELTNO(NOD)        ! OLD DT OF NODE
             ENDIF
             NDTN = LEVFAC_NODE(NOD)
             DTN = DELMIN*MAXSUB / NDTN ! NEW DT OF NODE
             GAMMAI = 0.5D0*(DTO + DTN)
             CGAMMA(K) = COEFL(K)*GAMMAI
           END DO
         ELSE
* normal case (synchronous) : <cgamma> = <c> * gamma
           GAMMAI = 0.5D0*DTSUM
           DO I = 1, NDL
             K = K + 1
             CGAMMA(K) = COEFL(K)*GAMMAI
           END DO
         ENDIF
       ELSE
* link on accelerations
         DO I = 1, NDL
           K = K + 1
           CGAMMA(K) = COEFL(K)
         END DO
       ENDIF
     END DO
 . . .
```

New code

```
      SUBROUTINE COMPUTE_CGAMMA (LL_ARRAY, NLIE, COEFL, DTSUM, LPAIR,
     >                           IFLAG2, CGAMMA)
*
*fc 9 march 2006: compute cgamma:
* for a link on velocities:     cgamma = coefl * gamma
* for a link on accelerations:  cgamma = coefl
*
* Prise en compte de l'option LIAJ <ALPH alph>:
* - dtsum calcule dans SOLVE_GROUP tient deja compte
*   de l'eventuelle presence de l'option LIAJ <ALPH alph>
* - par contre, dans le "(potentially) asynchronous case" ci-dessous
*   on n'utilise pas dtsum et donc il faut tenir compte
*   de l'eventuelle presence de l'option LIAJ <ALPH alph>
*
      USE M_PARTITION_DATA
*
      INCLUDE 'COPT.INC'  ! For ALPHA_LIAJ
      INCLUDE 'TEMPX.INC'
 . . .
*
* locals
      TYPE(LIAISON), POINTER :: L
      INTEGER :: K, ILIES, NDL, ITYP, I, IDOF, NOD, NDTN
      REAL(8) :: DTO, DTN, GAMMAI, ALPH
      LOGICAL :: ON_VELOCITIES
*
      K = 0
      DO ILIES = 1, NLIE
 . . .
*
       IF (ON_VELOCITIES) THEN
* link on velocities
         IF (IPARTI /= 0 .AND. PLINKS == 1) THEN
* (potentially) asynchronous case : <cgamma> = <c> * <gamma>
           IF (NPAS == 0) THEN
             ALPH = 1.D0 ! Au pas 0, on ecrit les liaison en vitesse
                         ! toujours sur (n+3/2) quelque soit la valeur
                         ! ALPHA_LIAJ dans l'input
           ELSE
             ALPH = ALPHA_LIAJ
           ENDIF
           DO I = 1, NDL
             K = K + 1
             IDOF = LPAIR(K)            ! CONCERNED DOF
             NOD = DOF_NODE(IDOF)       ! CORRESPONDING NODE
             IF (NPAS == 0) THEN
               DTO = 0.D0
             ELSE
               DTO = DELTNO(NOD)        ! OLD DT OF NODE
             ENDIF
             NDTN = LEVFAC_NODE(NOD)
             DTN = DELMIN*MAXSUB / NDTN ! NEW DT OF NODE
             GAMMAI = 0.5D0*(DTO + ALPH*DTN)
             CGAMMA(K) = COEFL(K)*GAMMAI
           END DO
         ELSE
* normal case (synchronous) : <cgamma> = <c> * gamma
           GAMMAI = 0.5D0*DTSUM
           DO I = 1, NDL
             K = K + 1
             CGAMMA(K) = COEFL(K)*GAMMAI
           END DO
         ENDIF
       ELSE
* link on accelerations
         DO I = 1, NDL
           K = K + 1
           CGAMMA(K) = COEFL(K)
         END DO
       ENDIF
     END DO
*
      END SUBROUTINE COMPUTE_CGAMMA
 . . .
```

A local value of α (ALPH) is computed, exactly like in subroutine FLIAIS as explained in Section 3.2.5. Then, the value of ALPH is used to take into account the parametrization for any velocity-based constraint (ON_VELOCITIES), in the case of potentially asynchronous constraints with partitioning (`PART`).

3.4 The `LINK DECO DEPL|VITE|ACCE` directive

The code flowchart for the `LINK DECO DEPL|VITE|ACCE` directive ("decoupled links" model) is presented in Figure 3. Although not formally a coupled liaison or coupled link model, this directive may be considered relevant here because the weak (decoupled) formulation of these particular links is obtained by a *direct solution* of the corresponding constraints, as detailed in reference [12].

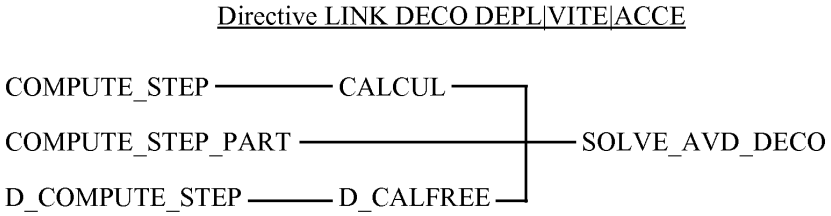


Figure 3: Code flowchart for the `LINK DECO DEPL|VITE|ACCE` directive.

The solution is implemented in subroutine SOLVE_AVD_DECO, which is very similar to the subroutine SOLVE_SINGLE_DVA described in the previous paragraph for the case of coupled links, which is used when a constraints group happens to contain just one condition and can therefore be solved analytically.

The main difference between SOLVE_AVD_DECO and SOLVE_SINGLE_DVA is that the latter contains also code for a specific treatment in the case with domain decomposition (parallel solution under MPI). This is not necessary in SOLVE_AVD_DECO because the general treatment of (formally) coupled links in the MPI case is done in other parts of the code.

Note that in principle the optional velocity parametrization (`LIAJ`) should be taken into account in all velocity-based decoupled links (`LINK DECO`) which use a *direct* solution (as opposed, for example, to a penalty-based solution). At the moment of this writing, however, this is done only for the `LINK DECO VITE` directive.

3.4.1 Subroutine SOLVE_AVD_DECO

The old and new relevant parts of the subroutine SOLVE_AVD_DECO are shown side-by-side to highlight the changes:

Old code

```
      SUBROUTINE SOLVE_AVD_DECO (XM, FI, V, FE, DFX, FDECO, POSP)
 . . .
      INCLUDE "CONTRO.INC" ! FOR NLIB (GLOBAL N. OF DOFS)
 . . .
      REAL(8) :: COEF, VAL, DTSUM, AUX
 . . .
      IF (NPAS == 0) THEN
        DTSUM = DT2
      ELSE
        DTSUM = DT1 + DT2
      ENDIF
 . . .
            CASE (2) ! VITE
              VAL = TIME_VALUE(IFONC,T+0.5D0*DT2)*COEF ! IMPOSED V^N+3/2
              FDECO(K) = 2.D0*XM(K)*(VAL-V(K))/DTSUM + FI(K) - FE(K)
              FE(K) = FE(K) + FDECO(K)
 . . .
            CASE (2) ! VITE
              VAL = TIME_VALUE(IFONC,T+0.5D0*DT2)*COEF ! IMPOSED V^N+3/2
              FDECO(K) = 2.D0*XM(K)*(VAL-V(K))/DTSUM + FI(K) - FE(K)
* fe will be treated later in the case with sub-domains (MPI)
 . . .
```

New code

```
      SUBROUTINE SOLVE_AVD_DECO (XM, FI, V, FE, DFX, FDECO, POSP)
 . . .
      INCLUDE "CONTRO.INC" ! FOR NLIB (GLOBAL N. OF DOFS)
      INCLUDE "COPT.INC"   ! FOR ALPHA_LIAJ
      INCLUDE 'CSCONT.INC' ! For IFLAG2
 . . .
      REAL(8) :: COEF, VAL, DTSUM, AUX, ALPH
 . . .
      IF (NPAS == 0) THEN
        ALPH = 1.D0 ! Au pas 0, on ecrit les liaisons en vitesse
                    ! toujours sur (n+3/2) quelque soit la valeur
                    ! de ALPHA_LIAJ dans l'input
      ELSE
        ALPH = ALPHA_LIAJ
      ENDIF
*
* calcul de dtsum
      IF (IFLAG2 == 1) THEN ! On ne tient pas compte de OPTI LIAJ eventuel
* cas de la collision et des chocs entre structures articulees composees
* de corps rigides : il y a choc
        DTSUM = 1.D0
      ELSE ! Cas normal (il n'y a pas de collisions)
!fc on n'a pas besoin de tester LIAJRC, il suffit d'utiliser ALPHA_LIAJ
!fc la liaison porte sur les vitesses au pas (n+1+alpha/2):
!fc    si OPTI LIAJ n'a pas ete specifie, par defaut ALPHA_LIAJ=1.0,
!fc    ce qui correspond a ecrire les liaisons en vitesse a (n+3/2)
        IF (NPAS == 0) THEN
          DTSUM = ALPH*DT2
        ELSE
          DTSUM = DT1 + ALPH*DT2
        ENDIF
      ENDIF
 . . .
            CASE (2) ! VITE IMPOSED AS V^N+1+ALPH/2
              VAL = TIME_VALUE(IFONC,T+0.5D0*ALPH*DT2)*COEF
              FDECO(K) = 2.D0*XM(K)*(VAL-V(K))/DTSUM + FI(K) - FE(K)
              FE(K) = FE(K) + FDECO(K)
 . . .
            CASE (2) ! VITE IMPOSED AS V^N+1+ALPH/2
              VAL = TIME_VALUE(IFONC,T+0.5D0*ALPH*DT2)*COEF
              FDECO(K) = 2.D0*XM(K)*(VAL-V(K))/DTSUM + FI(K) - FE(K)
* fe will be treated later in the case with sub-domains (MPI)
 . . .
```

First, a local value of α (ALPH) is computed, exactly like in subroutine FLIAIS as explained in Section 3.2.5. Then, the definition of DTSUM is modified by replacing DT2 with ALPH*DT2. This causes velocity-based links to be expressed at the parametrized time $t^{n+1} + \alpha\Delta_t^{n+1}/2$.

Finally, in the case of an imposed velocity constraint (`VITE`) the time function is evaluated at the parametrized time, again by replacing DT2 with ALPH*DT2.

4 Numerical examples

We present a series of numerical examples to illustrate and verify the developments detailed in the previous Sections.

4.1 Bar impact tests

We start by re-considering the bar impact problem that had been studied in references [9, 10] and which had led to the first implementation of the `LIAJ` option in the early 2000's. The tests performed are summarized in Table 2.

Case	Mesh	Description	T_{fin} [ms]	Steps	CPU [s]
PINB03	200 Q41L	Original from 2002, `LAGC`, `LIAI` with `LIAJ` option	100	566	0.3
PINL03	200 Q41L	Idem PINB03 but without `LIAJ`	100	558	0.3
PINB13	200 Q41L	Idem PINB03 but `LINK COUP`	100	566	0.2
PINL13	200 Q41L	Idem PINL03 but `LINK COUP`	100	558	0.2
PINB23	200 Q41L, 1 PMAT	Idem PINB03 but add fake PMAT	100	566	0.3
PINL23	200 Q41L, 1 PMAT	Idem PINL03 but add fake PMAT	100	558	0.3
PINB33	200 Q41L, 1 PMAT	Idem PINB13 but add fake PMAT and `SPLT NONE`	100	566	0.2
PINL33	200 Q41L, 1 PMAT	Idem PINL13 but add fake PMAT and `SPLT NONE`	100	558	0.2
PINB43	200 Q41L	Idem PINB03 but remove `OPTI PINS REB2`	100	566	0.3
PINB53	200 Q41L, 1 PMAT	Idem PINB33 but add `LIAJ ALPH 0.0`	100	566	0.2
PINB63	200 Q41L	Idem PINB13 but remove `OPTI PINS REB2`	100	566	0.3
PINB73	200 Q41L, 1 PMAT	Idem PINB53 but `LIAJ ALPH 0.05`	100	558	0.2
PINB83	200 Q41L, 1 PMAT	Idem PINB53 but `LIAJ ALPH 0.10`	100	557	0.3
PINB93	200 Q41L, 1 PMAT	Idem PINB53 but `LIAJ ALPH 0.50`	100	556	0.3

Table 2: Bar impact tests.

4.1.1 Case PINB03

This 2D problem consists of two identical elastic bars of dimensions 200×2 m, impacting each other with opposite initial velocities of 500 m/s. Each bar is discretized by 100 square continuum elements of type Q41L. The contact is modeled by pinballs (`PINB`) without hierarchic refinement, so that the treatment of contact is relatively imprecise geometrically (but this is not an issue for the purpose of this numerical test). The *a posteriori* rebound algorithm is adopted (`OPTI PINS REB2`). The velocity-based constraints (here the pinball contacts) are expressed at $n+1$ (i.e. by taking $\alpha = 0$ to achieve a full-step constraint), by means of the `OPTI LIAJ` keyword.

Figure 4 presents some results, namely the displacements of some characteristic points, the reactions at the contacting extremities of the bars, the stresses at the bar mid-points and the energies.

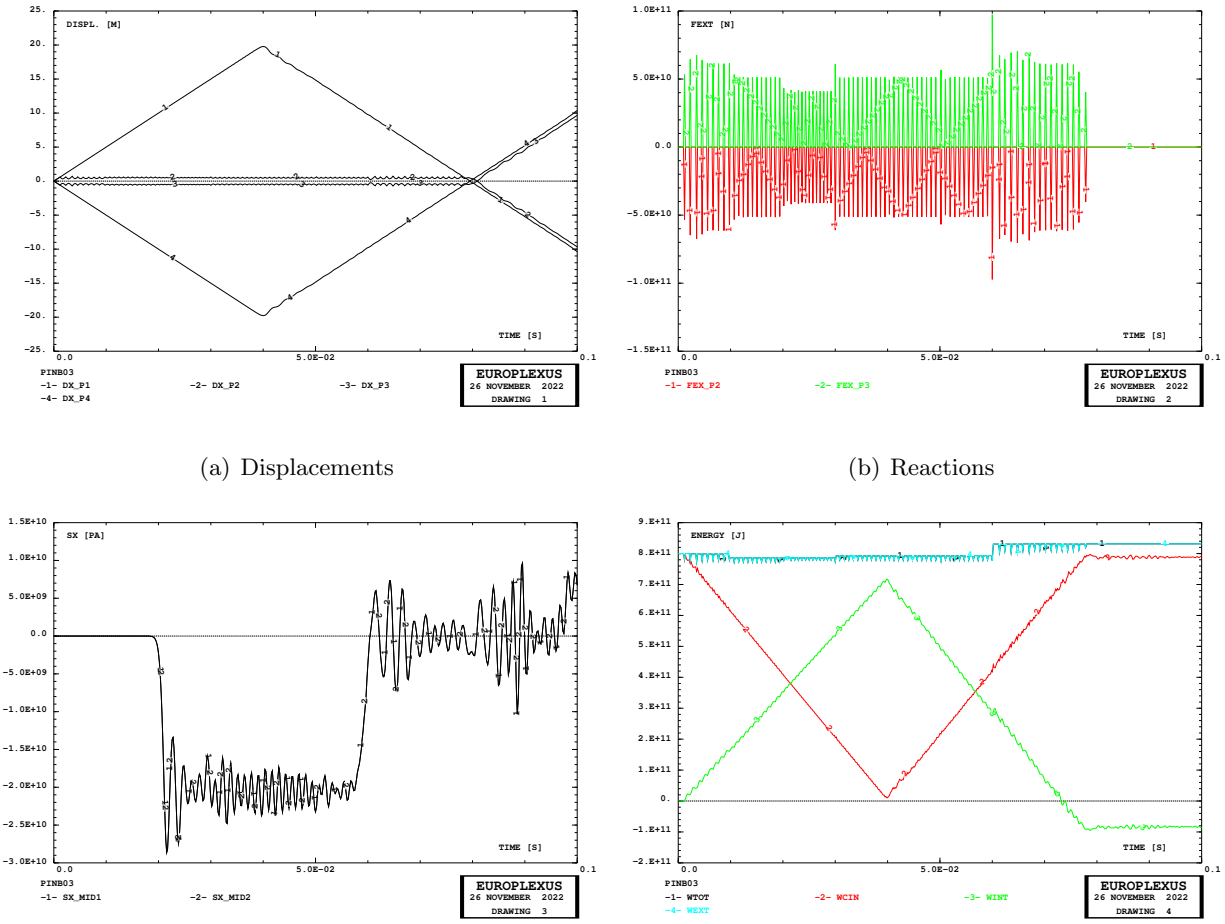
The solution is identical to that published in references [9, 10]. The contact forces are mutually opposite as expected, but present large oscillations (intermittent contact). A small amount of energy seems to be lost at the moment of the impact, although for $\alpha = 0$ the energy should be conserved. However, this test is not very significant to check the energy conservation properties of the constraints scheme, since contact occurs only on a small portion of the structure (the two bar tips).

4.1.2 Case PINL03

This test is identical to PINB03 but without the `LIAJ` option, so that velocity-based constraints are imposed at step $n + 3/2$ as per default in EPX (this corresponds to $\alpha = 1$).

The result is, again, identical to that published in [9, 10]. The contact force is much smoother than in the previous solution and presents almost no oscillations (continuous contact), as it can be seen in Figure 5.

It is difficult to say whether the energy conservation is worse or the same as in case PINB03, for the reasons mentioned in the previous paragraph.



(a) Displacements

(b) Reactions

(c) Stresses

(d) Energies

Figure 4: Results of test PINB03.

4.1.3 Case PINB13

This test is identical to PINB03 (`LIAJ`) but uses the `LINK COUP` directive for the constraints. It is meant to check the implementation of `LIAJ` for the modern constraints model (`LINK`) presented in this report.

The results obtained (`.ps` file) are identical (`diff` command) to those with the `LIAI` constraints model shown in Figure 4 and are not presented for brevity.

4.1.4 Case PINL13

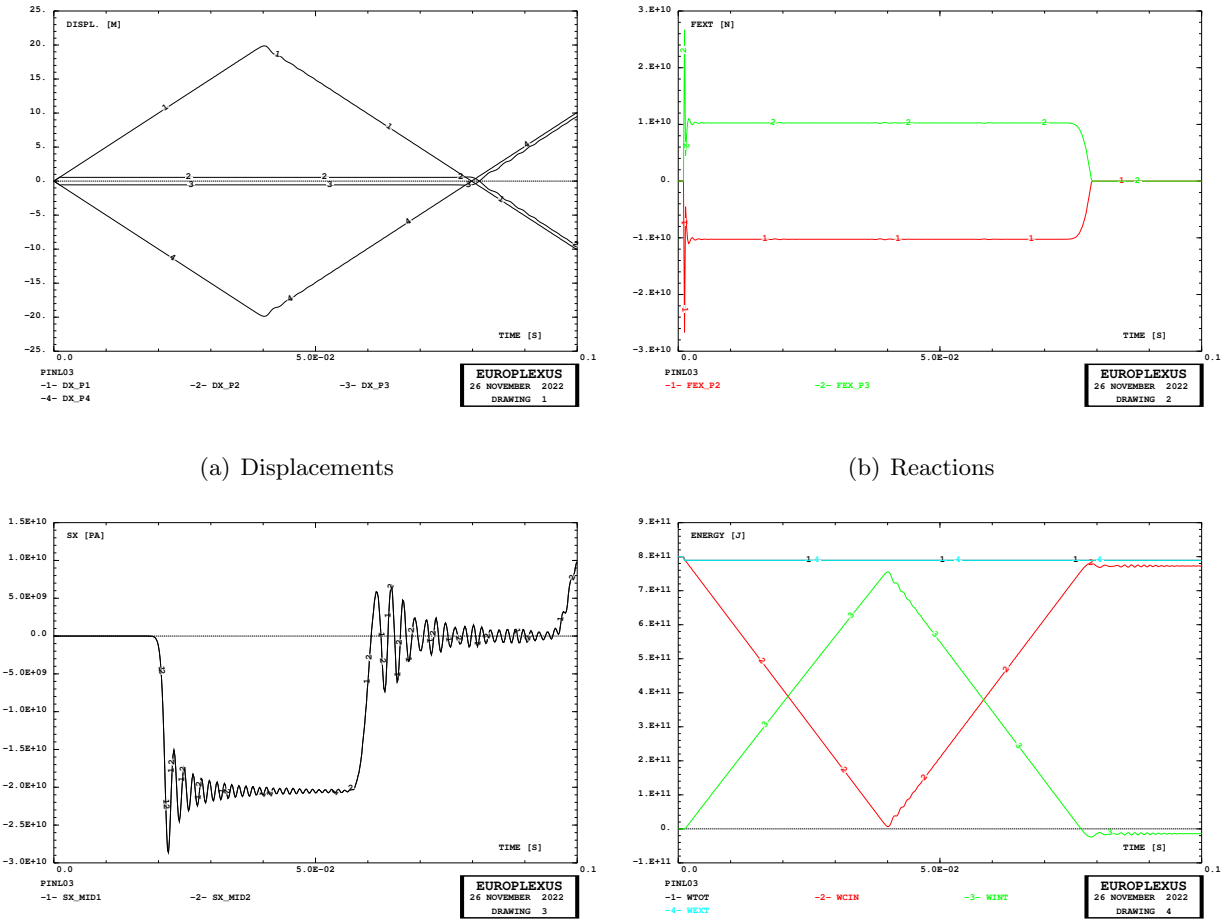
This test is identical to PINL03 (no `LIAJ`) but uses the `LINK COUP` directive for the constraints. As expected, also in this case the results obtained (`.ps` file) are identical to those with the `LIAI` constraints model shown in Figure 5 and are not presented for brevity.

4.1.5 Case PINB23

The two last tests show that the properties in terms of oscillations in the solution do not depend upon the constraints model chosen (`LIAI` or `LINK`), but only on the time discretization of the velocity-based constraints (`LIAJ`).

The tests run so far involved only one (contact) constraint. Therefore, when `LINK` is used, a direct solution is employed internally by EPX, i.e. subroutine SOLVE_SINGLE_LIAISON, see Figure 2 and Section 3.3.1.

However, it is desirable to check also the more general SOLVE_GROUP routine, which is activated



(a) Displacements

(b) Reactions

(c) Stresses

(d) Energies

Figure 5: Results of test PINL03.

when a group with several constraints has to be treated. To this end, we slightly modify the test PINB03 as follows. A fake material point (PMAT element) with a phantom (`FANT`) material is added to the mesh, and it is blocked (`BLOQ`) in both directions, thus generating two extra constraints. This element is not connected to the bar elements and therefore its presence should have no effect on the numerical solution.

We first solve the modified problem with `LIAI` and `LIAJ`, like in case PINB03. As expected, the results obtained (`.ps` file) are identical to those of test PINB03 constraints model shown in Figure 4 and are not presented for brevity.

4.1.6 Case PINL23

This test is identical to PINL03 (`LIAI`) but we add the fake PMAT and block it like in the previous example. As expected, the results obtained (`.ps` file) are identical to those of test PINL03 shown in Figure 5 and are not presented for brevity.

4.1.7 Case PINB33

This test is identical to PINB13 (`LINK COUP` and `LIAJ`) but we add the fake PMAT and block it like in the previous example. Note that in order to actually pass in subroutine SOLVE_GROUP, it is necessary to add the optional keywords `SPLT NONE` to the `LINK COUP` directive. In this way, EPX does not split the system of constraints into several groups but performs a monolithic solution (like for the `LIAI` model).

As expected, the results obtained (`.ps` file) are identical to those of test PINB03 shown in Figure 4, and to those of test PINB23, and are not presented for brevity.

4.1.8 Case PINL33

This test is identical to PINL13 (`LINK COUP` without `LIAJ`) but we add the fake PMAT and block it like in the previous example. We also add the optional keywords `SPLT NONE` to the `LINK COUP` directive, like in the previous example.

As expected, the results obtained (`.ps` file) are identical to those of test PINL03 shown in Figure 5, and to those of tests PINL13 and PINL23, and are not presented for brevity. These two latter tests verify also the SOLVE_GROUP routine.

4.1.9 Case PINB43

This test is a repetition of case PINB03 but we remove the option `PINS REB2` so that the default rebound algorithm (*a priori*) is used. The scope is to check whether the strong oscillations in the numerical solution of test PINB03 (see Figure 4) are at least partially due to the rebound algorithm, besides the numerical scheme of the constraints.

A bit surprisingly, the results obtained are identical (`.ps` file) to those of test PINB03 and are not shown for brevity. It seems therefore that the rebound algorithm has no effect on the solution in this test problem.

4.1.10 Case PINB53

This test is identical to case PINB33 but we add `ALPH 0.D0` to the `LIAJ` option. Since, when `LIAJ` is specified, the default value of α is 0.0 (for compatibility with the old syntax ans tests), we expect no changes in the results. This is indeed the case and the solution is not presented for brevity.

4.1.11 Case PINB63

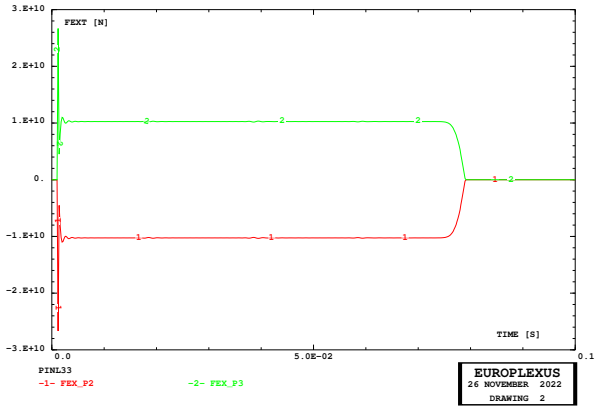
This test is a repetition of case PINB13 but we remove the option `PINS REB2` so that the default rebound algorithm (*a priori*) is used. The scope is to check whether the lack of influence of the rebound algorithm on the numerical solution observed in test PINB43 with the `LIAI` model is confirmed also for the `LINK` model.

Indeed, this is the case, since the solution is identical to that of case PINB13. This seems to confirm that the rebound algorithm has no effect on the solution in this test problem.

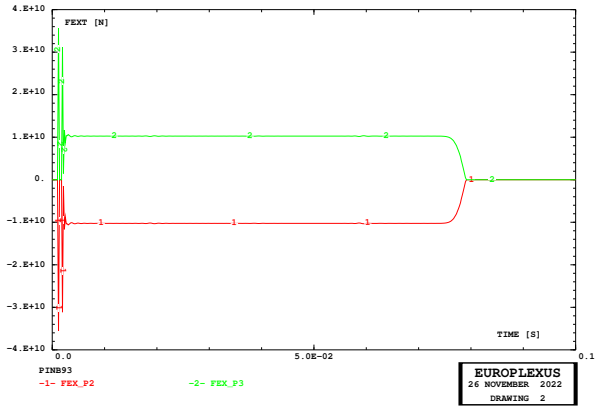
4.1.12 Cases PINB73, PINB83, PINB93

We do a small parametric study to show the influence of the value of α on the solution. These three tests use $\alpha = 0.05$, $\alpha = 0.10$ and $\alpha = 0.50$, respectively.

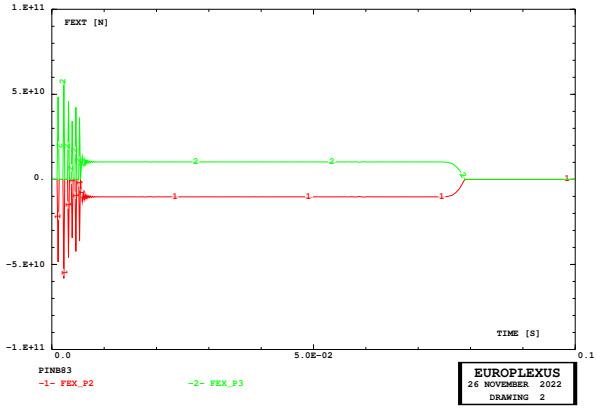
The results are summarized in Figure 6 together with those of case PINB53 ($\alpha = 0.0$) and PINL33 (no `LIAJ`, corresponding to $\alpha = 1.0$).



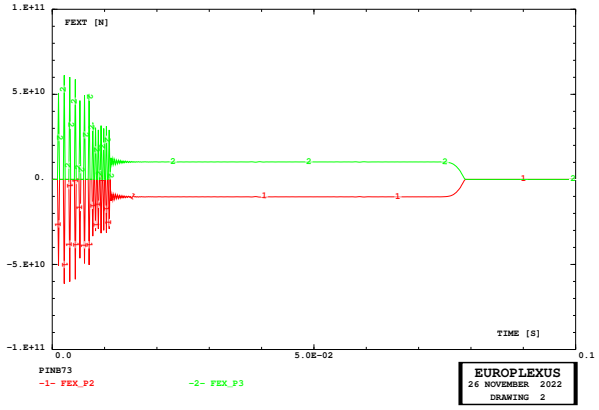
(a) $\alpha = 1.00$ PINL33



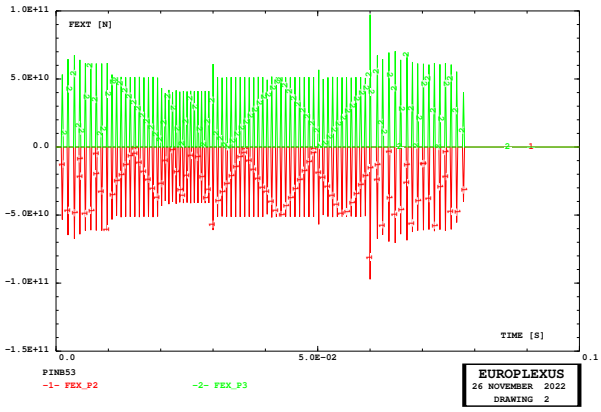
(b) $\alpha = 0.50$ PINB93



(c) $\alpha = 0.10$ PINB83



(d) $\alpha = 0.01$ PINB73



(e) $\alpha = 0.00$ PINB53

Figure 6: Influence of the α parameter.

4.2 Simplified crash tests

We now consider a more representative test, the simplified crash of a vehicle component (a cross beam structure) against a rigid wall. The cross beam is modeled by shell elements, see Figure 7(a), and a linear elastic material is assumed.

The simulations performed are summarized in Table 3.

Case	Mesh	Description	T_{fin} [ms]	Steps	CPU [s]
CBEA01	1536 Q4GS 8 T3GS	$\alpha = 0$	20	15 278	61.2
CBEA02	1536 Q4GS 8 T3GS	$\alpha = 1$	20	15 278	62.1

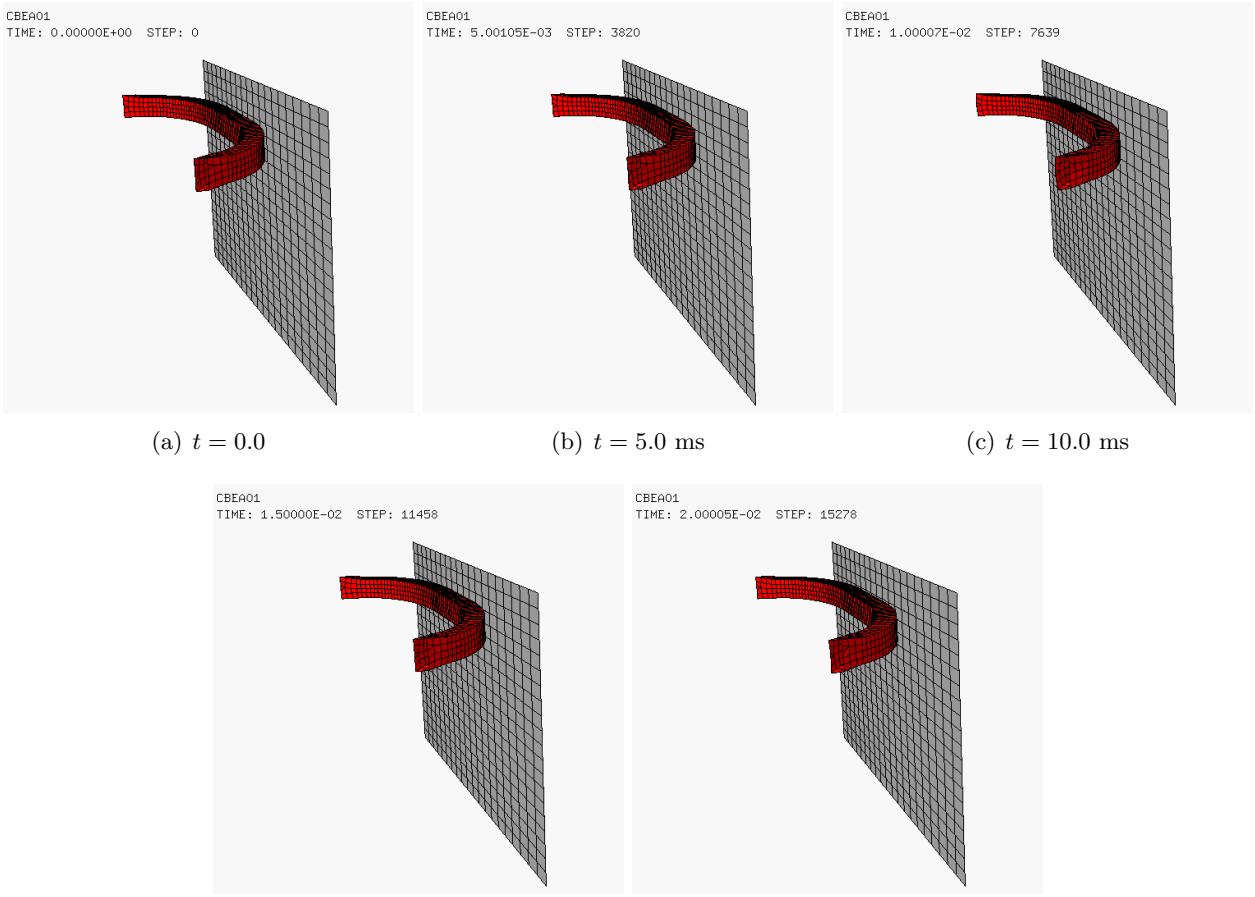
Table 3: Simplified crash tests.

4.2.1 Case CBEA01

This test uses $\alpha = 0$. The input file reads:

```
CBEA01
ECHO
KFIL 'cbea.k'
TRID LAGR
GEOM SCAL FACT 0.001
  Q4GS she4
  T3GS she3
TERM
COMP GROU 1 'vehi' LECT TOUS TERM
                   COND XB GT -10.E-3
MATE LINE RO 7800. YOUN 210E9 NU 0.28
          LECT PART 700 PART 126 TERM
INIT VITE 1 -10.
     LECT vehi TERM
LIAI LAGC
     BLOQ 123456 LECT PART 700 TERM
     GLIS 1 SELF CMAI LECT PART 700 TERM DIRE
                 PESC LECT PART 126 TERM
OPTI NOTE
     CSTA 0.4
     LOG 1000
     GLIS NORM ELEM
     LIAJ ALPH 0.
*    LIAJ ALPH 1.
REGI 'barr' RMAS BARY WEXT RESU IRES
            LECT PART 700 TERM
     'vehi' RMAS BARY RESU WINT WEXT ECIN
            DMOY VMOY AMOY IMPU IRES
            LECT vehi TERM
ECRI DEPL VITE ACCE PERF TFRE 0.0001
          POIN LECT NSET 32 NSET 33 NSET 34 NSET 35 TERM
          NOEL
     FICH ALIT TFRE 0.0001
          POIN LECT NSET 32 NSET 33 NSET 34 NSET 35 TERM
     FICH SPLI ALIC TFRE 0.001
!    FICH PVTK TFRE 0.001
!         GROU AUTO
!         VARI DEPL ECRO VITE FAIL CONT FLIA DTST
CALC TINI 0 TEND 0.02
SUIT
Post
ECHO
RESU ALIC TEMP GARD PSCR
SORT GRAP AXTE 1. 't [s]'
COUR 1 'N32-D' DEPL COMP 1 NOEU LECT 1   TERM
COUR 2 'N33-D' DEPL COMP 1 NOEU LECT 287 TERM
COUR 3 'N34-D' DEPL COMP 1 NOEU LECT 585 TERM
COUR 4 'N35-D' DEPL COMP 1 NOEU LECT 857 TERM
TRAC 1 2 3 4 AXES 1. 'DISP (m)'
LIST 1 2 3 4 AXES 1. 'DISP (m)'
!
COUR 5 'N32-V' VITE COMP 1 NOEU LECT 1   TERM
COUR 6 'N33-V' VITE COMP 1 NOEU LECT 287 TERM
COUR 7 'N34-V' VITE COMP 1 NOEU LECT 585 TERM
COUR 8 'N35-V' VITE COMP 1 NOEU LECT 857 TERM
TRAC 5 6 7 8 AXES 1. 'VELO (m/s)'
LIST 5 6 7 8 AXES 1. 'VELO (m/s)'
!
COUR  9 'N32-A' ACCE COMP 1 NOEU LECT 1   TERM
COUR 10 'N33-A' ACCE COMP 1 NOEU LECT 287 TERM
COUR 11 'N34-A' ACCE COMP 1 NOEU LECT 585 TERM
COUR 12 'N35-A' ACCE COMP 1 NOEU LECT 857 TERM
TRAC 9 10 11 12 AXES 1. 'ACCE (m/s2)'
LIST 9 10 11 12 AXES 1. 'ACCE (m/s2)'
!
COUR 13 'WTOT' WTOT
COUR 14 'WCIN' WCIN
COUR 15 'WINT' WINT
COUR 16 'WIMP' WIMP
TRAC 13 14 15 16 AXES 1. 'ENER (J)'
LIST 13 14 15 16 AXES 1. 'ENER (J)'
!
COUR 17 'WTOT' WTOT
COUR 18 'WCIN' WCIN
COUR 19 'WINT' WINT
COUR 20 'WIMP' WIMP
TRAC 17 18 19 20 AXES 1. 'ENER (J)'
LIST 17 18 19 20 AXES 1. 'ENER (J)'
!
COUR 21 'vehi EEXT' EEXT REGI 2
COUR 22 'vehi ECIN' ECIN REGI 2
COUR 23 'vehi EINT' EINT REGI 2
TRAC 21 22 23 AXES 1. 'ENER (J)'
LIST 21 22 23 AXES 1. 'ENER (J)'
!
COUR 24 'vehi IMPU1' IMPU COMP 1 REGI 2
COUR 25 'vehi IMPU2' IMPU COMP 2 REGI 2
COUR 26 'vehi IMPU3' IMPU COMP 3 REGI 2
TRAC 24 25 26 AXES 1. 'momen. (kg m/s)'
LIST 24 25 26 AXES 1. 'momen. (kg m/s)'
!
COUR 27 'vehi RESU1' RESU COMP 1 REGI 2
COUR 28 'vehi RESU2' RESU COMP 2 REGI 2
COUR 29 'vehi RESU3' RESU COMP 3 REGI 2
TRAC 27 28 29 AXES 1. 'force (N)'
LIST 27 28 29 AXES 1. 'force (N)'
!
COUR 30 'vehi IRES1' IRES COMP 1 REGI 2
COUR 31 'vehi IRES2' IRES COMP 2 REGI 2
COUR 32 'vehi IRES3' IRES COMP 3 REGI 2
TRAC 30 31 32 AXES 1. 'impul. (N s)'
LIST 30 31 32 AXES 1. 'impul. (N s)'
!
FIN
```

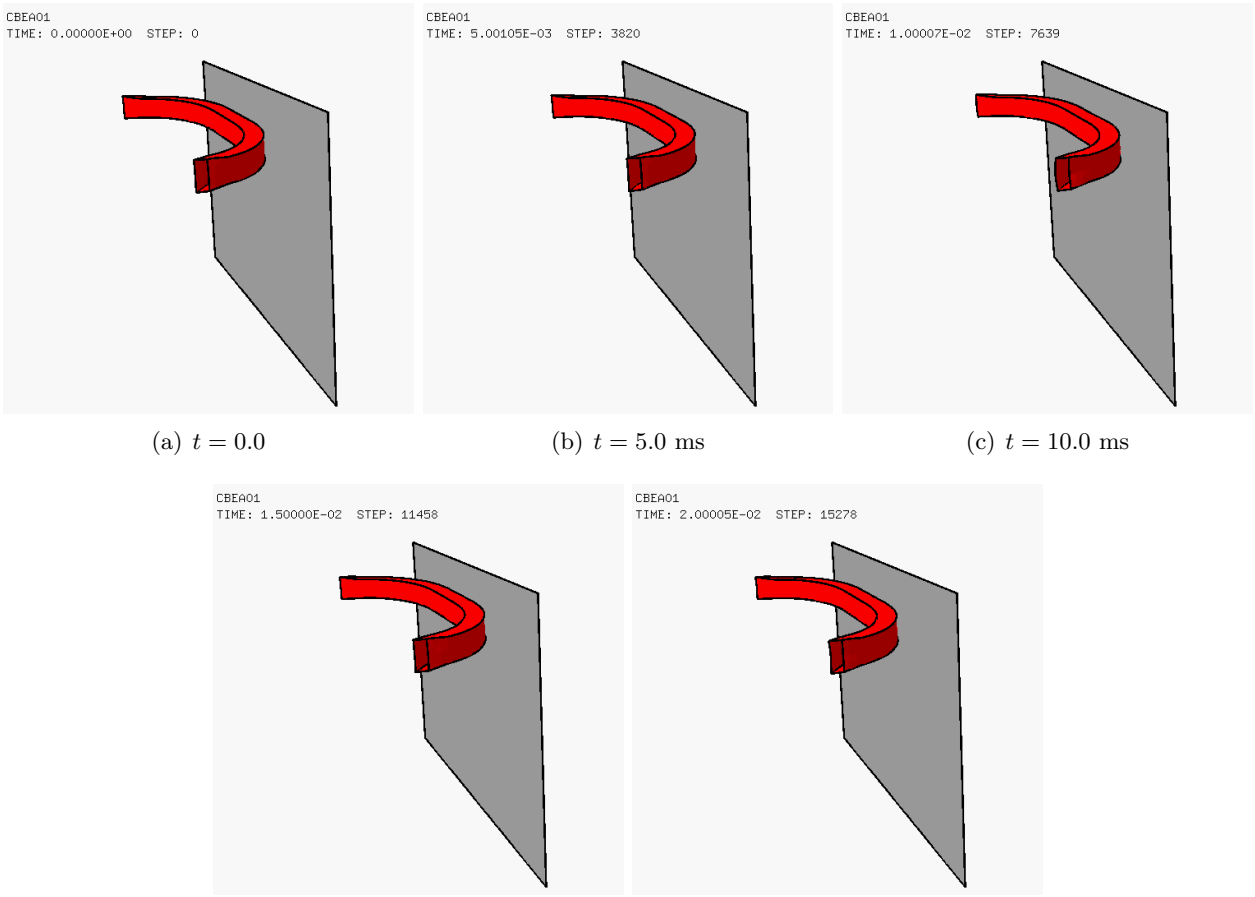
Figure 7 shows the deformed model at various time instants. Figure 7 shows the same results but with the mesh lines (element outlines) removed. The cross beam hits the rigid wall, deforms elastically and then bounces back.



(a) $t = 0.0$ (b) $t = 5.0$ ms (c) $t = 10.0$ ms

(d) $t = 15.0$ ms (e) $t = 20.0$ ms

Figure 7: Deformed geometry in test CBEA01.



(a) $t = 0.0$ (b) $t = 5.0$ ms (c) $t = 10.0$ ms

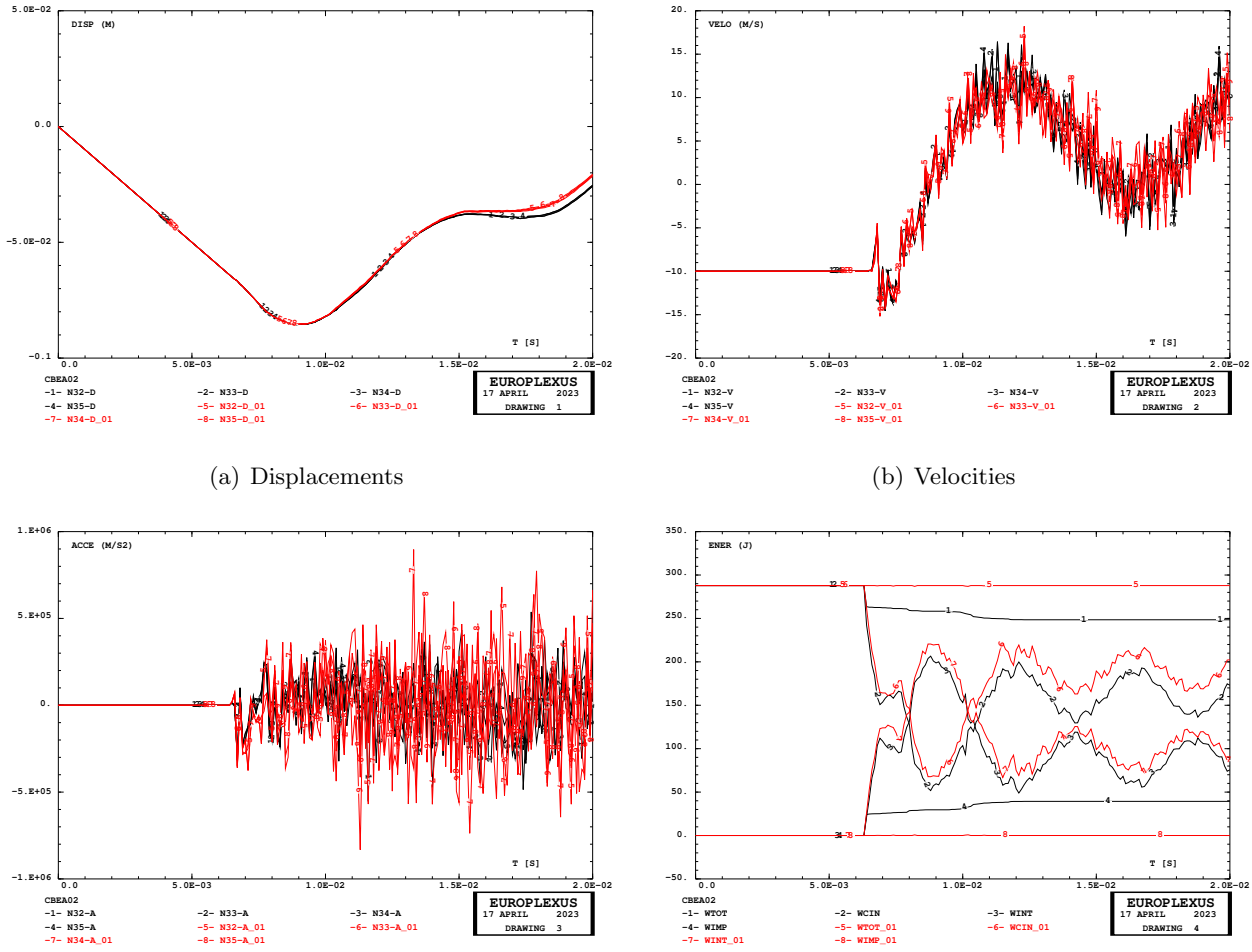
(d) $t = 15.0$ ms (e) $t = 20.0$ ms

Figure 8: Deformed geometry in test CBEA01 (without mesh lines).

4.2.2 Case CBEA02

This test is similar to CBEA02 but uses $\alpha = 1$. The results in terms of deformed mesh are visually very similar to those of case CBEA01 and are not shown for brevity.

A comparison of the results in the two tests in terms of time curves is offered in Figures 9 and 10, where the solution CBEA01 is shown in red while the solution CBEA02 is shown in black.



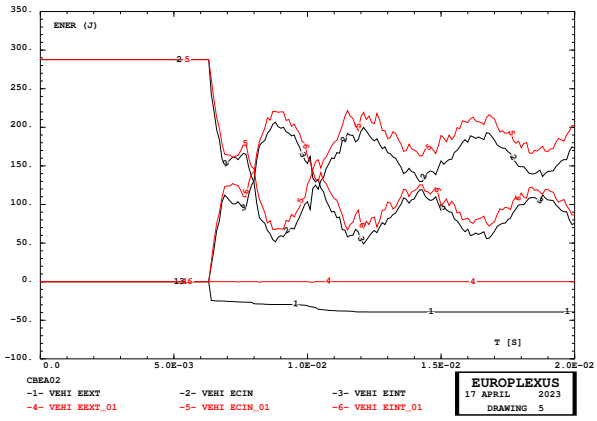
(a) Displacements

(b) Velocities

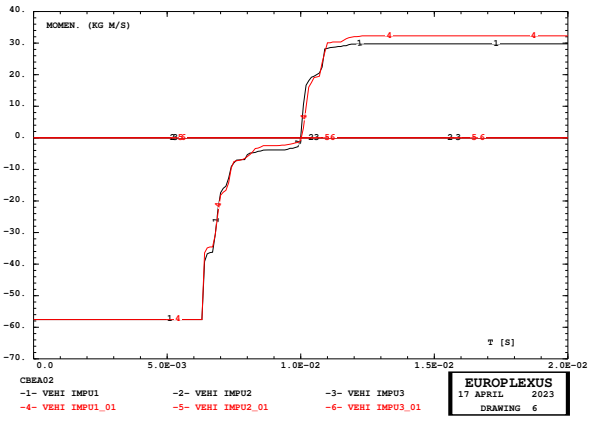
(c) Accelerations

(d) Energies

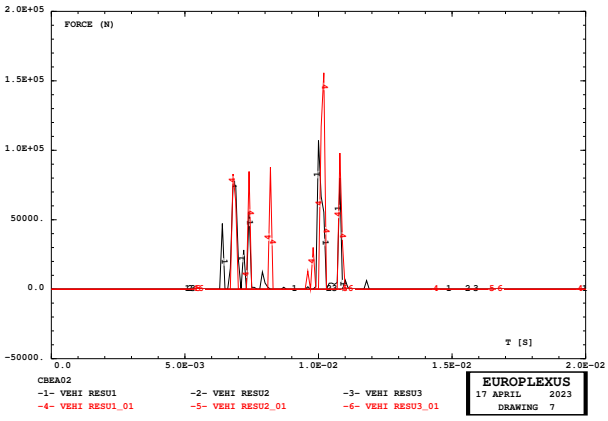
Figure 9: Comparison of results of tests CBEA01 and CBEA02.



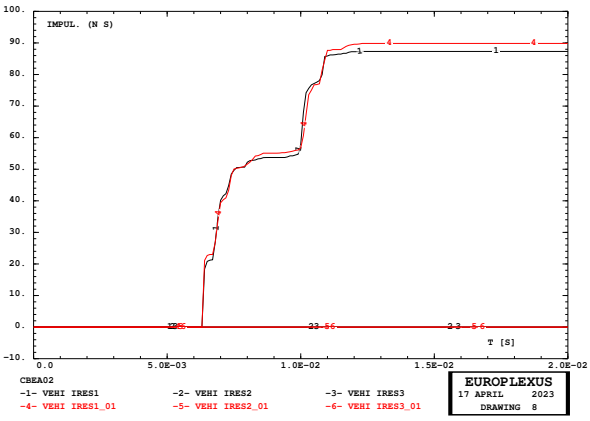
(a) Vehicle energies



(b) Momenta



(c) Vehicle forces



(d) Vehicle impulses

Figure 10: Further comparison of results of tests CBEA01 and CBEA02.

References

[1] EUROPLEXUS User's Manual, on-line version: http://europlexus.jrc.ec.europa.eu.

[2] Cast3m Software: http://www-cast3m.cea.fr/.

[3] M. Sebik, M. Popovic, K. Kleteckova. Generic vehicle model N1. JRC Technical Report 130165, 2022.

[4] https://counterterrorism.jrc.ec.europa.eu/generic_vehicle_models.php.

[5] 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.

[6] F. Casadei. A Revision of the Liaisons Model in EUROPLEXUS for Domain Decomposition Calculations - Second Edition. Technical Note N. I.03.164, November 2003.
Pdf/2003/Link2/Link2.pdf

[7] J.P. Halleux, F. Casadei. Spatial Time Step Partitioning in EUROPLEXUS. EUR Report 22464 EN, 2006.
Pdf/2006/Partition/Partition.pdf

[8] 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.
Pdf/1989/Smirt89_p3c/Smirt89_p3c.pdf

[9] F. Casadei. A Hierarchic Pinball Method for Contact-Impact in Fast Transient Dynamics. VI Congresso Nazionale della Società Italiana di Matematica Applicata e Industriale (SIMAI 2002), Chia (Cagliari), Italy, 27–31 May 2002.
Pdf/2002/Simai2002/Simai2002.pdf

[10] F. Casadei. A General Impact-Contact Algorithm Based on Hierarchic Pinballs for the EUROPLEXUS Software System. Technical Note N. I.03.176, December 2003.
Pdf/2003/Pinballs/Pinballs.pdf

[11] R.C. Fetecau, J.E. Marsden, M. Ortiz and M. West. Non-smooth Lagrangian Mechanics and Variational Collision Integrators. SIAM J. Applied Dynamical Systems, vol. 2, No. 3, pp. 381-416 (2003).

[12] F. Casadei, G. Valsamos, M. Larcher, V. Faucher. Coupled, decoupled and MPI formulations of the DEPL, VITE and ACCE constraints for imposed motions in EUROPLEXUS. JRC Technical Report, PUBSY No. JRC117476, 2019.
Pdf/2018/DVA/Report/Front/Req_JRC117476.pdf

Appendix I — Input files

All the input files used in the previous Sections are listed below.

cbea01.epx

```
CBEA01
ECHO
KFIL 'cbea.k'
TRID LAGR
GEOM SCAL FACT 0.001
  Q4GS she4
  T3GS she3
TERM
COMP GROU 1 'vehi' LECT TOUS TERM
                   COND XB GT -10.E-3
MATE LINE RO 7800. YOUN 210E9 NU 0.28
          LECT PART 700 PART 126 TERM
INIT VITE 1 -10.
     LECT vehi TERM
LIAI LAGC
     BLOQ 123456 LECT PART 700 TERM
     GLIS 1 SELF CMAI LECT PART 700 TERM DIRE
                 PESC LECT PART 126 TERM
OPTI NOTE
     CSTA 0.4
     LOG 1000
     GLIS NORM ELEM
     LIAJ ALPH 0.
*    LIAJ ALPH 1.
REGI 'barr' RMAS BARY WEXT RESU IRES
            LECT PART 700 TERM
     'vehi' RMAS BARY RESU WINT WEXT ECIN
            DMOY VMOY AMOY IMPU IRES
            LECT vehi TERM
ECRI DEPL VITE ACCE PERF TFRE 0.0001
          POIN LECT NSET 32 NSET 33 NSET 34 NSET 35 TERM
          NOEL
     FICH ALIT TFRE 0.0001
          POIN LECT NSET 32 NSET 33 NSET 34 NSET 35 TERM
     FICH SPLI ALIC TFRE 0.001
!    FICH PVTK TFRE 0.001
!         GROU AUTO
!         VARI DEPL ECRO VITE FAIL CONT FLIA DTST
CALC TINI 0 TEND 0.02
SUIT
Post
ECHO
RESU ALIC TEMP GARD PSCR
SORT GRAP AXTE 1. 't [s]'
COUR 1 'N32-D' DEPL COMP 1 NOEU LECT 1   TERM
COUR 2 'N33-D' DEPL COMP 1 NOEU LECT 287 TERM
COUR 3 'N34-D' DEPL COMP 1 NOEU LECT 585 TERM
COUR 4 'N35-D' DEPL COMP 1 NOEU LECT 857 TERM
TRAC 1 2 3 4 AXES 1. 'DISP (m)'
LIST 1 2 3 4 AXES 1. 'DISP (m)'
!
COUR 5 'N32-V' VITE COMP 1 NOEU LECT 1   TERM
COUR 6 'N33-V' VITE COMP 1 NOEU LECT 287 TERM
COUR 7 'N34-V' VITE COMP 1 NOEU LECT 585 TERM
COUR 8 'N35-V' VITE COMP 1 NOEU LECT 857 TERM
TRAC 5 6 7 8 AXES 1. 'VELO (m/s)'
LIST 5 6 7 8 AXES 1. 'VELO (m/s)'
!
COUR  9 'N32-A' ACCE COMP 1 NOEU LECT 1   TERM
COUR 10 'N33-A' ACCE COMP 1 NOEU LECT 287 TERM
COUR 11 'N34-A' ACCE COMP 1 NOEU LECT 585 TERM
COUR 12 'N35-A' ACCE COMP 1 NOEU LECT 857 TERM
TRAC 9 10 11 12 AXES 1. 'ACCE (m/s2)'
LIST 9 10 11 12 AXES 1. 'ACCE (m/s2)'
!
COUR 13 'WTOT' WTOT
COUR 14 'WCIN' WCIN
COUR 15 'WINT' WINT
COUR 16 'WIMP' WIMP
TRAC 13 14 15 16 AXES 1. 'ENER (J)'
LIST 13 14 15 16 AXES 1. 'ENER (J)'
!
COUR 17 'WTOT' WTOT
COUR 18 'WCIN' WCIN
COUR 19 'WINT' WINT
COUR 20 'WIMP' WIMP
TRAC 17 18 19 20 AXES 1. 'ENER (J)'
LIST 17 18 19 20 AXES 1. 'ENER (J)'
!
COUR 21 'vehi EEXT' EEXT REGI 2
COUR 22 'vehi ECIN' ECIN REGI 2
COUR 23 'vehi EINT' EINT REGI 2
TRAC 21 22 23 AXES 1. 'ENER (J)'
LIST 21 22 23 AXES 1. 'ENER (J)'
!
COUR 24 'vehi IMPU1' IMPU COMP 1 REGI 2
COUR 25 'vehi IMPU2' IMPU COMP 2 REGI 2
COUR 26 'vehi IMPU3' IMPU COMP 3 REGI 2
TRAC 24 25 26 AXES 1. 'momen. (kg m/s)'
LIST 24 25 26 AXES 1. 'momen. (kg m/s)'
!
COUR 27 'vehi RESU1' RESU COMP 1 REGI 2
COUR 28 'vehi RESU2' RESU COMP 2 REGI 2
COUR 29 'vehi RESU3' RESU COMP 3 REGI 2
TRAC 27 28 29 AXES 1. 'force (N)'
LIST 27 28 29 AXES 1. 'force (N)'
!
COUR 30 'vehi IRES1' IRES COMP 1 REGI 2
COUR 31 'vehi IRES2' IRES COMP 2 REGI 2
COUR 32 'vehi IRES3' IRES COMP 3 REGI 2
TRAC 30 31 32 AXES 1. 'impul. (N s)'
LIST 30 31 32 AXES 1. 'impul. (N s)'
!
FIN
```

cbea01a.epx

```
CBEA01A
ECHO
!CONV WIN
RESU SPLI ALIC 'cbea01.ali' GARD PSCR
COMP COUL ROUG LECT vehi TERM
          GR50 LECT tous DIFF vehi TERM
SORT VISU NSTO 1
PLAY
CAME   1 EYE   1.07227E+00  4.11384E+00  1.75673E+00
!        Q     1.09578E-01  4.57013E-02  6.06711E-01  7.86006E-01
         VIEW -2.04808E-01 -9.43742E-01 -2.59625E-01
         RIGH -9.71808E-01  2.27713E-01 -6.11218E-02
         UP   -1.16803E-01 -2.39788E-01  9.63773E-01
         FOV   2.48819E+01
!NAVIGATION MODE: ROTATING CAMERA
!CENTER :  1.79499E-01  1.84774E-06  6.25000E-01
!RSPHERE:  1.17813E+00
!RADIUS :  4.35907E+00
!ASPECT :  1.00000E+00
!NEAR   :  3.18095E+00
!FAR    :  6.71533E+00
SCEN GEOM NAVI FREE
          FACE SBAC
          LINE HEOU SSHA SFRE
     LIMA ON
SLER CAM1 1 NFRA 1
FREQ 1
TRAC OFFS FICH AVI NOCL NFTO 21 FPS 10 KFRE 10 COMP -1 REND
GOTR LOOP 19 OFFS FICH AVI CONT NOCL REND
GO
TRAC OFFS FICH AVI CONT REND
ENDP
FIN
```

cbea01z.epx

```
CBEA01Z
ECHO
 CONV WIN
RESU SPLI ALIC 'cbea01.ali' GARD PSCR
SORT VISU NSTO 1
FIN
```

cbea02.epx

```
CBEA02
ECHO
KFIL 'cbea.k'
TRID LAGR
GEOM SCAL FACT 0.001
  Q4GS she4
  T3GS she3
TERM
COMP GROU 1 'vehi' LECT TOUS TERM
                   COND XB GT -10.E-3
MATE LINE RO 7800. YOUN 210E9 NU 0.28
          LECT PART 700 PART 126 TERM
INIT VITE 1 -10.
     LECT vehi TERM
LIAI LAGC
     BLOQ 123456 LECT PART 700 TERM
     GLIS 1 SELF CMAI LECT PART 700 TERM DIRE
                 PESC LECT PART 126 TERM
OPTI NOTE
     CSTA 0.4
     LOG 1000
     GLIS NORM ELEM
*    LIAJ ALPH 0.
     LIAJ ALPH 1.
REGI 'barr' RMAS BARY WEXT RESU IRES
            LECT PART 700 TERM
     'vehi' RMAS BARY RESU WINT WEXT ECIN
            DMOY VMOY AMOY IMPU IRES
            LECT vehi TERM
ECRI DEPL VITE ACCE PERF TFRE 0.0001
          POIN LECT NSET 32 NSET 33 NSET 34 NSET 35 TERM
          NOEL
```

```
     FICH ALIT TFRE 0.0001
          POIN LECT NSET 32 NSET 33 NSET 34 NSET 35 TERM
     FICH SPLI ALIC TFRE 0.001
!    FICH PVTK TFRE 0.001
!         GROU AUTO
!         VARI DEPL ECRO VITE FAIL CONT FLIA DTST
CALC TINI 0 TEND 0.02
SUIT
Post
ECHO
RESU ALIC TEMP GARD PSCR
SORT GRAP AXTE 1. 't [s]'
COUR 1 'N32-D' DEPL COMP 1 NOEU LECT 1   TERM
COUR 2 'N33-D' DEPL COMP 1 NOEU LECT 287 TERM
COUR 3 'N34-D' DEPL COMP 1 NOEU LECT 585 TERM
COUR 4 'N35-D' DEPL COMP 1 NOEU LECT 857 TERM
TRAC 1 2 3 4 AXES 1. 'DISP (m)'
LIST 1 2 3 4 AXES 1. 'DISP (m)'
!
COUR 5 'N32-V' VITE COMP 1 NOEU LECT 1   TERM
COUR 6 'N33-V' VITE COMP 1 NOEU LECT 287 TERM
COUR 7 'N34-V' VITE COMP 1 NOEU LECT 585 TERM
COUR 8 'N35-V' VITE COMP 1 NOEU LECT 857 TERM
TRAC 5 6 7 8 AXES 1. 'VELO (m/s)'
LIST 5 6 7 8 AXES 1. 'VELO (m/s)'
!
COUR  9 'N32-A' ACCE COMP 1 NOEU LECT 1   TERM
COUR 10 'N33-A' ACCE COMP 1 NOEU LECT 287 TERM
COUR 11 'N34-A' ACCE COMP 1 NOEU LECT 585 TERM
COUR 12 'N35-A' ACCE COMP 1 NOEU LECT 857 TERM
TRAC 9 10 11 12 AXES 1. 'ACCE (m/s2)'
LIST 9 10 11 12 AXES 1. 'ACCE (m/s2)'
!
COUR 13 'WTOT' WTOT
COUR 14 'WCIN' WCIN
COUR 15 'WINT' WINT
COUR 16 'WIMP' WIMP
TRAC 13 14 15 16 AXES 1. 'ENER (J)'
LIST 13 14 15 16 AXES 1. 'ENER (J)'
!
COUR 17 'WTOT' WTOT
COUR 18 'WCIN' WCIN
COUR 19 'WINT' WINT
COUR 20 'WIMP' WIMP
TRAC 17 18 19 20 AXES 1. 'ENER (J)'
LIST 17 18 19 20 AXES 1. 'ENER (J)'
!
COUR 21 'vehi EEXT' EEXT REGI 2
COUR 22 'vehi ECIN' ECIN REGI 2
COUR 23 'vehi EINT' EINT REGI 2
TRAC 21 22 23 AXES 1. 'ENER (J)'
LIST 21 22 23 AXES 1. 'ENER (J)'
!
COUR 24 'vehi IMPU1' IMPU COMP 1 REGI 2
COUR 25 'vehi IMPU2' IMPU COMP 2 REGI 2
COUR 26 'vehi IMPU3' IMPU COMP 3 REGI 2
TRAC 24 25 26 AXES 1. 'momen. (kg m/s)'
LIST 24 25 26 AXES 1. 'momen. (kg m/s)'
!
COUR 27 'vehi RESU1' RESU COMP 1 REGI 2
COUR 28 'vehi RESU2' RESU COMP 2 REGI 2
COUR 29 'vehi RESU3' RESU COMP 3 REGI 2
TRAC 27 28 29 AXES 1. 'force (N)'
LIST 27 28 29 AXES 1. 'force (N)'
!
COUR 30 'vehi IRES1' IRES COMP 1 REGI 2
COUR 31 'vehi IRES2' IRES COMP 2 REGI 2
COUR 32 'vehi IRES3' IRES COMP 3 REGI 2
TRAC 30 31 32 AXES 1. 'impul. (N s)'
LIST 30 31 32 AXES 1. 'impul. (N s)'
!
FIN
```

cbea02a.epx

```
CBEA02A
ECHO
!CONV WIN
RESU SPLI ALIC 'cbea02.ali' GARD PSCR
COMP COUL ROUG LECT vehi TERM
          GR50 LECT tous DIFF vehi TERM
SORT VISU NSTO 1
PLAY
CAME   1 EYE   1.07227E+00  4.11384E+00  1.75673E+00
!        Q     1.09578E-01  4.57013E-02  6.06711E-01  7.86006E-01
         VIEW -2.04808E-01 -9.43742E-01 -2.59625E-01
         RIGH -9.71808E-01  2.27713E-01 -6.11218E-02
         UP   -1.16803E-01 -2.39788E-01  9.63773E-01
         FOV   2.48819E+01
!NAVIGATION MODE: ROTATING CAMERA
!CENTER :  1.79499E-01  1.84774E-06  6.25000E-01
!RSPHERE:  1.17813E+00
!RADIUS :  4.35907E+00
!ASPECT :  1.00000E+00
!NEAR   :  3.18095E+00
!FAR    :  6.71533E+00
SCEN GEOM NAVI FREE
          FACE SBAC
          LINE HEOU SSHA SFRE
     LIMA ON
SLER CAM1 1 NFRA 1
FREQ 1
TRAC OFFS FICH AVI NOCL NFTO 21 FPS 10 KFRE 10 COMP -1 REND
GOTR LOOP 19 OFFS FICH AVI CONT NOCL REND
GO
TRAC OFFS FICH AVI CONT REND
ENDP
FIN
```

cbea02c.epx

```
CBEA02C
ECHO
RESU ALIC TEMP 'cbea02.alt' GARD PSCR
SORT GRAP AXTE 1. 't [s]'
COUR 1 'N32-D' DEPL COMP 1 NOEU LECT 1   TERM
COUR 2 'N33-D' DEPL COMP 1 NOEU LECT 287 TERM
COUR 3 'N34-D' DEPL COMP 1 NOEU LECT 585 TERM
COUR 4 'N35-D' DEPL COMP 1 NOEU LECT 857 TERM
RCOU 101 'N32-D' FICH 'cbea01.pun' RENA 'N32-D_01'
RCOU 102 'N33-D' FICH 'cbea01.pun' RENA 'N33-D_01'
RCOU 103 'N34-D' FICH 'cbea01.pun' RENA 'N34-D_01'
RCOU 104 'N35-D' FICH 'cbea01.pun' RENA 'N35-D_01'
TRAC 1 2 3 4 101 102 103 104 AXES 1. 'DISP (m)'
COLO NOIR NOIR NOIR NOIR ROUG ROUG ROUG ROUG
!
COUR 5 'N32-V' VITE COMP 1 NOEU LECT 1   TERM
COUR 6 'N33-V' VITE COMP 1 NOEU LECT 287 TERM
COUR 7 'N34-V' VITE COMP 1 NOEU LECT 585 TERM
COUR 8 'N35-V' VITE COMP 1 NOEU LECT 857 TERM
RCOU 105 'N32-V' FICH 'cbea01.pun' RENA 'N32-V_01'
RCOU 106 'N33-V' FICH 'cbea01.pun' RENA 'N33-V_01'
RCOU 107 'N34-V' FICH 'cbea01.pun' RENA 'N34-V_01'
RCOU 108 'N35-V' FICH 'cbea01.pun' RENA 'N35-V_01'
TRAC 5 6 7 8 105 106 107 108 AXES 1. 'VELO (m/s)'
COLO NOIR NOIR NOIR NOIR ROUG ROUG ROUG ROUG
!
COUR  9 'N32-A' ACCE COMP 1 NOEU LECT 1   TERM
COUR 10 'N33-A' ACCE COMP 1 NOEU LECT 287 TERM
COUR 11 'N34-A' ACCE COMP 1 NOEU LECT 585 TERM
COUR 12 'N35-A' ACCE COMP 1 NOEU LECT 857 TERM
RCOU 109 'N32-A' FICH 'cbea01.pun' RENA 'N32-A_01'
RCOU 110 'N33-A' FICH 'cbea01.pun' RENA 'N33-A_01'
RCOU 111 'N34-A' FICH 'cbea01.pun' RENA 'N34-A_01'
RCOU 112 'N35-A' FICH 'cbea01.pun' RENA 'N35-A_01'
TRAC 9 10 11 12 109 110 111 112 AXES 1. 'ACCE (m/s2)'
COLO NOIR NOIR NOIR NOIR ROUG ROUG ROUG ROUG
!
COUR 13 'WTOT' WTOT
COUR 14 'WCIN' WCIN
COUR 15 'WINT' WINT
COUR 16 'WIMP' WIMP
RCOU 113 'WTOT' FICH 'cbea01.pun' RENA 'WTOT_01'
RCOU 114 'WCIN' FICH 'cbea01.pun' RENA 'WCIN_01'
RCOU 115 'WINT' FICH 'cbea01.pun' RENA 'WINT_01'
RCOU 116 'WIMP' FICH 'cbea01.pun' RENA 'WIMP_01'
TRAC 13 14 15 16 113 114 115 116 AXES 1. 'ENER (J)'
COLO NOIR NOIR NOIR NOIR ROUG ROUG ROUG ROUG
!
COUR 21 'vehi EEXT' EEXT REGI 2
COUR 22 'vehi ECIN' ECIN REGI 2
COUR 23 'vehi EINT' EINT REGI 2
RCOU 121 'vehi EEXT' FICH 'cbea01.pun' RENA 'vehi EEXT_01'
RCOU 122 'vehi ECIN' FICH 'cbea01.pun' RENA 'vehi ECIN_01'
RCOU 123 'vehi EINT' FICH 'cbea01.pun' RENA 'vehi EINT_01'
TRAC 21 22 23 121 122 123 AXES 1. 'ENER (J)'
COLO NOIR NOIR NOIR ROUG ROUG ROUG
!
COUR 24 'vehi IMPU1' IMPU COMP 1 REGI 2
COUR 25 'vehi IMPU2' IMPU COMP 2 REGI 2
COUR 26 'vehi IMPU3' IMPU COMP 3 REGI 2
RCOU 124 'vehi IMPU1' FICH 'cbea01.pun' RENA 'vehi IMPU1_01'
RCOU 125 'vehi IMPU2' FICH 'cbea01.pun' RENA 'vehi IMPU2_01'
RCOU 126 'vehi IMPU3' FICH 'cbea01.pun' RENA 'vehi IMPU3_01'
TRAC 24 25 26 124 125 126 AXES 1. 'momen. (kg m/s)'
COLO NOIR NOIR NOIR ROUG ROUG ROUG
!
COUR 27 'vehi RESU1' RESU COMP 1 REGI 2
COUR 28 'vehi RESU2' RESU COMP 2 REGI 2
COUR 29 'vehi RESU3' RESU COMP 3 REGI 2
RCOU 127 'vehi RESU1' FICH 'cbea01.pun' RENA 'vehi RESU1_01'
RCOU 128 'vehi RESU2' FICH 'cbea01.pun' RENA 'vehi RESU2_01'
RCOU 129 'vehi RESU3' FICH 'cbea01.pun' RENA 'vehi RESU3_01'
TRAC 27 28 29 127 128 129 AXES 1. 'force (N)'
COLO NOIR NOIR NOIR ROUG ROUG ROUG
!
COUR 30 'vehi IRES1' IRES COMP 1 REGI 2
COUR 31 'vehi IRES2' IRES COMP 2 REGI 2
COUR 32 'vehi IRES3' IRES COMP 3 REGI 2
RCOU 130 'vehi IRES1' FICH 'cbea01.pun' RENA 'vehi IRES1_01'
RCOU 131 'vehi IRES2' FICH 'cbea01.pun' RENA 'vehi IRES2_01'
RCOU 132 'vehi IRES3' FICH 'cbea01.pun' RENA 'vehi IRES3_01'
TRAC 30 31 32 130 131 132 AXES 1. 'impul. (N s)'
COLO NOIR NOIR NOIR ROUG ROUG ROUG
!
FIN
```

cbea02z.epx

```
CBEA02Z
ECHO
 CONV WIN
RESU SPLI ALIC 'cbea02.ali' GARD PSCR
SORT VISU NSTO 1
FIN
```

pinb03.dgibi

```
opti donn 'pxordpoi.proc';
*
*opti titr 'PINB - 03';
opti dime 2 elem qua4;
*
p1=-201 -1;
p2=-1 -1;
p3=1 -1;
p4=201 -1;
p5=-101 -1;
p6=102 -1;
*
tol=0.001;
*
c1=p1 d 100 p2;
c2=p3 d 100 p4;
stru1=c1 TRAN 1 (0 2);
stru2=c2 TRAN 1 (0 2);
stru = stru1 et stru2;
elim tol (stru et p5 et p6);
*
pin1 = stru elem cont p2;
pin2 = stru elem cont p3;
pin = pin1 et pin2;
mid1 = stru elem cont p5;
mid2 = stru elem cont p6;
*
xp = stru poin droi p1 p4 tol;
xpln = pxordpoi xp p1;
*
mesh = stru et xpln;
*
tass mesh;
*
opti sauv form 'pinb03.msh';
sauv form mesh;
*
opti trac psc;
trac mesh;
trac qual pin;
list pin;
fin;
```

pinb03.epx

```
PINB03
*
ECHO
*CONV win
*
CAST MESH
*
LAGR CPLA LAGC
*
*opti dump
*
GEOM Q41L stru TERM
COMP EPAI 1. LECT stru TERM
     COUL VERT LECT stru TERM
MATE VM23 RO 8000. YOUN 2.E11 NU 0.0 ELAS 2.E11
          TRAC 1 2.E11 1.D0
          LECT stru TERM
*
opti pins reb2
LIAI PINB BODY LECT pin1 TERM
          BODY LECT pin2 TERM
*
INIT VITE 1  500 LECT stru1 TERM
     VITE 1 -500 LECT stru2 TERM
*
ECRI DEPL VITE ACCE FINT FEXT CONT ECRO TFRE 10.E-3
     fich tplo desc 'PINB03' freq 1
                    poin lect p1 p2 p3 p4 term
                    elem lect pin mid1 mid2 term
     fich xplo desc 'PINB03' tfre 10.e-3
                    poin lect xpln term
     fich alic temp FREQ 1
                    poin lect p1 p2 p3 p4 term
                    elem lect pin mid1 mid2 term
*
OPTI PAS AUTO NOTE STEP IO
 csta 0.5E0
 log 1
 liaj
*
CALCUL TINI 0. TEND 100.E-3
*=============================================================
SUIT
Post-treatment
ECHO
*
RESU ALIC TEMP GARD PSCR
*
SORT GRAP
*
AXTE 1.0 'Time [s]'
*
COUR 1 'dx_p1'  DEPL COMP 1 NOEU LECT p1 TERM
COUR 2 'dx_p2'  DEPL COMP 1 NOEU LECT p2 TERM
COUR 3 'dx_p3'  DEPL COMP 1 NOEU LECT p3 TERM
COUR 4 'dx_p4'  DEPL COMP 1 NOEU LECT p4 TERM
COUR 5 'fex_p2'  FORC COMP 1 NOEU LECT p2 TERM
COUR 6 'fex_p3'  FORC COMP 1 NOEU LECT p3 TERM
COUR 7 'sx_mid1'  CONT COMP 1 ELEM LECT mid1 TERM
COUR 8 'sx_mid2'  CONT COMP 1 ELEM LECT mid2 TERM
COUR 11 'wtot' WTOT
COUR 12 'wcin' WCIN
COUR 13 'wint' WINT
COUR 14 'wext' WEXT
COUR 15 'bila' BILA
*
trac 1 2 3 4 axes 1.0 'DISPL. [M]' YZER
trac 5 6 axes 1.0 'FEXT [N]'
COLO ROUG VERT
trac 7 8 axes 1.0 'SX [PA]' YZER
TRAC 11 12 13 14 AXES 1.0 'Energy [J]' YZER
COLO NOIR ROUG VERT TURQ
TRAC 15 AXES 1.0 'Balance [%]' YZER
TRAC 11 12 13 14 AXES 1.0 'Energy [J]' YZER
                                    XMIN 0.0 XMAX 5.E-3 NX 10
COLO NOIR ROUG VERT TURQ
*
QUAL DEPL COMP 1 LECT p2 TERM REFE -9.67354E+0 TOLE 5.E-3
     DEPL COMP 1 LECT p3 TERM REFE  9.67354E+0 TOLE 5.E-3
     VITE COMP 1 LECT p2 TERM REFE -5.75553E+2 TOLE 5.E-3
     VITE COMP 1 LECT p3 TERM REFE  5.75553E+2 TOLE 5.E-3
*=============================================================
FIN
```

pinb13.dgibi

```
opti donn 'pxordpoi.proc';
*
*opti titr 'PINB - 03';
opti dime 2 elem qua4;
*
p1=-201 -1;
p2=-1 -1;
p3=1 -1;
p4=201 -1;
p5=-101 -1;
p6=102 -1;
*
tol=0.001;
*
c1=p1 d 100 p2;
c2=p3 d 100 p4;
stru1=c1 TRAN 1 (0 2);
stru2=c2 TRAN 1 (0 2);
stru = stru1 et stru2;
elim tol (stru et p5 et p6);
*
pin1 = stru elem cont p2;
pin2 = stru elem cont p3;
pin = pin1 et pin2;
mid1 = stru elem cont p5;
mid2 = stru elem cont p6;
*
xp = stru poin droi p1 p4 tol;
xpln = pxordpoi xp p1;
*
mesh = stru et xpln;
*
tass mesh;
*
opti sauv form 'pinb13.msh';
sauv form mesh;
*
opti trac psc;
trac mesh;
trac qual pin;
list pin;
fin;
```

pinb13.epx

```
PINB13
*
ECHO
*CONV win
*
CAST MESH
*
LAGR CPLA
*
*opti dump
*
GEOM Q41L stru TERM
COMP EPAI 1. LECT stru TERM
     COUL VERT LECT stru TERM
MATE VM23 RO 8000. YOUN 2.E11 NU 0.0 ELAS 2.E11
          TRAC 1 2.E11 1.D0
```

```
         LECT stru TERM
*
opti pins reb2
LINK COUP
     PINB BODY LECT pin1 TERM
          BODY LECT pin2 TERM
*
INIT VITE 1  500 LECT stru1 TERM
     VITE 1 -500 LECT stru2 TERM
*
ECRI DEPL VITE ACCE FINT FEXT CONT ECRO TFRE 10.E-3
     fich alic temp FREQ 1
                    poin lect p1 p2 p3 p4 term
                    elem lect pin mid1 mid2 term
*
OPTI PAS AUTO NOTE STEP IO
 csta 0.5E0
 log 1
 liaj
 LNKS STAT
*
CALCUL TINI 0. TEND 100.E-3
*============================================================
SUIT
Post-treatment
ECHO
*
RESU ALIC TEMP GARD PSCR
*
SORT GRAP
*
AXTE 1.0 'Time [s]'
*
COUR 1 'dx_p1'  DEPL COMP 1 NOEU LECT p1 TERM
COUR 2 'dx_p2'  DEPL COMP 1 NOEU LECT p2 TERM
COUR 3 'dx_p3'  DEPL COMP 1 NOEU LECT p3 TERM
COUR 4 'dx_p4'  DEPL COMP 1 NOEU LECT p4 TERM
COUR 5 'fex_p2'  FORC COMP 1 NOEU LECT p2 TERM
COUR 6 'fex_p3'  FORC COMP 1 NOEU LECT p3 TERM
COUR 7 'sx_mid1'  CONT COMP 1 ELEM LECT mid1 TERM
COUR 8 'sx_mid2'  CONT COMP 1 ELEM LECT mid2 TERM
COUR 11 'wtot' WTOT
COUR 12 'wcin' WCIN
COUR 13 'wint' WINT
COUR 14 'wext' WEXT
COUR 15 'bila' BILA
*
trac 1 2 3 4 axes 1.0 'DISPL. [M]' YZER
trac 5 6 axes 1.0 'FEXT [N]'
COLO ROUG VERT
trac 7 8 axes 1.0 'SX [PA]' YZER
TRAC 11 12 13 14 AXES 1.0 'Energy [J]' YZER
COLO NOIR ROUG VERT TURQ
TRAC 15 AXES 1.0 'Balance [%]' YZER
TRAC 11 12 13 14 AXES 1.0 'Energy [J]' YZER
                                  XMIN 0.0 XMAX 5.E-3 NX 10
COLO NOIR ROUG VERT TURQ
*
QUAL DEPL COMP 1 LECT p2 TERM REFE -9.67354E+0 TOLE 5.E-3
     DEPL COMP 1 LECT p3 TERM REFE  9.67354E+0 TOLE 5.E-3
     VITE COMP 1 LECT p2 TERM REFE -5.75553E+2 TOLE 5.E-3
     VITE COMP 1 LECT p3 TERM REFE  5.75553E+2 TOLE 5.E-3
*============================================================
FIN
```

pinb23.dgibi

```
opti donn 'pxordpoi.proc';
*
*opti titr 'PINB - 03';
opti dime 2 elem qua4;
*
p1=-201 -1;
p2=-1 -1;
p3=1 -1;
p4=201 -1;
p5=-101 -1;
p6=102 -1;
*
tol=0.001;
*
c1=p1 d 100 p2;
c2=p3 d 100 p4;
stru1=c1 TRAN 1 (0 2);
stru2=c2 TRAN 1 (0 2);
stru = stru1 et stru2;
elim tol (stru et p5 et p6);
*
pin1 = stru elem cont p2;
pin2 = stru elem cont p3;
pin = pin1 et pin2;
mid1 = stru elem cont p5;
mid2 = stru elem cont p6;
*
xp = stru poin droi p1 p4 tol;
xpln = pxordpoi xp p1;
*
p0 = 0 0;
pm = manu poi1 p0;
*
mesh = stru et xpln et pm;
*
tass mesh;
*
opti sauv form 'pinb23.msh';
sauv form mesh;
*
opti trac psc;
trac mesh;
trac qual pin;
list pin;
fin;
```

pinb23.epx

```
PINB23
*
ECHO
!CONV win
*
CAST MESH
*
LAGR CPLA LAGC
*
*opti dump
*
GEOM Q41L stru PMAT pm TERM
COMP EPAI  1. LECT stru TERM
          0.5 LECT pm TERM
     COUL VERT LECT stru TERM
          ROUG LECT pm TERM
MATE VM23 RO 8000. YOUN 2.E11 NU 0.0 ELAS 2.E11
          TRAC 1 2.E11 1.D0
          LECT stru TERM
     FANT 0.1 LECT pm TERM
*
opti pins reb2
LIAI PINB BODY LECT pin1 TERM
          BODY LECT pin2 TERM
     BLOQ 12 LECT pm TERM
*
INIT VITE 1  500 LECT stru1 TERM
     VITE 1 -500 LECT stru2 TERM
*
ECRI DEPL VITE ACCE FINT FEXT CONT ECRO TFRE 10.E-3
     fich tplo desc 'PINB23' freq 1
                    poin lect p1 p2 p3 p4 term
                    elem lect pin mid1 mid2 term
     fich xplo desc 'PINB23' tfre 10.e-3
                    poin lect xpln term
     fich alic temp FREQ 1
                    poin lect p1 p2 p3 p4 term
                    elem lect pin mid1 mid2 term
*
OPTI PAS AUTO NOTE STEP IO
 csta 0.5E0
 log 1
 liaj
*
CALCUL TINI 0. TEND 100.E-3
*============================================================
SUIT
Post-treatment
ECHO
*
RESU ALIC TEMP GARD PSCR
*
SORT GRAP
*
AXTE 1.0 'Time [s]'
*
COUR 1 'dx_p1'  DEPL COMP 1 NOEU LECT p1 TERM
COUR 2 'dx_p2'  DEPL COMP 1 NOEU LECT p2 TERM
COUR 3 'dx_p3'  DEPL COMP 1 NOEU LECT p3 TERM
COUR 4 'dx_p4'  DEPL COMP 1 NOEU LECT p4 TERM
COUR 5 'fex_p2'  FORC COMP 1 NOEU LECT p2 TERM
COUR 6 'fex_p3'  FORC COMP 1 NOEU LECT p3 TERM
COUR 7 'sx_mid1'  CONT COMP 1 ELEM LECT mid1 TERM
COUR 8 'sx_mid2'  CONT COMP 1 ELEM LECT mid2 TERM
COUR 11 'wtot' WTOT
COUR 12 'wcin' WCIN
COUR 13 'wint' WINT
COUR 14 'wext' WEXT
COUR 15 'bila' BILA
*
trac 1 2 3 4 axes 1.0 'DISPL. [M]' YZER
trac 5 6 axes 1.0 'FEXT [N]'
COLO ROUG VERT
trac 7 8 axes 1.0 'SX [PA]' YZER
TRAC 11 12 13 14 AXES 1.0 'Energy [J]' YZER
COLO NOIR ROUG VERT TURQ
TRAC 15 AXES 1.0 'Balance [%]' YZER
TRAC 11 12 13 14 AXES 1.0 'Energy [J]' YZER
                                  XMIN 0.0 XMAX 5.E-3 NX 10
COLO NOIR ROUG VERT TURQ
*
QUAL DEPL COMP 1 LECT p2 TERM REFE -9.67354E+0 TOLE 5.E-3
     DEPL COMP 1 LECT p3 TERM REFE  9.67354E+0 TOLE 5.E-3
     VITE COMP 1 LECT p2 TERM REFE -5.75553E+2 TOLE 5.E-3
     VITE COMP 1 LECT p3 TERM REFE  5.75553E+2 TOLE 5.E-3
*============================================================
FIN
```

pinb33.dgibi

```
opti donn 'pxordpoi.proc';
*
*opti titr 'PINB - 03';
opti dime 2 elem qua4;
*
p1=-201 -1;
p2=-1 -1;
p3=1 -1;
p4=201 -1;
p5=-101 -1;
p6=102 -1;
*
tol=0.001;
*
c1=p1 d 100 p2;
c2=p3 d 100 p4;
stru1=c1 TRAN 1 (0 2);
stru2=c2 TRAN 1 (0 2);
stru = stru1 et stru2;
elim tol (stru et p5 et p6);
*
pin1 = stru elem cont p2;
pin2 = stru elem cont p3;
pin = pin1 et pin2;
mid1 = stru elem cont p5;
mid2 = stru elem cont p6;
*
xp = stru poin droi p1 p4 tol;
xpln = pxordpoi xp p1;
*
p0 = 0 0;
pm = manu poi1 p0;
*
mesh = stru et xpln et pm;
*
tass mesh;
*
opti sauv form 'pinb33.msh';
sauv form mesh;
*
opti trac psc;
trac mesh;
trac qual pin;
list pin;
fin;
```

pinb33.epx

```
PINB33
*
ECHO
*CONV win
*
CAST MESH
*
LAGR CPLA
*
*opti dump
*
GEOM Q41L stru PMAT pm TERM
COMP EPAI  1. LECT stru TERM
          0.5 LECT pm TERM
     COUL VERT LECT stru TERM
          ROUG LECT pm TERM
MATE VM23 RO 8000. YOUN 2.E11 NU 0.0 ELAS 2.E11
          TRAC 1 2.E11 1.D0
          LECT stru TERM
     FANT 0.1 LECT pm TERM
*
opti pins reb2
LINK COUP SPLT NONE
     PINB BODY LECT pin1 TERM
          BODY LECT pin2 TERM
     BLOQ 12 LECT pm TERM
*
INIT VITE 1  500 LECT stru1 TERM
     VITE 1 -500 LECT stru2 TERM
*
ECRI DEPL VITE ACCE FINT FEXT CONT ECRO TFRE 10.E-3
     fich alic temp FREQ 1
                    poin lect p1 p2 p3 p4 term
                    elem lect pin mid1 mid2 term
*
OPTI PAS AUTO NOTE STEP IO
 csta 0.5E0
 log 1
 liaj
 LNKS STAT
*
CALCUL TINI 0. TEND 100.E-3
*=============================================================
SUIT
Post-treatment
ECHO
*
RESU ALIC TEMP GARD PSCR
*
SORT GRAP
*
AXTE 1.0 'Time [s]'
*
COUR 1 'dx_p1'  DEPL COMP 1 NOEU LECT p1 TERM
COUR 2 'dx_p2'  DEPL COMP 1 NOEU LECT p2 TERM
COUR 3 'dx_p3'  DEPL COMP 1 NOEU LECT p3 TERM
COUR 4 'dx_p4'  DEPL COMP 1 NOEU LECT p4 TERM
COUR 5 'fex_p2'  FORC COMP 1 NOEU LECT p2 TERM
COUR 6 'fex_p3'  FORC COMP 1 NOEU LECT p3 TERM
COUR 7 'sx_mid1'  CONT COMP 1 ELEM LECT mid1 TERM
COUR 8 'sx_mid2'  CONT COMP 1 ELEM LECT mid2 TERM
COUR 11 'wtot' WTOT
COUR 12 'wcin' WCIN
COUR 13 'wint' WINT
COUR 14 'wext' WEXT
COUR 15 'bila' BILA
*
trac 1 2 3 4 axes 1.0 'DISPL. [M]' YZER
trac 5 6 axes 1.0 'FEXT [N]'
COLO ROUG VERT
trac 7 8 axes 1.0 'SX [PA]' YZER
TRAC 11 12 13 14 AXES 1.0 'Energy [J]' YZER
COLO NOIR ROUG VERT TURQ
TRAC 15 AXES 1.0 'Balance [%]' YZER
TRAC 11 12 13 14 AXES 1.0 'Energy [J]' YZER
                                    XMIN 0.0 XMAX 5.E-3 NX 10
COLO NOIR ROUG VERT TURQ
*
QUAL DEPL COMP 1 LECT p2 TERM REFE -9.67354E+0 TOLE 5.E-3
     DEPL COMP 1 LECT p3 TERM REFE  9.67354E+0 TOLE 5.E-3
     VITE COMP 1 LECT p2 TERM REFE -5.75553E+2 TOLE 5.E-3
     VITE COMP 1 LECT p3 TERM REFE  5.75553E+2 TOLE 5.E-3
*=============================================================
FIN
```

pinb43.dgibi

```
opti donn 'pxordpoi.proc';
*
*opti titr 'PINB - 03';
opti dime 2 elem qua4;
*
p1=-201 -1;
p2=-1 -1;
p3=1 -1;
p4=201 -1;
p5=-101 -1;
p6=102 -1;
*
tol=0.001;
*
c1=p1 d 100 p2;
c2=p3 d 100 p4;
stru1=c1 TRAN 1 (0 2);
stru2=c2 TRAN 1 (0 2);
stru = stru1 et stru2;
elim tol (stru et p5 et p6);
*
pin1 = stru elem cont p2;
pin2 = stru elem cont p3;
pin = pin1 et pin2;
mid1 = stru elem cont p5;
mid2 = stru elem cont p6;
*
xp = stru poin droi p1 p4 tol;
xpln = pxordpoi xp p1;
*
mesh = stru et xpln;
*
tass mesh;
*
opti sauv form 'pinb43.msh';
sauv form mesh;
*
opti trac psc;
trac mesh;
trac qual pin;
list pin;
fin;
```

pinb43.epx

```
PINB43
*
ECHO
*CONV win
*
CAST MESH
*
LAGR CPLA LAGC
*
*opti dump
*
GEOM Q41L stru TERM
COMP EPAI 1. LECT stru TERM
     COUL VERT LECT stru TERM
MATE VM23 RO 8000. YOUN 2.E11 NU 0.0 ELAS 2.E11
          TRAC 1 2.E11 1.D0
          LECT stru TERM
*
LIAI PINB BODY LECT pin1 TERM
```

```
         BODY LECT pin2 TERM
*
INIT VITE 1  500 LECT stru1 TERM
     VITE 1 -500 LECT stru2 TERM
*
ECRI DEPL VITE ACCE FINT FEXT CONT ECRO TFRE 10.E-3
     fich tplo desc 'PINB43' freq 1
                    poin lect p1 p2 p3 p4 term
                    elem lect pin mid1 mid2 term
     fich xplo desc 'PINB43' tfre 10.e-3
                    poin lect xpln term
     fich alic temp FREQ 1
                    poin lect p1 p2 p3 p4 term
                    elem lect pin mid1 mid2 term
*
OPTI PAS AUTO NOTE STEP IO
 csta 0.5E0
 log 1
 liaj
*
CALCUL TINI 0. TEND 100.E-3
*============================================================
SUIT
Post-treatment
ECHO
*
RESU ALIC TEMP GARD PSCR
*
SORT GRAP
*
AXTE 1.0 'Time [s]'
*
COUR 1 'dx_p1'  DEPL COMP 1 NOEU LECT p1 TERM
COUR 2 'dx_p2'  DEPL COMP 1 NOEU LECT p2 TERM
COUR 3 'dx_p3'  DEPL COMP 1 NOEU LECT p3 TERM
COUR 4 'dx_p4'  DEPL COMP 1 NOEU LECT p4 TERM
COUR 5 'fex_p2'  FORC COMP 1 NOEU LECT p2 TERM
COUR 6 'fex_p3'  FORC COMP 1 NOEU LECT p3 TERM
COUR 7 'sx_mid1'  CONT COMP 1 ELEM LECT mid1 TERM
COUR 8 'sx_mid2'  CONT COMP 1 ELEM LECT mid2 TERM
COUR 11 'wtot' WTOT
COUR 12 'wcin' WCIN
COUR 13 'wint' WINT
COUR 14 'wext' WEXT
COUR 15 'bila' BILA
*
trac 1 2 3 4 axes 1.0 'DISPL. [M]' YZER
trac 5 6 axes 1.0 'FEXT [N]'
COLO ROUG VERT
trac 7 8 axes 1.0 'SX [PA]' YZER
TRAC 11 12 13 14 AXES 1.0 'Energy [J]' YZER
COLO NOIR ROUG VERT TURQ
TRAC 15 AXES 1.0 'Balance [%]' YZER
TRAC 11 12 13 14 AXES 1.0 'Energy [J]' YZER
                                    XMIN 0.0 XMAX 5.E-3 NX 10
COLO NOIR ROUG VERT TURQ
*
QUAL DEPL COMP 1 LECT p2 TERM REFE -9.67354E+0 TOLE 5.E-3
     DEPL COMP 1 LECT p3 TERM REFE  9.67354E+0 TOLE 5.E-3
     VITE COMP 1 LECT p2 TERM REFE -5.75553E+2 TOLE 5.E-3
     VITE COMP 1 LECT p3 TERM REFE  5.75553E+2 TOLE 5.E-3
*============================================================
FIN
```

pinb53.dgibi

```
opti donn 'pxordpoi.proc';
*
*opti titr 'PINB - 03';
opti dime 2 elem qua4;
*
p1=-201 -1;
p2=-1 -1;
p3=1 -1;
p4=201 -1;
p5=-101 -1;
p6=102 -1;
*
tol=0.001;
*
c1=p1 d 100 p2;
c2=p3 d 100 p4;
stru1=c1 TRAN 1 (0 2);
stru2=c2 TRAN 1 (0 2);
stru = stru1 et stru2;
elim tol (stru et p5 et p6);
*
pin1 = stru elem cont p2;
pin2 = stru elem cont p3;
pin = pin1 et pin2;
mid1 = stru elem cont p5;
mid2 = stru elem cont p6;
*
xp = stru poin droi p1 p4 tol;
xpln = pxordpoi xp p1;
*
p0 = 0 0;
pm = manu poi1 p0;
*
mesh = stru et xpln et pm;
*
tass mesh;
*
opti sauv form 'pinb53.msh';
sauv form mesh;
*
opti trac psc;
trac mesh;
trac qual pin;
list pin;
fin;
```

pinb53.epx

```
PINB53
*
ECHO
*CONV win
*
CAST MESH
*
LAGR CPLA
*
*opti dump
*
GEOM Q41L stru PMAT pm TERM
COMP EPAI  1. LECT stru TERM
           0.5 LECT pm TERM
     COUL VERT LECT stru TERM
          ROUG LECT pm TERM
MATE VM23 RO 8000. YOUN 2.E11 NU 0.0 ELAS 2.E11
          TRAC 1 2.E11 1.D0
          LECT stru TERM
     FANT 0.1 LECT pm TERM
*
opti pins reb2
LINK COUP SPLT NONE
     PINB BODY LECT pin1 TERM
          BODY LECT pin2 TERM
     BLOQ 12 LECT pm TERM
*
INIT VITE 1  500 LECT stru1 TERM
     VITE 1 -500 LECT stru2 TERM
*
ECRI DEPL VITE ACCE FINT FEXT CONT ECRO TFRE 10.E-3
     fich alic temp FREQ 1
                    poin lect p1 p2 p3 p4 term
                    elem lect pin mid1 mid2 term
*
OPTI PAS AUTO NOTE STEP IO
 csta 0.5E0
 log 1
 liaj alph 0.D0
 LNKS STAT
*
CALCUL TINI 0. TEND 100.E-3
*============================================================
SUIT
Post-treatment
ECHO
*
RESU ALIC TEMP GARD PSCR
*
SORT GRAP
*
AXTE 1.0 'Time [s]'
*
COUR 1 'dx_p1'  DEPL COMP 1 NOEU LECT p1 TERM
COUR 2 'dx_p2'  DEPL COMP 1 NOEU LECT p2 TERM
COUR 3 'dx_p3'  DEPL COMP 1 NOEU LECT p3 TERM
COUR 4 'dx_p4'  DEPL COMP 1 NOEU LECT p4 TERM
COUR 5 'fex_p2'  FORC COMP 1 NOEU LECT p2 TERM
COUR 6 'fex_p3'  FORC COMP 1 NOEU LECT p3 TERM
COUR 7 'sx_mid1'  CONT COMP 1 ELEM LECT mid1 TERM
COUR 8 'sx_mid2'  CONT COMP 1 ELEM LECT mid2 TERM
COUR 11 'wtot' WTOT
COUR 12 'wcin' WCIN
COUR 13 'wint' WINT
COUR 14 'wext' WEXT
COUR 15 'bila' BILA
*
trac 1 2 3 4 axes 1.0 'DISPL. [M]' YZER
trac 5 6 axes 1.0 'FEXT [N]'
COLO ROUG VERT
trac 7 8 axes 1.0 'SX [PA]' YZER
TRAC 11 12 13 14 AXES 1.0 'Energy [J]' YZER
COLO NOIR ROUG VERT TURQ
TRAC 15 AXES 1.0 'Balance [%]' YZER
TRAC 11 12 13 14 AXES 1.0 'Energy [J]' YZER
                                    XMIN 0.0 XMAX 5.E-3 NX 10
COLO NOIR ROUG VERT TURQ
*
QUAL DEPL COMP 1 LECT p2 TERM REFE -9.67354E+0 TOLE 5.E-3
     DEPL COMP 1 LECT p3 TERM REFE  9.67354E+0 TOLE 5.E-3
     VITE COMP 1 LECT p2 TERM REFE -5.75553E+2 TOLE 5.E-3
     VITE COMP 1 LECT p3 TERM REFE  5.75553E+2 TOLE 5.E-3
*============================================================
FIN
```

pinb63.dgibi

```
opti donn 'pxordpoi.proc';
*
*opti titr 'PINB - 03';
opti dime 2 elem qua4;
*
p1=-201 -1;
p2=-1 -1;
p3=1 -1;
p4=201 -1;
p5=-101 -1;
p6=102 -1;
*
tol=0.001;
*
c1=p1 d 100 p2;
c2=p3 d 100 p4;
stru1=c1 TRAN 1 (0 2);
stru2=c2 TRAN 1 (0 2);
stru = stru1 et stru2;
elim tol (stru et p5 et p6);
*
pin1 = stru elem cont p2;
pin2 = stru elem cont p3;
pin = pin1 et pin2;
mid1 = stru elem cont p5;
mid2 = stru elem cont p6;
*
xp = stru poin droi p1 p4 tol;
xpln = pxordpoi xp p1;
*
mesh = stru et xpln;
*
tass mesh;
*
opti sauv form 'pinb63.msh';
sauv form mesh;
*
opti trac psc;
trac mesh;
trac qual pin;
list pin;
fin;
```

pinb63.epx

```
PINB63
*
ECHO
*CONV win
*
CAST MESH
*
LAGR CPLA
*
*opti dump
*
GEOM Q41L stru TERM
COMP EPAI 1. LECT stru TERM
     COUL VERT LECT stru TERM
MATE VM23 RO 8000. YOUN 2.E11 NU 0.0 ELAS 2.E11
          TRAC 1 2.E11 1.D0
          LECT stru TERM
*
LINK COUP
     PINB BODY LECT pin1 TERM
          BODY LECT pin2 TERM
*
INIT VITE 1  500 LECT stru1 TERM
     VITE 1 -500 LECT stru2 TERM
*
ECRI DEPL VITE ACCE FINT FEXT CONT ECRO TFRE 10.E-3
     fich alic temp FREQ 1
                    poin lect p1 p2 p3 p4 term
                    elem lect pin mid1 mid2 term
*
OPTI PAS AUTO NOTE STEP IO
 csta 0.5E0
 log 1
 liaj
 LNKS STAT
*
CALCUL TINI 0. TEND 100.E-3
*=============================================================
SUIT
Post-treatment
ECHO
*
RESU ALIC TEMP GARD PSCR
*
SORT GRAP
*
AXTE 1.0 'Time [s]'
*
COUR 1 'dx_p1'  DEPL COMP 1 NOEU LECT p1 TERM
COUR 2 'dx_p2'  DEPL COMP 1 NOEU LECT p2 TERM
COUR 3 'dx_p3'  DEPL COMP 1 NOEU LECT p3 TERM
COUR 4 'dx_p4'  DEPL COMP 1 NOEU LECT p4 TERM
COUR 5 'fex_p2'  FORC COMP 1 NOEU LECT p2 TERM
COUR 6 'fex_p3'  FORC COMP 1 NOEU LECT p3 TERM
COUR 7 'sx_mid1'  CONT COMP 1 ELEM LECT mid1 TERM
COUR 8 'sx_mid2'  CONT COMP 1 ELEM LECT mid2 TERM
COUR 11 'wtot' WTOT
COUR 12 'wcin' WCIN
COUR 13 'wint' WINT
COUR 14 'wext' WEXT
COUR 15 'bila' BILA
*
trac 1 2 3 4 axes 1.0 'DISPL. [M]' YZER
trac 5 6 axes 1.0 'FEXT [N]'
COLO ROUG VERT
trac 7 8 axes 1.0 'SX [PA]' YZER
TRAC 11 12 13 14 AXES 1.0 'Energy [J]' YZER
COLO NOIR ROUG VERT TURQ
TRAC 15 AXES 1.0 'Balance [%]' YZER
TRAC 11 12 13 14 AXES 1.0 'Energy [J]' YZER
                                  XMIN 0.0 XMAX 5.E-3 NX 10
COLO NOIR ROUG VERT TURQ
*
QUAL DEPL COMP 1 LECT p2 TERM REFE -9.67354E+0 TOLE 5.E-3
     DEPL COMP 1 LECT p3 TERM REFE  9.67354E+0 TOLE 5.E-3
     VITE COMP 1 LECT p2 TERM REFE -5.75553E+2 TOLE 5.E-3
     VITE COMP 1 LECT p3 TERM REFE  5.75553E+2 TOLE 5.E-3
*=============================================================
FIN
```

pinb73.dgibi

```
opti donn 'pxordpoi.proc';
*
*opti titr 'PINB - 03';
opti dime 2 elem qua4;
*
p1=-201 -1;
p2=-1 -1;
p3=1 -1;
p4=201 -1;
p5=-101 -1;
p6=102 -1;
*
tol=0.001;
*
c1=p1 d 100 p2;
c2=p3 d 100 p4;
stru1=c1 TRAN 1 (0 2);
stru2=c2 TRAN 1 (0 2);
stru = stru1 et stru2;
elim tol (stru et p5 et p6);
*
pin1 = stru elem cont p2;
pin2 = stru elem cont p3;
pin = pin1 et pin2;
mid1 = stru elem cont p5;
mid2 = stru elem cont p6;
*
xp = stru poin droi p1 p4 tol;
xpln = pxordpoi xp p1;
*
p0 = 0 0;
pm = manu poi1 p0;
*
mesh = stru et xpln et pm;
*
tass mesh;
*
opti sauv form 'pinb73.msh';
sauv form mesh;
*
opti trac psc;
trac mesh;
trac qual pin;
list pin;
fin;
```

pinb73.epx

```
PINB73
*
ECHO
*CONV win
*
CAST MESH
*
LAGR CPLA
*
*opti dump
*
GEOM Q41L stru PMAT pm TERM
COMP EPAI  1. LECT stru TERM
           0.5 LECT pm TERM
     COUL VERT LECT stru TERM
          ROUG LECT pm TERM
MATE VM23 RO 8000. YOUN 2.E11 NU 0.0 ELAS 2.E11
          TRAC 1 2.E11 1.D0
          LECT stru TERM
     FANT 0.1 LECT pm TERM
*
opti pins reb2
LINK COUP SPLT NONE
     PINB BODY LECT pin1 TERM
          BODY LECT pin2 TERM
     BLOQ 12 LECT pm TERM
```

```
*
INIT VITE 1  500 LECT stru1 TERM
     VITE 1 -500 LECT stru2 TERM
*
ECRI DEPL VITE ACCE FINT FEXT CONT ECRO TFRE 10.E-3
     fich alic temp FREQ 1
                    poin lect p1 p2 p3 p4 term
                    elem lect pin mid1 mid2 term
*
OPTI PAS AUTO NOTE STEP IO
 csta 0.5E0
 log 1
 liaj alph 0.05D0
 LNKS STAT
*
CALCUL TINI 0. TEND 100.E-3
*============================================================
SUIT
Post-treatment
ECHO
*
RESU ALIC TEMP GARD PSCR
*
SORT GRAP
*
AXTE 1.0 'Time [s]'
*
COUR 1 'dx_p1'  DEPL COMP 1 NOEU LECT p1 TERM
COUR 2 'dx_p2'  DEPL COMP 1 NOEU LECT p2 TERM
COUR 3 'dx_p3'  DEPL COMP 1 NOEU LECT p3 TERM
COUR 4 'dx_p4'  DEPL COMP 1 NOEU LECT p4 TERM
COUR 5 'fex_p2'  FORC COMP 1 NOEU LECT p2 TERM
COUR 6 'fex_p3'  FORC COMP 1 NOEU LECT p3 TERM
COUR 7 'sx_mid1'  CONT COMP 1 ELEM LECT mid1 TERM
COUR 8 'sx_mid2'  CONT COMP 1 ELEM LECT mid2 TERM
COUR 11 'wtot' WTOT
COUR 12 'wcin' WCIN
COUR 13 'wint' WINT
COUR 14 'wext' WEXT
COUR 15 'bila' BILA
*
trac 1 2 3 4 axes 1.0 'DISPL. [M]' YZER
trac 5 6 axes 1.0 'FEXT [N]'
COLO ROUG VERT
trac 7 8 axes 1.0 'SX [PA]' YZER
TRAC 11 12 13 14 AXES 1.0 'Energy [J]' YZER
COLO NOIR ROUG VERT TURQ
TRAC 15 AXES 1.0 'Balance [%]' YZER
TRAC 11 12 13 14 AXES 1.0 'Energy [J]' YZER
                                  XMIN 0.0 XMAX 5.E-3 NX 10
COLO NOIR ROUG VERT TURQ
*
QUAL DEPL COMP 1 LECT p2 TERM REFE -9.46829E+0 TOLE 5.E-3
     DEPL COMP 1 LECT p3 TERM REFE  9.46829E+0 TOLE 5.E-3
     VITE COMP 1 LECT p2 TERM REFE -6.11061E+2 TOLE 5.E-3
     VITE COMP 1 LECT p3 TERM REFE  6.11061E+2 TOLE 5.E-3
*============================================================
FIN
```

pinb83.dgibi

```
opti donn 'pxordpoi.proc';
*
*opti titr 'PINB - 03';
opti dime 2 elem qua4;
*
p1=-201 -1;
p2=-1 -1;
p3=1 -1;
p4=201 -1;
p5=-101 -1;
p6=102 -1;
*
tol=0.001;
*
c1=p1 d 100 p2;
c2=p3 d 100 p4;
stru1=c1 TRAN 1 (0 2);
stru2=c2 TRAN 1 (0 2);
stru = stru1 et stru2;
elim tol (stru et p5 et p6);
*
pin1 = stru elem cont p2;
pin2 = stru elem cont p3;
pin = pin1 et pin2;
mid1 = stru elem cont p5;
mid2 = stru elem cont p6;
*
xp = stru poin droi p1 p4 tol;
xpln = pxordpoi xp p1;
*
p0 = 0 0;
pm = manu poi1 p0;
*
mesh = stru et xpln et pm;
*
tass mesh;
*
opti sauv form 'pinb83.msh';
sauv form mesh;
*
opti trac psc;
trac mesh;
trac qual pin;
list pin;
fin;
```

pinb83.epx

```
PINB83
*
ECHO
*CONV win
*
CAST MESH
*
LAGR CPLA
*
*opti dump
*
GEOM Q41L stru PMAT pm TERM
COMP EPAI  1. LECT stru TERM
           0.5 LECT pm TERM
     COUL VERT LECT stru TERM
           ROUG LECT pm TERM
MATE VM23 RO 8000. YOUN 2.E11 NU 0.0 ELAS 2.E11
           TRAC 1 2.E11 1.D0
           LECT stru TERM
     FANT 0.1 LECT pm TERM
*
opti pins reb2
LINK COUP SPLT NONE
     PINB BODY LECT pin1 TERM
           BODY LECT pin2 TERM
     BLOQ 12 LECT pm TERM
*
INIT VITE 1  500 LECT stru1 TERM
     VITE 1 -500 LECT stru2 TERM
*
ECRI DEPL VITE ACCE FINT FEXT CONT ECRO TFRE 10.E-3
     fich alic temp FREQ 1
                    poin lect p1 p2 p3 p4 term
                    elem lect pin mid1 mid2 term
*
OPTI PAS AUTO NOTE STEP IO
 csta 0.5E0
 log 1
 liaj alph 0.10D0
 LNKS STAT
*
CALCUL TINI 0. TEND 100.E-3
*============================================================
SUIT
Post-treatment
ECHO
*
RESU ALIC TEMP GARD PSCR
*
SORT GRAP
*
AXTE 1.0 'Time [s]'
*
COUR 1 'dx_p1'  DEPL COMP 1 NOEU LECT p1 TERM
COUR 2 'dx_p2'  DEPL COMP 1 NOEU LECT p2 TERM
COUR 3 'dx_p3'  DEPL COMP 1 NOEU LECT p3 TERM
COUR 4 'dx_p4'  DEPL COMP 1 NOEU LECT p4 TERM
COUR 5 'fex_p2'  FORC COMP 1 NOEU LECT p2 TERM
COUR 6 'fex_p3'  FORC COMP 1 NOEU LECT p3 TERM
COUR 7 'sx_mid1'  CONT COMP 1 ELEM LECT mid1 TERM
COUR 8 'sx_mid2'  CONT COMP 1 ELEM LECT mid2 TERM
COUR 11 'wtot' WTOT
COUR 12 'wcin' WCIN
COUR 13 'wint' WINT
COUR 14 'wext' WEXT
COUR 15 'bila' BILA
*
trac 1 2 3 4 axes 1.0 'DISPL. [M]' YZER
trac 5 6 axes 1.0 'FEXT [N]'
COLO ROUG VERT
trac 7 8 axes 1.0 'SX [PA]' YZER
TRAC 11 12 13 14 AXES 1.0 'Energy [J]' YZER
COLO NOIR ROUG VERT TURQ
TRAC 15 AXES 1.0 'Balance [%]' YZER
TRAC 11 12 13 14 AXES 1.0 'Energy [J]' YZER
                                  XMIN 0.0 XMAX 5.E-3 NX 10
COLO NOIR ROUG VERT TURQ
*
QUAL DEPL COMP 1 LECT p2 TERM REFE -9.49386E+0 TOLE 5.E-3
     DEPL COMP 1 LECT p3 TERM REFE  9.49386E+0 TOLE 5.E-3
     VITE COMP 1 LECT p2 TERM REFE -5.98300E+2 TOLE 5.E-3
     VITE COMP 1 LECT p3 TERM REFE  5.98300E+2 TOLE 5.E-3
*============================================================
FIN
```

pinb93.dgibi

```
opti donn 'pxordpoi.proc';
*
*opti titr 'PINB - 03';
opti dime 2 elem qua4;
*
p1=-201 -1;
```

```
p2=-1 -1;
p3=1 -1;
p4=201 -1;
p5=-101 -1;
p6=102 -1;
*
tol=0.001;
*
c1=p1 d 100 p2;
c2=p3 d 100 p4;
stru1=c1 TRAN 1 (0 2);
stru2=c2 TRAN 1 (0 2);
stru = stru1 et stru2;
elim tol (stru et p5 et p6);
*
pin1 = stru elem cont p2;
pin2 = stru elem cont p3;
pin = pin1 et pin2;
mid1 = stru elem cont p5;
mid2 = stru elem cont p6;
*
xp = stru poin droi p1 p4 tol;
xpln = pxordpoi xp p1;
*
p0 = 0 0;
pm = manu poi1 p0;
*
mesh = stru et xpln et pm;
*
tass mesh;
*
opti sauv form 'pinb93.msh';
sauv form mesh;
*
opti trac psc;
trac mesh;
trac qual pin;
list pin;
fin;
```

pinb93.epx

```
PINB93
*
ECHO
*CONV win
*
CAST MESH
*
LAGR CPLA
*
*opti dump
*
GEOM Q41L stru PMAT pm TERM
COMP EPAI  1. LECT stru TERM
          0.5 LECT pm TERM
     COUL VERT LECT stru TERM
          ROUG LECT pm TERM
MATE VM23 RO 8000. YOUN 2.E11 NU 0.0 ELAS 2.E11
          TRAC 1 2.E11 1.D0
          LECT stru TERM
     FANT 0.1 LECT pm TERM
*
opti pins reb2
LINK COUP SPLT NONE
     PINB BODY LECT pin1 TERM
          BODY LECT pin2 TERM
     BLOQ 12 LECT pm TERM
*
INIT VITE 1  500 LECT stru1 TERM
     VITE 1 -500 LECT stru2 TERM
*
ECRI DEPL VITE ACCE FINT FEXT CONT ECRO TFRE 10.E-3
     fich alic temp FREQ 1
                  poin lect p1 p2 p3 p4 term
                  elem lect pin mid1 mid2 term
*
OPTI PAS AUTO NOTE STEP IO
 csta 0.5E0
 log 1
 liaj alph 0.5D0
 LNKS STAT
*
CALCUL TINI 0. TEND 100.E-3
*============================================================
SUIT
Post-treatment
ECHO
*
RESU ALIC TEMP GARD PSCR
*
SORT GRAP
*
AXTE 1.0 'Time [s]'
*
COUR 1 'dx_p1'  DEPL COMP 1 NOEU LECT p1 TERM
COUR 2 'dx_p2'  DEPL COMP 1 NOEU LECT p2 TERM
COUR 3 'dx_p3'  DEPL COMP 1 NOEU LECT p3 TERM
COUR 4 'dx_p4'  DEPL COMP 1 NOEU LECT p4 TERM
COUR 5 'fex_p2'  FORC COMP 1 NOEU LECT p2 TERM
COUR 6 'fex_p3'  FORC COMP 1 NOEU LECT p3 TERM
COUR 7 'sx_mid1'  CONT COMP 1 ELEM LECT mid1 TERM
COUR 8 'sx_mid2'  CONT COMP 1 ELEM LECT mid2 TERM
COUR 11 'wtot' WTOT
COUR 12 'wcin' WCIN
COUR 13 'wint' WINT
COUR 14 'wext' WEXT
COUR 15 'bila' BILA
*
trac 1 2 3 4 axes 1.0 'DISPL. [M]' YZER
trac 5 6 axes 1.0 'FEXT [N]'
COLO ROUG VERT
trac 7 8 axes 1.0 'SX [PA]' YZER
TRAC 11 12 13 14 AXES 1.0 'Energy [J]' YZER
COLO NOIR ROUG VERT TURQ
TRAC 15 AXES 1.0 'Balance [%]' YZER
TRAC 11 12 13 14 AXES 1.0 'Energy [J]' YZER
                                  XMIN 0.0 XMAX 5.E-3 NX 10
COLO NOIR ROUG VERT TURQ
*
QUAL DEPL COMP 1 LECT p2 TERM REFE -9.47558E+0 TOLE 5.E-3
     DEPL COMP 1 LECT p3 TERM REFE  9.47558E+0 TOLE 5.E-3
     VITE COMP 1 LECT p2 TERM REFE -5.30215E+2 TOLE 5.E-3
     VITE COMP 1 LECT p3 TERM REFE  5.30215E+2 TOLE 5.E-3
*============================================================
FIN
```

pinl03.dgibi

```
opti donn 'pxordpoi.proc';
*
*opti titr 'PINL - 03';
opti dime 2 elem qua4;
*
p1=-201 -1;
p2=-1 -1;
p3=1 -1;
p4=201 -1;
p5=-101 -1;
p6=102 -1;
*
tol=0.001;
*
c1=p1 d 100 p2;
c2=p3 d 100 p4;
stru1=c1 TRAN 1 (0 2);
stru2=c2 TRAN 1 (0 2);
stru = stru1 et stru2;
elim tol (stru et p5 et p6);
*
pin1 = stru elem cont p2;
pin2 = stru elem cont p3;
pin = pin1 et pin2;
mid1 = stru elem cont p5;
mid2 = stru elem cont p6;
*
xp = stru poin droi p1 p4 tol;
xpln = pxordpoi xp p1;
*
mesh = stru et xpln;
*
tass mesh;
*
opti sauv form 'pinl03.msh';
sauv form mesh;
*
opti trac psc;
trac mesh;
trac qual pin;
list pin;
fin;
```

pinl03.epx

```
PINL03
*
ECHO
*CONV win
*
CAST MESH
*
LAGR CPLA LAGC
*
*opti dump
*
GEOM Q41L stru TERM
COMP EPAI 1. LECT stru TERM
     COUL VERT LECT stru TERM
MATE VM23 RO 8000. YOUN 2.E11 NU 0.0 ELAS 2.E11
          TRAC 1 2.E11 1.D0
          LECT stru TERM
*
opti pins reb2
LIAI PINB BODY LECT pin1 TERM
          BODY LECT pin2 TERM
*
INIT VITE 1  500 LECT stru1 TERM
     VITE 1 -500 LECT stru2 TERM
*
ECRI DEPL VITE ACCE FINT FEXT CONT ECRO TFRE 10.E-3
     fich tplo desc 'PINL03' freq 1
                  poin lect p1 p2 p3 p4 term
```

```
                  elem lect pin mid1 mid2 term
     fich xplo desc 'PINL03' tfre 10.e-3
                  poin lect xpln term
     fich alic temp FREQ 1
                  poin lect p1 p2 p3 p4 term
                  elem lect pin mid1 mid2 term
*
OPTI PAS AUTO NOTE STEP IO
 csta 0.5E0
 log 1
*
CALCUL TINI 0. TEND 100.E-3
*============================================================
SUIT
Post-treatment
ECHO
*
RESU ALIC TEMP GARD PSCR
*
SORT GRAP
*
AXTE 1.0 'Time [s]'
*
COUR 1 'dx_p1'  DEPL COMP 1 NOEU LECT p1 TERM
COUR 2 'dx_p2'  DEPL COMP 1 NOEU LECT p2 TERM
COUR 3 'dx_p3'  DEPL COMP 1 NOEU LECT p3 TERM
COUR 4 'dx_p4'  DEPL COMP 1 NOEU LECT p4 TERM
COUR 5 'fex_p2'  FORC COMP 1 NOEU LECT p2 TERM
COUR 6 'fex_p3'  FORC COMP 1 NOEU LECT p3 TERM
COUR 7 'sx_mid1'  CONT COMP 1 ELEM LECT mid1 TERM
COUR 8 'sx_mid2'  CONT COMP 1 ELEM LECT mid2 TERM
COUR 11 'wtot' WTOT
COUR 12 'wcin' WCIN
COUR 13 'wint' WINT
COUR 14 'wext' WEXT
COUR 15 'bila' BILA
*
trac 1 2 3 4 axes 1.0 'DISPL. [M]' YZER
trac 5 6 axes 1.0 'FEXT [N]'
COLO ROUG VERT
trac 7 8 axes 1.0 'SX [PA]' YZER
TRAC 11 12 13 14 AXES 1.0 'Energy [J]' YZER
COLO NOIR ROUG VERT TURQ
TRAC 15 AXES 1.0 'Balance [%]' YZER
TRAC 11 12 13 14 AXES 1.0 'Energy [J]' YZER
                                    XMIN 0.0 XMAX 5.E-3 NX 10
COLO NOIR ROUG VERT TURQ
*
QUAL DEPL COMP 1 LECT p2 TERM REFE -9.51025E+0 TOLE 5.E-3
     DEPL COMP 1 LECT p3 TERM REFE  9.51025E+0 TOLE 5.E-3
     VITE COMP 1 LECT p2 TERM REFE -5.13299E+2 TOLE 5.E-3
     VITE COMP 1 LECT p3 TERM REFE  5.13299E+2 TOLE 5.E-3
*============================================================
FIN
```

pinl13.dgibi

```
opti donn 'pxordpoi.proc';
*
*opti titr 'PINB - 03';
opti dime 2 elem qua4;
*
p1=-201 -1;
p2=-1 -1;
p3=1 -1;
p4=201 -1;
p5=-101 -1;
p6=102 -1;
*
tol=0.001;
*
c1=p1 d 100 p2;
c2=p3 d 100 p4;
stru1=c1 TRAN 1 (0 2);
stru2=c2 TRAN 1 (0 2);
stru = stru1 et stru2;
elim tol (stru et p5 et p6);
*
pin1 = stru elem cont p2;
pin2 = stru elem cont p3;
pin = pin1 et pin2;
mid1 = stru elem cont p5;
mid2 = stru elem cont p6;
*
xp = stru poin droi p1 p4 tol;
xpln = pxordpoi xp p1;
*
mesh = stru et xpln;
*
tass mesh;
*
opti sauv form 'pinl13.msh';
sauv form mesh;
*
opti trac psc;
trac mesh;
trac qual pin;
list pin;
fin;
```

pinl13.epx

```
PINL13
*
ECHO
*CONV win
*
CAST MESH
*
LAGR CPLA
*
*opti dump
*
GEOM Q41L stru TERM
COMP EPAI 1. LECT stru TERM
     COUL VERT LECT stru TERM
MATE VM23 RO 8000. YOUN 2.E11 NU 0.0 ELAS 2.E11
          TRAC 1 2.E11 1.D0
          LECT stru TERM
*
opti pins reb2
LINK COUP
     PINB BODY LECT pin1 TERM
          BODY LECT pin2 TERM
*
INIT VITE 1  500 LECT stru1 TERM
     VITE 1 -500 LECT stru2 TERM
*
ECRI DEPL VITE ACCE FINT FEXT CONT ECRO TFRE 10.E-3
     fich alic temp FREQ 1
                    poin lect p1 p2 p3 p4 term
                    elem lect pin mid1 mid2 term
*
OPTI PAS AUTO NOTE STEP IO
 csta 0.5E0
 log 1
 LNKS STAT
*
CALCUL TINI 0. TEND 100.E-3
*============================================================
SUIT
Post-treatment
ECHO
*
RESU ALIC TEMP GARD PSCR
*
SORT GRAP
*
AXTE 1.0 'Time [s]'
*
COUR 1 'dx_p1'  DEPL COMP 1 NOEU LECT p1 TERM
COUR 2 'dx_p2'  DEPL COMP 1 NOEU LECT p2 TERM
COUR 3 'dx_p3'  DEPL COMP 1 NOEU LECT p3 TERM
COUR 4 'dx_p4'  DEPL COMP 1 NOEU LECT p4 TERM
COUR 5 'fex_p2'  FORC COMP 1 NOEU LECT p2 TERM
COUR 6 'fex_p3'  FORC COMP 1 NOEU LECT p3 TERM
COUR 7 'sx_mid1'  CONT COMP 1 ELEM LECT mid1 TERM
COUR 8 'sx_mid2'  CONT COMP 1 ELEM LECT mid2 TERM
COUR 11 'wtot' WTOT
COUR 12 'wcin' WCIN
COUR 13 'wint' WINT
COUR 14 'wext' WEXT
COUR 15 'bila' BILA
*
trac 1 2 3 4 axes 1.0 'DISPL. [M]' YZER
trac 5 6 axes 1.0 'FEXT [N]'
COLO ROUG VERT
trac 7 8 axes 1.0 'SX [PA]' YZER
TRAC 11 12 13 14 AXES 1.0 'Energy [J]' YZER
COLO NOIR ROUG VERT TURQ
TRAC 15 AXES 1.0 'Balance [%]' YZER
TRAC 11 12 13 14 AXES 1.0 'Energy [J]' YZER
                                    XMIN 0.0 XMAX 5.E-3 NX 10
COLO NOIR ROUG VERT TURQ
*
QUAL DEPL COMP 1 LECT p2 TERM REFE -9.51025E+0 TOLE 5.E-3
     DEPL COMP 1 LECT p3 TERM REFE  9.51025E+0 TOLE 5.E-3
     VITE COMP 1 LECT p2 TERM REFE -5.13299E+2 TOLE 5.E-3
     VITE COMP 1 LECT p3 TERM REFE  5.13299E+2 TOLE 5.E-3
*============================================================
FIN
```

pinl23.dgibi

```
opti donn 'pxordpoi.proc';
*
*opti titr 'PINL - 03';
opti dime 2 elem qua4;
*
p1=-201 -1;
p2=-1 -1;
p3=1 -1;
p4=201 -1;
p5=-101 -1;
p6=102 -1;
*
tol=0.001;
*
c1=p1 d 100 p2;
c2=p3 d 100 p4;
stru1=c1 TRAN 1 (0 2);
stru2=c2 TRAN 1 (0 2);
```

```
stru = stru1 et stru2;
elim tol (stru et p5 et p6);
*
pin1 = stru elem cont p2;
pin2 = stru elem cont p3;
pin = pin1 et pin2;
mid1 = stru elem cont p5;
mid2 = stru elem cont p6;
*
xp = stru poin droi p1 p4 tol;
xpln = pxordpoi xp p1;
*
p0 = 0 0;
pm = manu poi1 p0;
*
mesh = stru et xpln et pm;
*
tass mesh;
*
opti sauv form 'pinl23.msh';
sauv form mesh;
*
opti trac psc;
trac mesh;
trac qual pin;
list pin;
fin;
```

pinl23.epx

```
PINL23
*
ECHO
*CONV win
*
CAST MESH
*
LAGR CPLA LAGC
*
*opti dump
*
GEOM Q41L stru PMAT pm TERM
COMP EPAI  1. LECT stru TERM
          0.5 LECT pm TERM
     COUL VERT LECT stru TERM
          ROUG LECT pm TERM
MATE VM23 RO 8000. YOUN 2.E11 NU 0.0 ELAS 2.E11
          TRAC 1 2.E11 1.D0
          LECT stru TERM
     FANT 0.1 LECT pm TERM
*
opti pins reb2
LIAI PINB BODY LECT pin1 TERM
          BODY LECT pin2 TERM
     BLOQ 12 LECT pm TERM
*
INIT VITE 1  500 LECT stru1 TERM
     VITE 1 -500 LECT stru2 TERM
*
ECRI DEPL VITE ACCE FINT FEXT CONT ECRO TFRE 10.E-3
     fich tplo desc 'PINL23' freq 1
                    poin lect p1 p2 p3 p4 term
                    elem lect pin mid1 mid2 term
     fich xplo desc 'PINL23' tfre 10.e-3
                    poin lect xpln term
     fich alic temp FREQ 1
                    poin lect p1 p2 p3 p4 term
                    elem lect pin mid1 mid2 term
*
OPTI PAS AUTO NOTE STEP IO
 csta 0.5E0
 log 1
*
CALCUL TINI 0. TEND 100.E-3
*================================================================
SUIT
Post-treatment
ECHO
*
RESU ALIC TEMP GARD PSCR
*
SORT GRAP
*
AXTE 1.0 'Time [s]'
*
COUR 1 'dx_p1'  DEPL COMP 1 NOEU LECT p1 TERM
COUR 2 'dx_p2'  DEPL COMP 1 NOEU LECT p2 TERM
COUR 3 'dx_p3'  DEPL COMP 1 NOEU LECT p3 TERM
COUR 4 'dx_p4'  DEPL COMP 1 NOEU LECT p4 TERM
COUR 5 'fex_p2'  FORC COMP 1 NOEU LECT p2 TERM
COUR 6 'fex_p3'  FORC COMP 1 NOEU LECT p3 TERM
COUR 7 'sx_mid1'  CONT COMP 1 ELEM LECT mid1 TERM
COUR 8 'sx_mid2'  CONT COMP 1 ELEM LECT mid2 TERM
COUR 11 'wtot' WTOT
COUR 12 'wcin' WCIN
COUR 13 'wint' WINT
COUR 14 'wext' WEXT
COUR 15 'bila' BILA
*
trac 1 2 3 4 axes 1.0 'DISPL. [M]' YZER
trac 5 6 axes 1.0 'FEXT [N]'
COLO ROUG VERT
trac 7 8 axes 1.0 'SX [PA]' YZER
TRAC 11 12 13 14 AXES 1.0 'Energy [J]' YZER
COLO NOIR ROUG VERT TURQ
TRAC 15 AXES 1.0 'Balance [%]' YZER
TRAC 11 12 13 14 AXES 1.0 'Energy [J]' YZER
                                    XMIN 0.0 XMAX 5.E-3 NX 10
COLO NOIR ROUG VERT TURQ
*
QUAL DEPL COMP 1 LECT p2 TERM REFE -9.51025E+0 TOLE 5.E-3
     DEPL COMP 1 LECT p3 TERM REFE  9.51025E+0 TOLE 5.E-3
     VITE COMP 1 LECT p2 TERM REFE -5.13299E+2 TOLE 5.E-3
     VITE COMP 1 LECT p3 TERM REFE  5.13299E+2 TOLE 5.E-3
*================================================================
FIN
```

pinl33.dgibi

```
opti donn 'pxordpoi.proc';
*
*opti titr 'PINB - 03';
opti dime 2 elem qua4;
*
p1=-201 -1;
p2=-1 -1;
p3=1 -1;
p4=201 -1;
p5=-101 -1;
p6=102 -1;
*
tol=0.001;
*
c1=p1 d 100 p2;
c2=p3 d 100 p4;
stru1=c1 TRAN 1 (0 2);
stru2=c2 TRAN 1 (0 2);
stru = stru1 et stru2;
elim tol (stru et p5 et p6);
*
pin1 = stru elem cont p2;
pin2 = stru elem cont p3;
pin = pin1 et pin2;
mid1 = stru elem cont p5;
mid2 = stru elem cont p6;
*
xp = stru poin droi p1 p4 tol;
xpln = pxordpoi xp p1;
*
p0 = 0 0;
pm = manu poi1 p0;
*
mesh = stru et xpln et pm;
*
tass mesh;
*
opti sauv form 'pinl33.msh';
sauv form mesh;
*
opti trac psc;
trac mesh;
trac qual pin;
list pin;
fin;
```

pinl33.epx

```
PINL33
*
ECHO
*CONV win
*
CAST MESH
*
LAGR CPLA
*
*opti dump
*
GEOM Q41L stru PMAT pm TERM
COMP EPAI  1. LECT stru TERM
          0.5 LECT pm TERM
     COUL VERT LECT stru TERM
          ROUG LECT pm TERM
MATE VM23 RO 8000. YOUN 2.E11 NU 0.0 ELAS 2.E11
          TRAC 1 2.E11 1.D0
          LECT stru TERM
     FANT 0.1 LECT pm TERM
*
opti pins reb2
LINK COUP SPLT NONE
     PINB BODY LECT pin1 TERM
          BODY LECT pin2 TERM
     BLOQ 12 LECT pm TERM
*
INIT VITE 1  500 LECT stru1 TERM
     VITE 1 -500 LECT stru2 TERM
*
ECRI DEPL VITE ACCE FINT FEXT CONT ECRO TFRE 10.E-3
     fich alic temp FREQ 1
                    poin lect p1 p2 p3 p4 term
                    elem lect pin mid1 mid2 term
*
```

```
OPTI PAS AUTO NOTE STEP IO
 csta 0.5E0
 log 1
 LNKS STAT
*
CALCUL TINI 0. TEND 100.E-3
*============================================================
SUIT
Post-treatment
ECHO
*
RESU ALIC TEMP GARD PSCR
*
SORT GRAP
*
AXTE 1.0 'Time [s]'
*
COUR 1 'dx_p1'  DEPL COMP 1 NOEU LECT p1 TERM
COUR 2 'dx_p2'  DEPL COMP 1 NOEU LECT p2 TERM
COUR 3 'dx_p3'  DEPL COMP 1 NOEU LECT p3 TERM
COUR 4 'dx_p4'  DEPL COMP 1 NOEU LECT p4 TERM
COUR 5 'fex_p2'  FORC COMP 1 NOEU LECT p2 TERM
COUR 6 'fex_p3'  FORC COMP 1 NOEU LECT p3 TERM
COUR 7 'sx_mid1'  CONT COMP 1 ELEM LECT mid1 TERM
COUR 8 'sx_mid2'  CONT COMP 1 ELEM LECT mid2 TERM
COUR 11 'wtot' WTOT
COUR 12 'wcin' WCIN
COUR 13 'wint' WINT
COUR 14 'wext' WEXT
COUR 15 'bila' BILA
*
trac 1 2 3 4 axes 1.0 'DISPL. [M]' YZER
trac 5 6 axes 1.0 'FEXT [N]'
COLO ROUG VERT
trac 7 8 axes 1.0 'SX [PA]' YZER
TRAC 11 12 13 14 AXES 1.0 'Energy [J]' YZER
COLO NOIR ROUG VERT TURQ
TRAC 15 AXES 1.0 'Balance [%]' YZER
TRAC 11 12 13 14 AXES 1.0 'Energy [J]' YZER
                                    XMIN 0.0 XMAX 5.E-3 NX 10
COLO NOIR ROUG VERT TURQ
*
QUAL DEPL COMP 1 LECT p2 TERM REFE -9.51025E+0 TOLE 5.E-3
     DEPL COMP 1 LECT p3 TERM REFE  9.51025E+0 TOLE 5.E-3
     VITE COMP 1 LECT p2 TERM REFE -5.13299E+2 TOLE 5.E-3
     VITE COMP 1 LECT p3 TERM REFE  5.13299E+2 TOLE 5.E-3
*============================================================
FIN
```

pxordpoi.proc

```
*$$$$ PXORDPOI
*
* pour ordonner une serie de points PLIN en partant de P1
*
* Input:
* =====
* PLIN = objet MAILLAGE de type POI1 (ligne de points)
* P1   = premier point de la ligne (typ POINT)
*
* Output:
* ======
* PORDO = objet MAILLAGE de type POI1 (ligne de points) contenant
*         les points ordonnes a partir de P1
*
'DEBPROC' PXORDPOI PLIN*'MAILLAGE' P1*'POINT' ;
*-------------------------------------------------------------
*
PORDO=P1;
PPA=P1;
NE='NBEL' PLIN;
*
I=0;
'REPETER' LAB1 (NE-1);
  I=I + 1;
* mess I;
  PLIN= 'DIFF' ((PPA 'ET' PPA) 'ELEM' 1) PLIN;
  PPA=PLIN 'POIN' 'PROC' PPA;
  PORDO=PORDO 'ET' PPA;
'FIN' LAB1;
*
'FINPROC' PORDO;
```

List of input files

C

P

GETTING IN TOUCH WITH THE EU

In person

All over the European Union there are hundreds of Europe Direct centres. You can find the address of the centre nearest you online (european-union.europa.eu/contact-eu/meet-us_en).

On the phone or in writing

Europe Direct is a service that answers your questions about the European Union. You can contact this service:

— by freephone: 00 800 6 7 8 9 10 11 (certain operators may charge for these calls),

— at the following standard number: +32 22999696,

— via the following form: european-union.europa.eu/contact-eu/write-us_en.

FINDING INFORMATION ABOUT THE EU

Online

Information about the European Union in all the official languages of the EU is available on the Europa website (european-union.europa.eu).

EU publications

You can view or order EU publications at op.europa.eu/en/publications. Multiple copies of free publications can be obtained by contacting Europe Direct or your local documentation centre (european-union.europa.eu/contact-eu/meet-us_en).

EU law and related documents

For access to legal information from the EU, including all EU law since 1951 in all the official language versions, go to EUR-Lex (eur-lex.europa.eu).

Open data from the EU

The portal data.europa.eu provides access to open datasets from the EU institutions, bodies and agencies. These can be downloaded and reused for free, for both commercial and non-commercial purposes. The portal also provides access to a wealth of datasets from European countries.

Science for policy

The Joint Research Centre (JRC) provides independent, evidence-based knowledge and science, supporting EU policies to positively impact society

EU Science Hub
joint-research-centre.ec.europa.eu

@EU_ScienceHub
EU Science Hub - Joint Research Centre
EU Science, Research and Innovation
EU Science Hub
@eu_science

Publications Office
of the European Union