Implementation of Assembled Surface Normals and of a Penalty Contact Formulation in the Pinball Model of EUROPLEXUS

Revision 1

Casadei, F.
Larcher, M.
Valsamos, G.
Faucher, V.

2018
This publication is a Technical report by the Joint Research Centre (JRC), the European Commission’s science and knowledge service. It aims to provide evidence-based scientific support to the European policymaking process. The scientific output expressed does not imply a policy position of the European Commission. Neither the European Commission nor any person acting on behalf of the Commission is responsible for the use that might be made of this publication.

Contact information
Name: Martin Larcher
Address: JRC Ispra
Email: martin.larcher@ec.europa.eu
Tel.: +39-0332-789563

JRC Science Hub
https://ec.europa.eu/jrc

JRC111429
EUR 29217 EN


Luxembourg: Publications Office of the European Union, 2018
© European Union, 2018

Reuse is authorised provided the source is acknowledged. The reuse policy of European Commission documents is regulated by Decision 2011/833/EU (OJ L 330, 14.12.2011, p. 39).

For any use or reproduction of photos or other material that is not under the EU copyright, permission must be sought directly from the copyright holders.

How to cite this report: Author(s), Title, EUR (where available), Publisher, Publisher City, Year of Publication, ISBN (where available), doi (where available), PUBSY No.

All images © European Union 2018
Implementation of Assembled Surface Normals and of a Penalty Contact Formulation in the Pinball Model of EUROPLEXUS - Revision 1

F. Casadei, M.Larcher, G. Valsamos
European Commission,
Institute for the Protection and Security of the Citizen
Joint Research Centre, 21027 Ispra, Italy

V. Faucher
CEA, DEN, DANS, DM2S, SEMT, DYN, F-91191 Gif-sur-Yvette, France
Contents

1 Introduction .................................................. 2

2 Contact algorithm for continuum elements ....................... 3
   2.1 Assembled Surface Normals (ASN) .......................... 3
   2.2 Calculation of the penetration ............................ 4
      2.2.1 Penetration direction (contact normal) ............... 5
      2.2.2 Penetration ........................................... 5
   2.3 Calculation of the rate of penetration ..................... 6
   2.4 Interaction between a couple of pinballs .................. 7
   2.5 Calculation of the penalty force .......................... 8
      2.5.1 Classical expressions of penalty coefficients in 3D .... 9
      2.5.2 Penalty coefficients in 2D ............................ 10
      2.5.3 Classical expressions of penalty coefficients for shells . 10
      2.5.4 Expressions of penalty coefficients in EUROPLEXUS for sliding surfaces . 11
   2.6 Bulk modulus .............................................. 12
   2.7 Distribution of penalty force on the element nodes. ......... 13
   2.8 Considerations on penalty force between contacting elements or pinballs. .... 13
   2.9 Treatment of rebound with the penalty method ............... 15
   2.10 Generalized ASN algorithm for hierarchic pinballs ......... 16
      2.10.1 Single element. ....................................... 16
      2.10.2 Simple mesh .......................................... 17
      2.10.3 Flat aligned contact ................................... 17
      2.10.4 Flat mis-aligned contact ................................ 17
      2.10.5 Vertex-to-face contact .................................. 17
   2.11 Contact rules ............................................. 18
      2.11.1 ASNs and pinball types ................................ 18
      2.11.2 Implementation of the ASN algorithm .................. 20
      2.11.3 Penetration direction (contact normal) ............... 21
   2.12 Implementation of the contact force calculation .......... 22
      2.12.1 3D continuum .......................................... 23
      2.12.2 2D continuum .......................................... 24
   2.13 New input directives ...................................... 25

3 Contact algorithm for shell/beam/bar elements ................... 45
   3.1 2D case .................................................. 45
   3.2 3D case .................................................. 46
      3.2.1 Beam/bar elements in 3D ............................... 46
      3.2.2 Shell elements in 3D ................................... 47
   3.3 Calculation of the contact normal .......................... 48
4 Implementation details .......................................................... 55
  4.1 Implemented version of the contact normal (penetration direction) 58
    4.1.1 Contact normal between two parent pinballs ...................... 58
    4.1.2 Contact normal between two pinballs of which at least one is a descendent . . 59
  4.2 Calculation of the ASN for a descendent pinball ..................... 61
    4.2.1 The IFACE array .................................................... 61
    4.2.2 Material point .................................................... 65
    4.2.3 Continuum element ................................................. 65
    4.2.4 Shell element .................................................... 66
    4.2.5 Beam/bar element ................................................. 67
    4.2.6 Summary of cases in which the ASN of a pinball is zero .......... 68
  5 Numerical examples .......................................................... 75
    5.1 Visualization of the ASNs ............................................. 75
    5.2 Visualization of the ASNs and of the (sub-)pinball types .............. 84
    5.3 Simple impact tests ................................................... 95
      5.3.1 Impact between continuum elements ............................. 95
      5.3.2 Impacts between material points ............................... 105
      5.3.3 Impacts between material points and other element types ........... 108
      5.3.4 Impacts of “sliding” type between continuum bodies .............. 116
    5.4 Tube array crash test .................................................. 135
  6 References ........................................................................... 143
  7 Appendix A - Closest points on two segments ............................ 145
  8 Appendix B - Unpublished Material on Pinballs .......................... 149
  9 Appendix C - Input Files ..................................................... 173
List of Figures

1 - Continuum elements mesh. .............................................................. 27
2 - Element face normals. ................................................................. 27
3 - Assembled element nodal normals. ................................................. 28
4 - Assembled global nodal normals. .................................................. 28
5 - Assembled global element normals. ................................................. 29
6 - Normalized element normals. ....................................................... 29
7 - Normalized global nodal normals. ............................................... 30
8 - Assembled global element normals (variant). .................................. 30
9 - Normalized global element normals (variant). .................................. 30
10 - Pathological case (insufficient discretization). ............................... 31
11 - Penetration between two pinballs. ............................................... 31
12 - Rate of penetration between two pinballs. .................................... 32
13 - Variation of the penetration between two pinballs. ......................... 32
14 - Contacting hexahedra in 3D. ..................................................... 33
15 - Contact in 3D computed by zero-level pinballs. ............................. 33
16 - Contact in 3D computed by hierarchic pinballs. ............................ 34
17 - ASN with hierarchic pinballs on an isolated 2D quadrilateral. ............ 34
18 - ASN with hierarchic pinballs on a simple mesh of 2D quadrilaterals. .... 35
19 - “Flat” aligned contact with hierarchic pinballs. ............................... 35
20 - “Flat” misaligned contact with hierarchic pinballs. ......................... 36
21 - Vertex-to-face contact with hierarchic pinballs. ............................. 36
22 - Element external face normals (built only along external faces). .......... 37
23 - Assembled global external nodal normals. .................................... 37
24 - Assembled global element normals. ............................................. 38
25 - Normalized element normals. ..................................................... 38
26 - Normalized global external nodal normals. ................................... 39
27 - Example of stand-alone triangle. ............................................... 39
28 - Pinball types. ............................................................................. 40
29 - Penetration direction (contact normal) between two “element” pinballs. 41
30 - Penetration direction (contact normal) between two pinballs of which only one has a defined normal. ................................................................. 41
31 - Penetration direction (contact normal) between two face pinballs. ....... 42
32 - Penetration direction (contact normal) between a face pinball and a vertex pinball. ................................................................. 43
103 - Particle velocity in cases PMAT05, PMAT06 and PMAT07.
104 - Particle contact force in cases PMAT05, PMAT06 and PMAT07.
105 - Sliding contact test SLID01.
106 - Displacement of the upper block center in case SLID01.
107 - Velocity components of the upper block center in case SLID01.
108 - Contact force components of the upper block center in case SLID01.
109 - Displacement of the upper block center in case SLID02.
110 - Velocity components of the upper block center in case SLID02.
111 - Contact force components of the upper block center in case SLID02.
112 - Horizontal velocity components of the upper block center in cases SLID01 and SLID02.
113 - Displacement of the upper block center in case SLID03.
114 - Velocity components of the upper block center in case SLID03.
115 - Contact force components of the upper block center in case SLID03.
116 - Sliding contact test SLID04.
117 - Displacement of the upper block center in case SLID04.
118 - Velocity components of the upper block center in case SLID04.
119 - Contact force components of the upper block center in case SLID04.
120 - Sliding contact test SLID05.
121 - Sliding contact test SLID06.
122 - Contact configurations in test SLID06 (steps 0 to 7).
123 - Sliding contact test SLID07.
124 - Sliding contact test SLID08.
125 - Sliding contact test SLID09.
126 - Sliding contact test SLID09, details on contact conditions.
127 - Sliding contact test SLID10.
128 - Sketch of the structure crash problem (simplified version).
129 - Nodal ASNs and parent pinball ASNs in case CARN07 (initial configuration).
130 - Nodal ASNs in case CARN07 at 0.615 s.
131 - Contact normals in case CARN07 at 0.76 s.
132 - Contact forces in case CARN07 at 0.76 s.
133 - Final contacts in case CARN07.
134 - Final plastic strains in case CARN07.
135 - Upper and lower crushing forces in case CARN07.
136 - Crushing forces in cases CARA07 and CARN07.
137 - Final contacts in case CARO07.
138 - Final plastic strains in case CARO07. ................................. 142
139 - Crushing forces in cases CARA07, CARN07 and CARO07. ........... 142
140 - Closest points on two lines. ................................................. 147
List of Tables

1 - Summary of precedences. ................................................................. 22
2 - Color code for the different pinball types. ........................................ 26
3 - Precedence rules in the contact between two pinballs. ....................... 60
4 - Tests to show the nodal, parent pinball and descendent pinball ASNs. ...... 75
5 - Tests to show the ASNs and the (sub-)pinball types. ............................ 84
6 - Simple impact tests between continuum elements. .............................. 95
7 - Impact tests between material points. ............................................. 105
8 - Impact tests between material points and other element types. ............... 108
9 - Sliding-type contact between continuum elements. ............................. 116
10 - Tube array crash tests. ................................................................. 136
Foreword

This is a completely revised version of the report “Implementation of Assembled Surface Normals and of a Penalty Contact Formulation in the Pinball Model of EUROPLEXUS”, EUR 26714 EN, JRC90939, 2014 [18]. The major changes are:

• A different choice of the definition of the contact normal. The old definition was taken from the original work of Belytschko et al. The new definition is opposite to the previous one, to be in accordance with the pre-existing implementation of the pinball contact algorithm without ASN and based on Lagrange multipliers (strong formulation) in EUROPLEXUS and documented in reference [13].

• The fact that for shell/beam/bar elements without a topological thickness the ASN is defined only apart from the sign is now taken into account in the algorithm to compute the contact normal.

• The case of contact between two corner pinballs in 3D (C-C contact) is now treated by the more accurate expression using the line joining the closest points on the two segments, instead of the generic expression.

• The calculation of the ASN for a descendent pinball of shell type in 3D has been revised, as concerns the case of a descendent near a node of shell element. This yields more consistent normals in this case.
1. Introduction

The most popular contact algorithms available in finite element computer codes are probably the so-called slide line (in 2D) and slide surface (in 3D) algorithms proposed by Hallquist and Benson [1-2]. They are based on the notion of penetration of slave nodes into master segments (in 2D) or into master surfaces (in 3D). These algorithms suffer from a number of geometrically pathological cases in which physical penetration is not detected.

The pinball method proposed by Belytschko and co-workers from the late 80’s [3-10] for application in impact problems with penetration is much more robust as concerns penetration detection. The pinball contact-impact method has been implemented in EUROPLEXUS in [12-15], initially based upon a strong, Lagrange-multiplier based solution strategy of the contact constraints (see [11] for details of the method).

EUROPLEXUS [16] is a computer code for fast explicit transient dynamic analysis of fluid-structure systems jointly developed by the French Commissariat à l’Energie Atomique et aux Energies Alternatives (CEA Saclay) and by the Joint Research Centre of the European Commission (JRC Ispra).

Recently, the so-called Assembled Surface Normal (ASN) algorithm of Belytschko [3] and an alternative penalty-based solution of the contact constraints have also been introduced as an option in the code. They are described in the present report, which is organized as follows:

- Section 2 presents the contact-impact model based upon pinballs with Assembled Surface Normals (ASN) and using a penalty approach.
- Section 4 presents some implementation details.
- Section 5 presents some numerical examples for the validation of the newly implemented models.
- Appendix A contains the algorithm used to compute the closest points on two segments.
- Appendix B contains an unpublished (and slightly incomplete) paper which gives many details on the hierarchic pinball contact-impact model with Lagrange Multipliers.
- Finally, Appendix C contains a listing of all the input files mentioned in the present report.
2. Contact algorithm for continuum elements

The version of the pinball method based upon a penalty approach to compute contact forces is presented in this Section. We assume a mesh discretization consisting of (bulky) continuum elements, see Figure 1 for a 2D example, and an erosion criterion to eliminate failed elements from the calculation. Due to erosion, the active surface of the contacting bodies tends to become very irregular during penetration. Therefore, a method is needed to compute a relatively smoothly-varying normal to the surface. This normal should not follow the local variations due to irregular mesh boundaries, but represents an “average” normal to the surface. An algorithm called ASN (Assembled Surface Normal) to compute such a normal was proposed by Belytschko and Law in 1985 [3].

2.1 Assembled Surface Normals (ASN)

For a bulky domain composed of continuum elements, the algorithm consists of the following steps (see the sample mesh of Figure 1):

- Build element-by-element face normals, see Figure 2. These normals are oriented “outgoing” from the element and are not normalized, i.e. the norm of the vector equals the length (in 2D) or area (in 3D) of the face.

- Assemble element-by-element face normals at each element’s nodes, to obtain element-by-element nodal normals, see Figure 3. These are not normalized.

- Assemble element-by-element nodal normals at global (common) nodes, to obtain (global) nodal normals, see Figure 4. These are not normalized. Note that the resulting normals are zero (within round-off errors) at internal nodes, while they are non-zero at nodes on the surface of the mesh. Vanishing normals are set exactly to zero.

- Assemble (global) nodal normals to obtain (global) element normals, see Figure 5. These are not normalized.

- Normalize the resulting vectors to unit length, to obtain normalized (global) element normals, see Figure 6.

As an alternative one could, after the first three steps of the previous procedure:

- Normalize the global nodal normals, see Figure 7.

- Assemble (the normalized global) nodal normals to obtain (variant) element (global) normals, see Figure 8. These are not normalized.

- Normalize the resulting vectors to unit length, to obtain normalized (variant) global element normals, see Figure 9.
By comparing Figures 6 and 9 one sees that the final result depends upon the chosen procedure (basic or variant), but only slightly.

2.2 Calculation of the penetration

To apply the penalty method in a contact problem, it is necessary to compute the penetration between the two contacting bodies. In the pinball method, the penetration is computed for each couple of overlapping pinballs. A parent (i.e. zero-level) pinball is embedded in each element having a non-zero normal as a result of the ASN procedure described at the previous paragraph. Normally such elements are all and only the elements having at least one face on the surface of the body (surface elements), at least if a sufficient spatial discretization of the problem is assumed.

A pathological case is illustrated in Figure 10, where some elements “on the surface” have a zero normal. This happens because there is only one layer of continuum elements across the body. This situation is not acceptable in the class of problems of interest here (impact problems with penetration discretized by continuum elements) and is therefore ignored in the following treatment.

Consider a generic couple of pinballs \( A, B \) which overlap partially, thus indicating the presence of contact, see Figure 11a. Let \( R_A, R_B \) be the radii and \( C_A, C_B \) the centres of the two pinballs. The vector joining the two centres, oriented from \( A \) to \( B \) by convention, is then†:

\[
C_{AB} = C_A - C_B
\]  

and, since we have assumed that the two pinballs are overlapping, it is:

\[
\|C_{AB}\| < (R_A + R_B).
\]  

We indicate by \( n_A, n_B \) “the” normals to the two pinballs, i.e. the approximate normals to the two bodies in the elements to which the pinballs are attached, computed by the ASN procedure described above. Note that in both versions of the ASN procedure the final “element” normals are normalized and should therefore be of the same (unit) length. However, in Figure 11a, taken from reference [5], the two normals have different lengths (not normalized). This choice is more general, although in contrast with the ASN procedure as listed above and in the literature.

In order to study the interaction between the two pinballs it is useful to consider a fictitious “enlarged” pinball of center \( C_B \) and of radius equal to the sum of the radii:

\[
R_{AB} = R_A + R_B.
\]  

† Note that in the work of Belytschko et al. the opposite definition is taken \( C_{AB} = C_B - C_A \). The sign is changed here in order to be consistent with the pre-existing implementation of pinballs in EPX, without use of ASN and based upon Lagrange multipliers, described in reference [13].
This sphere, shown in Figure 11b, contains the $B$ pinball entirely and its surface is the locus of the centres $C_A$ of all pinballs $A$ of radius $R_A$ in contact with $B$ with zero overlap.

### 2.2.1 Penetration direction (contact normal)

First, the penetration direction, i.e. the direction along which the penetration is assumed to occur, is defined as the “average” normal direction of the two pinballs (see Figure 11b), given by:

$$\hat{n}_{AB} = \frac{\hat{n}_A - \hat{n}_B}{\|\hat{n}_A - \hat{n}_B\|}. \quad (4)$$

Note that this vector is normalized to unit length, as indicated by a superposed “hat”, and is loosely directed “from” $A$ “to” $B$, if $\hat{n}_A$ and $\hat{n}_B$ are outward normals to the two contacting bodies (assumed bulky) in which pinballs $A$ and $B$ are embedded. The penetration direction $\hat{n}_{AB}$ is also referred to as the contact normal (for contact between pinballs between $A$ and $B$) in the following. Note that a generic pinball $A$ can be in contact with more than one pinball ($B$, $C$, etc.) and for each contact a different contact normal $\hat{n}_{AB}$, $\hat{n}_{AC}$, etc. can be defined.

Then, the penetration $p$ is defined* as the displacement of pinball $B$ along this direction, needed to exactly eliminate the overlap, as shown in Figure 11b by the vector $p\hat{n}_{AB}$. The situation of Figure 11b is enlarged in Figure 11c for clarity.

### 2.2.2 Penetration

Then the penetration $p$ can be computed as follows. The distance $b$ is given by:

$$b = C_{AB} \cdot \hat{n}_{AB}, \quad (5)$$

the distance $h$ is such that:

$$h^2 = \|C_{AB}\|^2 - b^2 \quad (6)$$

and the distance $l$ is given by:

$$l = \sqrt{R_{AB}^2 - h^2}. \quad (7)$$

Finally, the penetration $p$ is given by:

$$p = l - b = \sqrt{(R_{AB}^2 - h^2 - C_{AB} \cdot \hat{n}_{AB}^2) - \|C_{AB}\|^2 + (C_{AB} \cdot \hat{n}_{AB})^2}, \quad (8)$$

‡. Again, this definition has opposite sign to the one given by Belytschko et al. ($\hat{n}_{AB} = (\hat{n}_B - \hat{n}_A) / \|\hat{n}_B - \hat{n}_A\|$), for consistency with the pinball method implementation in EPX described in [13].

* Belytschko et al. define the penetration as “the (relative) displacement of the two pinball centres along this direction, needed to exactly eliminate the overlap”.

5
It is important to check that the expression given for the penetration is symmetric, i.e. that by exchanging the two pinballs \( A \) and \( B \) (and by denoting the resulting quantities by a superposed prime) the same value of \( p \) is obtained. This is readily verified. In fact, one has:

\[
\begin{align*}
\hat{n}_{BA} &= -\hat{n}_{AB} \\
\hat{C}_{BA} &= -\hat{C}_{AB} \\
\|\hat{C}_{BA}\| &= \|\hat{C}_{AB}\| \\
R_{BA} &= R_{AB}
\end{align*}
\]  

(9)

so that instead of (5-7) we obtain:

\[
\begin{align*}
\hat{b}' &= \hat{C}_{BA} \cdot \hat{n}_{BA} = b, \\
\hat{h}'^2 &= \|\hat{C}_{BA}\|^2 - \hat{b}'^2 = \hat{h}^2, \\
\hat{\ell}' &= \sqrt{R_{BA}^2 - \hat{h}'^2} = \ell.
\end{align*}
\]

(10-12)

Finally one sees that:

\[
\hat{p}' = \hat{\ell}' - \hat{b}' = \hat{p}.
\]

(13)

### 2.3 Calculation of the rate of penetration

Some penalty methods involve not only the penetration \( p \), but also the rate of penetration \( \dot{p} \):

\[
\dot{p} = \frac{dp}{dt}.
\]

(14)

In reference [5], the following expression to compute \( p \) for a couple of contacting hexahedral elements (8 nodes each) \( A \) and \( B \) is given\(^\dagger\dagger\):

\[
\dot{p}_{AB} = \frac{1}{8} \sum_{I=1}^{8} [(\nu_{A1} v_{B1}) \cdot \hat{n}_{AB}]
\]

(15)

where \( I \) is the generic node of either \( A \) or \( B \), \( \nu_{AI} \) are the nodal velocities and \( \hat{n}_{AB} \) is the direction of penetration between the two pinballs, given by (4). This expression corresponds to assuming that “the” velocity associated with a pinball is the velocity at the centroid of the corresponding element. In fact, (15) can be re-written as:

\[
\dot{p}_{AB} = \frac{1}{8} (\sum_{I=1}^{8} v_{AI}) \cdot \hat{n}_{AB} - \frac{1}{8} (\sum_{I=1}^{8} v_{BI}) \cdot \hat{n}_{AB}.
\]

(16)

\(^\dagger\dagger\) Like for similar expressions in Section 2.2, we change sign here to the original definition given by Belytschko et al. \((\dot{p}_{AB} = \frac{1}{8} \sum_{I=1}^{8} [v_{BI} - v_{AI}) \cdot \hat{n}_{AB}]) for consistency with the pinball algorithm implementation in EPX described in reference [13].
But the velocity at any point \( P \) in the hexahedron element can be interpolated as:

\[
\mathbf{v}_P = \sum_{I=1}^{8} N_I \mathbf{v}_I
\]  

(17)

where \( N_I \) are the tri-linear shape functions:

\[
N_I(\xi, \eta, \zeta) = \frac{1}{8} (1 - \xi)(1 - \eta)(1 - \zeta)
\]  

(18)

over the parent element and \( \xi, \eta, \zeta \) are normalized coordinates. At the element centroid \( C \) it is \( \xi = \eta = \zeta = 0 \), so that from (18) \( N_I(C) = 1/8 \) and from (17) the velocity at the centroid is:

\[
\mathbf{v}_C = \frac{1}{8} \sum_{I=1}^{8} \mathbf{v}_I.
\]  

(19)

Therefore, the expression (15) or (16) can be re-written as:

\[
\dot{\mathbf{p}}_{AB} = (\mathbf{v}_A - \mathbf{v}_B) \cdot \mathbf{n}_{AB},
\]  

(20)

where \( \mathbf{v}_A, \mathbf{v}_B \) are the (interpolated) velocities at the centres of the two pinballs, which in this case coincide with the centroids of the corresponding elements. The right-hand side of (20) represents the relative velocity of the pinballs projected along the direction of penetration, and is indicated by the (signed) quantity \( \mathbf{v}_{ABn} \) in the example of Figure 12. In conclusion, the rate of penetration is:

\[
\dot{\mathbf{p}}_{AB} = \mathbf{v}_{ABn}.
\]  

(21)

Note that the penetration tends to increase when \( \mathbf{v}_{ABn} > 0 \), while it tends to decrease (incipient rebound) when \( \mathbf{v}_{ABn} < 0 \).

### 2.4 Interaction between a couple of pinballs

To summarize, the interaction between any couple of pinballs \( A, B \) is computed as follows.

- Compute the vector \( \mathbf{C}_{AB} \) by (1):

\[
\mathbf{C}_{AB} = \mathbf{C}_B - \mathbf{C}_A
\]  

(22)

- Compute the sum of the radii by (3):

\[
R_{AB} = R_A + R_B
\]  

(23)

- Check whether the two pinballs are in contact by (2):

\[
\|\mathbf{C}_{AB}\| < R_{AB}
\]  

(24)

- If the above inequality does not hold, then there is no interaction between these two pinballs: go to the next couple of pinballs. Else:
• Compute the penetration direction, i.e. the contact normal, by (4):

\[ \hat{n}_{AB} = \frac{(n_A - n_B)}{\|n_A - n_B\|} \]  

(25)

• Compute the distance \( b \) by (5):

\[ b = C_{AB} \cdot \hat{n}_{AB} \]  

(26)

• Compute the quantity \( h^2 \) by (6):

\[ h^2 = \|C_{AB}\|^2 - b^2 \]  

(27)

• Compute the distance \( l \) by (7):

\[ l = \sqrt{R_{AB}^2 - h^2} \]  

(28)

• Finally, compute the penetration \( p \) by (8):

\[ p = l - b \]  

(29)

• If the rate of penetration is required by the chosen penalty model, then:

\[ \dot{p}_{AB} = (v_A - v_B) \cdot \hat{n}_{AB} \]  

(31)

It is interesting to consider how the penetration between two pinballs varies as a function of the relative positions of the two pinballs. This is illustrated in Figure 13. We assume two pinballs \( A \) and \( B \) of different radii, for generality. The relative normal \( \hat{n}_{AB} \) is assumed vertical in the example. The Figure shows the variation of the penetration \( p = p\hat{n}_{AB} \) as the \( A \) pinball moves along a line normal to \( \hat{n}_{AB} \).

2.5 Calculation of the penalty force

When two pinballs \( A \) and \( B \) are in contact, a (repulsive) penalty force is applied. The expression given in reference [8] for the penalty force exerted by \( B \) on \( A \) is\( ^{++} \):

\[ F_{pA} = -(k_p p + k_v \dot{p})\hat{n}_{AB} \]  

(32)

\( ^{++} \) Again, we have changed sign here with respect to the work of Belytschko et al., due to our opposite definition of the contact normal \( \hat{n}_{AB} \).
while, of course, an equal in modulus and opposite force is exerted by $A$ on $B$:

$$E_{pB} = -E_{pA} = (k_p p + k_v \dot{p}) \hat{n}_{AB}.$$  \hfill (33)

The direction of the force coincides with the direction of penetration $\hat{n}_{AB}$, given by (4). The first term in the penalty force is proportional via a coefficient $k_p$ to the penetration $p$ given by (8), while the second term is proportional via a coefficient $k_v$ to the rate of penetration $\dot{p}$ given by (20).

### 2.5.1 Classical expressions of penalty coefficients in 3D

In [8] the following expression is proposed for the coefficient $k_p$:

$$k_p = \frac{\beta KS^2}{V}$$ \hfill (34)

where $K$ is the material’s bulk modulus (see below), $S$ is the “area of the impacted surface” and $V$ is the volume of the element. The quantity $\beta$ is a scaling (“tuning”) coefficient (dimensionless number) which should be prescribed by the user. The same expressions are given also in reference [5].

Let us consider the 3D case, where the pinballs are spheres of radius $R$. By assuming that the area $S$ is the cross-section of the pinball:

$$S = \pi R^2$$ \hfill (35)

and that the volume $V$ of the element is (at least approximately) equal to the volume of the pinball:

$$V = \frac{4}{3} \pi R^3.$$ \hfill (36)

the expression (34) becomes:

$$k_p = \frac{\beta KR}{V}$$ \hfill (37)

where the constant factor $(3\pi/4)$ has been incorporated in the $\beta$ parameter, which has to be tuned anyway. If the interaction occurs between two pinballs of different materials (different bulk moduli) and/or with different radii, one should use in place of (37) the expression (taken from refs. [5, 8]):

$$k_p = \frac{K_A R_A + K_B R_B}{2}$$ \hfill (38)

which accounts for the properties of both pinballs. Of course, (38) reduces to (37) if the two pinballs are identical in radius and material.
2.5.2 Penalty coefficients in 2D

In the 2D (plane) case, strictly speaking pinballs are circles. In this case in principle one would have in place of (35) and (36), respectively:

\[ S = 2R \]  \hspace{1cm} (39)
\[ V = \pi R^2 \]  \hspace{1cm} (40)

and (34) would become:

\[ k_p = \beta K \]  \hspace{1cm} (41)

where the factor \((4/\pi)\) has been incorporated in the \(\beta\) parameter, which has to be tuned anyway. This expression differs from (37) for the 3D case and is dimensionally incorrect because \(K\) has the dimensions of a pressure (N/m²), see below, while the stiffness \(k_p\) should be expressed in N/m (see e.g. eq. 32).

Unfortunately, reference [8] only reports the 3D case and no mention is made of the 2D case. In addition, no expression is provided in [8] (nor in any other of the cited references) for the coefficient \(k_v\). Therefore it is assumed that, if needed, \(k_v\) must be provided by the user.

The same expressions of the penalty force (32-38) are given also in reference [5] and in reference [6] (which, however, contains various evident typing errors in these formulas).

2.5.3 Classical expressions of penalty coefficients for shells

In reference [9], which deals with hierarchic pinballs mainly for contact between shell elements, different expressions from (32) are given for the penalty force:

\[ F_{pA} = F_{pA} \hat{C}_{AB} = F_{pA} \frac{C_{AB}}{C_{AB}}, \]  \hspace{1cm} (42)

\[ F_{pA} = \min(F_1, F_2) \]

where:

\[ F_1 = \begin{cases} \frac{\rho_A \rho_B R_A^3 R_B^3 \dot{p}}{\rho_A R_A^3 + \rho_B R_B^3 \Delta t} & \text{if } \dot{p} > 0 \\ 0 & \text{if } \dot{p} < 0 \end{cases} \]  \hspace{1cm} (43)

\[ F_2 = \frac{K_A K_B}{K_A + K_B} \left( \frac{R_A R_B}{R_A + R_B} \right)^{3/2} \]  \hspace{1cm} (44)

where \(\rho_A\), \(\rho_B\) are the densities of the two materials.
Note that according to reference [9], from the first of (42), the penalty force is always exerted in the
direction between the pinball centers \( \mathbf{C}_{AB} \), and not along the “penetration direction” (or contact
normal) \( \mathbf{n}_{AB} \) (4). This may be due to the fact that reference [9] uses hierarchic pinballs in shell ele-
ments. In fact, it is stated that “this choice enables to handle edge-to-surface, surface-to-surface and
edge-to-edge contact”.

2.5.4 Expressions of penalty coefficients in EUROPLEXUS for sliding surfaces

Finally, in EUROPLEXUS the following expressions are implemented in subroutine CALPEN
(called from FGLIS3, in turn called from GLIS3D) for contact computed with the so-called sliding
surface algorithm [1-2] using the penalty method (input directive \texttt{LINK DECO GLIS PENA ...}, see
User’s manual on page D2.180). This model is at the moment available only in 3D.

The contact penalty force includes only the term proportional to the penetration \( p \) :

\[
E_p = k_p p \mathbf{n}
\]  

(45)
The contact stiffness \( k_p \) is computed as follows from the stiffness of master elements.

For a continuum master element:

\[
k_p = \frac{\beta K S^2}{V}
\]  

(46)
(same expression as eq. 34), where \( \beta \) is a user-defined scaling factor, \( K \) is the material’s bulk modu-
lus (see below), \( S \) is the area of the contacting face (of the master element), and \( V \) is the volume of
the master element. Note that in (34) \( S \) and \( V \) represent the cross-section and volume of the pinball,
instead.

For a shell master element:

\[
k_p = \frac{\beta K S}{L}
\]  

(47)
where \( \beta \), \( K \), \( S \) have the same meaning as in (46) and \( L \) is the maximum length of the master ele-
ment’s (shell) edges.

When contact occurs between two pinballs or sub-pinballs with different characteristics (bulk modu-
lus, radius etc.), an expression such as (46) or (47) is evaluated for each of the pinballs obtaining thus
two values \( k_{pA} \), \( k_{pB} \). Then the penalty coefficient is taken as the \textit{minimum} of the two coefficients
\textit{(not as an average like in expression 38)}:

\[
k_p = \min (k_{pA}, k_{pB}).
\]  

(48)
This seems more physically intuitive than taking the average, since for a given inter-penetration of the two pinballs the contact force is clearly determined by the “weakest” of the two bodies, not by an average of the two material properties.

### 2.6 Bulk modulus

The bulk modulus $K$, or modulus of volume expansion, of a material subjected to a uniform (hydrostatic) pressure ($P$) field ($\sigma_x = \sigma_y = \sigma_z = -P$ and $\tau_{xy} = \tau_{yz} = \tau_{zx} = 0$) is the coefficient that relates the pressure $P$ (with the minus sign to conform to the usual stress rule for compression) to the volume expansion $e = \varepsilon_x + \varepsilon_y + \varepsilon_z$:

$$-P = Ke$$  \hspace{1cm} (49)

Therefore one can say that the bulk modulus characterizes the resistance of a material to a uniform pressure.

For an elastic material of Young’s modulus $E$ and Poisson’s coefficient $\nu$ the normal strains are related to the normal stresses by:

$$\varepsilon_x = \frac{1}{E} [\sigma_x - \nu (\sigma_y + \sigma_z)]$$

$$\varepsilon_y = \frac{1}{E} [\sigma_y - \nu (\sigma_x + \sigma_z)]$$ \hspace{1cm} (50)

$$\varepsilon_z = \frac{1}{E} [\sigma_z - \nu (\sigma_x + \sigma_y)]$$

By adding up these three equations and by denoting the sum of normal stresses $\Theta = \sigma_x + \sigma_y + \sigma_z$ one has:

$$e = \frac{1 - 2\nu}{E} \Theta$$ \hspace{1cm} (51)

Under a hydrostatic pressure $P$ field it is $\Theta = -3P$ and (51) becomes:

$$e = \frac{3(1 - 2\nu)}{E} P$$ \hspace{1cm} (52)

so that by comparing (49) and (52) one obtains the following expression for the bulk modulus:

$$K = \frac{E}{3(1 - 2\nu)}$$ \hspace{1cm} (53)

The bulk modulus has the same units as Young’s modulus, stress and pressure: Pa (i.e. N/m$^2$) in the standard system.
2.7 Distribution of penalty force on the element nodes

The penalty force (32) or (42) has to be distributed among the element’s nodes. In reference [5], which deals with 0-level pinballs embedded in 8-node hexahedra, the following formula is proposed:

\[ F_{pI} = \frac{1}{8} F_p \quad I = 1, \ldots, 8, \]  

i.e., the force is equally distributed among all the nodes \( I \) of the element. An alternative possibility is mentioned, whereby the force would be distributed only among the nodes of the impacted face of the element. The same formula is proposed also in reference [8].

In reference [6] a similar formula is proposed, but the force is (equally) distributed among “the four nodes of each element which are closest to the center of the other element”. This is because hexahedral element faces have four nodes.

In reference [9], which deals with hierarchic pinballs mainly for shell elements, the penalty force (given by eq. 42) is distributed onto all the element’s nodes (in this case a face has the same nodes as the whole element) according to the element’s shape functions.

2.8 Considerations on penalty force between contacting elements or pinballs

Consider a couple of contacting (regular) hexahedral elements in 3D, see Figure 14. Let \( S \) denote the contact surface (in this case the area of the element face), \( h \) the element side and \( V = Sh \) the element volume.

The contact stiffness according to (46) is:

\[ k_p = \frac{\beta KS^2}{V} = \frac{\beta KS}{V} = \frac{\beta KS}{h} \]  

The contact force for a penetration \( p \) is therefore:

\[ F_p = k_p p = \frac{\beta KS^2 p}{h} \]  

But \( p/h \) is the (engineering) strain in one of the elements in the direction of penetration:

\[ \frac{p}{h} = \varepsilon_h \]  

so that (56) can be re-written:

\[ F_p = \beta K \varepsilon_h S \]
From (49) we see that the term $K\varepsilon_h$ is proportional (through a coefficient which can be included in the parameter $\beta$) to the (engineering) stress $\sigma_h$ in the element along the direction of penetration (assuming a linear elastic material behavior):

$$K\varepsilon_h \approx \sigma_h$$

so that (58) becomes:

$$F_p = \beta\sigma_h S$$

From this expression we see that the contact penalty force is proportional to the force $(\sigma_h S)$ that would arise in the element when subjected to a displacement equal to the penetration $p$ on one of its faces, assuming a linear elastic material.

From these considerations and from physical intuition it appears therefore that the value of the parameter $\beta$ should be of the order of 1. The value assumed by default in the code is 1.

Consider now the same couple of contacting elements as before, but let the contact be described by (zero-level) pinballs, see Figure 15. The contact stiffness is given by (37):

$$k_p = \beta KR$$

where $R$ is the radius of the pinballs ($R = h/2$). This, multiplied by the penetration $p$, gives the contact force which must then be distributed on the nodes. By default the contact force is assumed to act at the pinball centre, so it is distributed equally on the 8 nodes of the element. Another possibility, shown in Figure 15, is to apply the force at the center of the contacting face and therefore distribute it only among the four nodes of the face.

Finally, let us examine the case of hierarchic pinballs at a generic level $L$, see Figure 16. Recall that the parent pinball has level 0, and that at each successive refinement the radius of the pinballs is divided by 2. In the example a level $L = 2$ has been assumed.

In the case of “flat” “perfect” contact shown in the Figure, there are $2^L$ contacting pinball couples along each spatial direction of the (quadrilateral) face of the hexahedron, i.e. $2^{2L}$ contacts altogether on the 3D quadrilateral face, i.e. 16 contacts in the example (of which only the four “frontal” ones are visible).

Intuitively, for a given penetration the total contact force on the face should be (roughly) the same, irrespective of the chosen hierarchy level of the pinballs (in fact, hierarchic pinballs are used only to make contact detection more precise geometrically). This force is indicated as $F_{p0}$ in Figure 16 at
the zero-level (just one contact). Since for a quadrilateral face at level \( L \) there are \( 2^{2L} \) contacts, the single contact force has to be:

\[
F_{pL} = \frac{1}{2^{2L}} F_{p0}
\]  

(62)

Since the penetration does not depend upon the level, it is the contact stiffness (61) that has to be scaled by the number of contacts:

\[
k_{pL} = \frac{1}{2^{2L}} k_{p0}
\]  

(63)

Since the pinball radius varies with the level according to:

\[
R_{L} = \frac{R_0}{2^{L}}
\]  

(64)

one finds that for each single pinball at level \( L \) of an impacting quadrilateral face the penalty contact force is given by:

\[
F_{pL}^{\text{quad}} = k_{pL}^{\text{quad}} p = \frac{1}{2^{L}} \beta KR_{L} p.
\]  

(65)

At zero-level \( 2^{L} = 1 \) and one recovers the expression for the parent pinball, see eq. (61).

Similarly, for an impacting triangular face at level \( L \) the number of descendent pinballs (i.e. the number of contacts) is \( 2^{L} (2^{L} + 1)/2 \). Therefore:

\[
F_{pL}^{\text{tria}} = k_{pL}^{\text{tria}} p = \frac{2}{2^{L} + 1} \beta KR_{L} p.
\]  

(66)

A similar expression remains to be derived for the 2D case.

2.9 Treatment of rebound with the penalty method

The treatment of rebound in the description of an impact problem by the penalty method deserves some attention.

When using the Lagrange Multipliers method, rebound must be specifically addressed by an ad hoc strategy. In reference [13] two alternatives are considered: the so-called a-priori rebound or the a-posteriori rebound.

- In the first case, which is also by default the one chosen in the code, the relative penetration velocity of the two pinballs is estimated and, depending on the sign that this quantity assumes, the contact constraint is or is not enforced. The constraint is enforced when the penetration tends to increase, but is not enforced when the penetration starts to diminish (incipient rebound).
• In the second case, a constraint is always retained as long as penetration occurs, but the sign of the contact forces is inspected after solution of the system of constraints and, if the force is attractive, then the force is set to zero. This method works in very simple cases where each contact constraint is independent from any other constraints, but it may fail in more complex situations, and therefore the a priori method is preferred (and is assumed by default).

The type of rebound algorithm can be chosen by an input option: OPTI PINS REB1 chooses the a priori rebound (which is usually redundant since this is the default). OPTI PINS REB2 chooses the a posteriori rebound. Finally, OPTI PINS NORB completely disables the rebound treatment.

When using the penalty approach, it is necessary to let the contact (penalty) force act as long as there is penetration, irrespective of the relative velocity of the pinballs. In other words, we do not want any special treatment of rebound with penalty. Therefore, any rebound-related options (REB1, REB2, NORB) are simply ignored if the penalty approach for pinball contact is specified.

2.10 Generalized ASN algorithm for hierarchic pinballs

In a previous Section (2.1) the ASN algorithm proposed by Belytschko and Law in 1985 [3] has been recalled for the case of zero-level (parent) pinballs. In their paper on hierarchic pinballs [9] (mostly focusing on contacts between shells) Belytschko and Yeh use the ASN algorithm only to find the external faces of the domain. In fact, in their implementation penalty forces are always directed along the line that joins the centers of the two contacting pinballs.

Here we want to explore the possibility of using a generalized ASN concept also for hierarchic pinballs, both with continuum and with shell elements. The original ASN algorithm as described above will be used only for contact between 0-level pinballs, i.e. in applications without hierarchic pinballs (e.g. perforation of bulky structures with erosion), which typically do not require a high precision of contact detection anyway, and in which too precise a representation of the eroded surface would probably be a drawback rather than an advantage.

2.10.1 Single element

Consider first a single (isolated) 2D quadrilateral, as shown in Figure 17. At zero level (left part of the Figure), the standard ASN algorithm yields the elemental face normals shown in red, the nodal normals shown in green, and the element (or pinball) assembled normal shown in black (at the pinball center). The element normal vanishes in this case and is therefore not visible.

Passing to refinement level 1 (central part of the Figure), four descendent pinballs are created. Each of these is adjacent to a “vertex” of the element, and therefore they are called vertex pinballs (marked
by a $V$ on the drawing). It seems natural that a vertex pinball has associated the nodal normal of the corresponding vertex, as shown in the Figure.

Passing to refinement level 2 (right part of the Figure), twelve descendent pinballs are created. Of these, four are vertex pinballs ($V$) while the others are face pinballs ($F$) because they are adjacent to a face of the parent element. It seems natural that a face pinball has associated the element face normal of the corresponding face, as shown in the Figure. Note that, upon refinement, only descendent pinballs which are in contact with an external face of the element are generated in the recursive pinball splitting process: “internal” descendent pinballs are not generated. This refinement procedure could go on at will, with the same strategy.

2.10.2 Simple mesh
Now consider a simple mesh of 4-node quadrilaterals as shown in Figure 18. At zero-level, there are four (parent) pinballs, one for each element of the mesh. These pinballs now look like vertex pinballs (with respect to the mesh) and have non-zero ASN normals, which happen to coincide with the normals at the corresponding vertices of the mesh. At refinement levels 1 and 2 the resulting descendent pinballs and the associated normals are as shown in the central and right parts of Figure 18, respectively.

2.10.3 Flat aligned contact
The case of “flat” “aligned” contact is shown in Figure 19. For “internal” contacting pairs of pinballs the penetration direction (contact normal) is normal to the interface, as expected. It is important to note that the same holds also for contacting pairs at the vertices, despite the fact that the pinball normals have a certain inclination at vertices.

2.10.4 Flat mis-aligned contact
In the case of moderate mis-alignment of the pinballs (by less than a pinball radius) as shown in Figure 20, the algorithm continues to give the same penetration directions (contact normals) as in the perfectly aligned case.

2.10.5 Vertex-to-face contact
The case of vertex-to-face contact is illustrated in Figure 21. The standard ASN formula for the calculation of the penetration direction (contact normal) gives a vector which is reasonably normal to the face. Alternatively, one can define a rule stating that, in case of contact between a vertex pinball and a face pinball, the penetration direction (contact normal) is the normal to the face rather than an average of the two normals.
2.11 Contact rules

On the basis of the considerations detailed in the previous Sections, we tentatively define a set of rules for the contact between bodies by the (basic or hierarchic) pinballs method. These rules exploit the concept of an Assembled Surface Normal (ASN) associated (whenever this makes sense) with each parent and with each descendent pinball. They are developed primarily for use with a contact enforcement formulation based on penalty forces. However, their use is not excluded also in a Lagrange multipliers context. The advantage of penalty with respect to Lagrange multiplier (LM) formulations is that they avoid all the problems related with contact redundancies, which lead to ill-conditioned matrices in the LM method.

2.11.1 ASNs and pinball types

To each parent pinball is associated a unique Assembled Surface normal (ASN), obtained with (a variant of) the original algorithm by Belytschko and Law of Section 2.1 and representing an (average) normal direction to the body surface in the associated finite element \( e \). The exact implementation of the ASN algorithm is described below in Section 2.11.2. Tentatively, such normal is normalized, i.e. it is represented by a vector of unit length \( \hat{\mathbf{n}}_e \)—except in the case where a normal cannot be defined, in which all normal components are set to 0. This happens when the assembly process yields a vanishing resultant vector, i.e. for the pathological configuration of Figure 10, for an “isolated” finite element (not connected to any other elements) and, finally, for a material point (single-node element).

Also to each descendent pinball is associated a unique normal (always normalized to unit length, since zero-length normals cannot occur for descendent pinballs), whose calculation is detailed below in this paragraph, according to the “type” of pinball introduced next.

Therefore, the ASN associated with a parent pinball (\( E \) pinball, see below) can have either length 1.0 or length 0.0 (when it is undefined). Instead, the ASN associated with a descendent pinball (\( F \), \( C \) or \( V \), see below) is always defined and therefore it always has length 1.0.

Types of pinballs

In addition, each pinball (parent or descendent) has an associated type: “element” (\( E \)), “face” (\( F \)), “corner” (\( C \)) or “vertex” (\( V \)).

Element (\( E \)) pinballs are parent (\( L = 0 \)) pinballs. They are associated with an (entire) element.

The other pinball types (\( F \), \( C \), \( V \)) are descendent (\( L > 0 \)) pinballs and are associated with a portion of an element.
As shown in Figure 28 for the various cases both in 2D and in 3D, \( F \) pinballs are located along an (external) face. \( C \) pinballs occur only in 3D and are located along an (external) corner, the intersection line between two (external) element faces. Finally, \( V \) pinballs are adjacent to a vertex of the element, the intersection point (mesh node) of two (external) element faces in 2D, or of three (external) element faces in 3D.

Both the ASN and the type of each pinball have to be re-computed at each time step because of changes in the mesh configuration (large motions and large strains) and in the connectivity (erosion, adaptivity). In particular, when an element fails and is eroded some previously internal element faces become external, some new pinballs have to be created on these new external surfaces and all the neighboring nodal normals, ASNs and pinball types are affected.

**ASNs for the various pinball types**

For an \( F \) pinball, the associated normal is the (unit) normal \( \hat{n}_F \) to the corresponding face. The calculation of these normals is straightforward for 2D faces and for triangular 3D faces, which are always planar. In the case of a quadrilateral 3D face, which may present some warping, “the” normal is defined here as the vector product of the two medians of the face, which intersect each other at the element centroid.

For a \( V \) pinball, the associated normal is the (unit) normal \( \hat{n}_V \) to the corresponding vertex, which coincides with a node of the mesh. These normals are computed and used to determine the ASN and are therefore readily available.

For a \( C \) pinball, the associated normal is the (unit) normal to the corresponding corner \( \hat{n}_C \). This could in principle be obtained by assembling and normalizing the normals to the two 3D (external) faces which form the corner under consideration. However, this task is complicated, in general, because the two external faces can belong to different elements (and therefore to different parent pinballs). To simplify this task, the algorithm tentatively implemented exploits the knowledge of the vertex normals. The two vertices of the corner are identified and the average of the two vertex normals is built:

\[
\hat{n}_a = \frac{\hat{n}_1 + \hat{n}_2}{2}.
\]

(67)

Then, this vector is projected onto the plane normal to the corner (to eliminate any out-of-plane components), and finally it is normalized to unit length.

The (unit) tangent vector, directed along the corner of vertices \( \hat{x}_1, \hat{x}_2 \) is given by:

\[
\hat{t} = \frac{\hat{x}_2 - \hat{x}_1}{\|\hat{x}_2 - \hat{x}_1\|}.
\]

(68)
The component of the average normal along the tangent is the following vector:

\[ n_{at} = (n_a \cdot \hat{t}) \hat{t}. \]  

Finally, the unit normal to the corner is:

\[ \hat{n}_c = \frac{n_a - n_{at}}{\|n_a - n_{at}\|}. \]

For an \textit{E} pinball, the associated normal is the corresponding ASN. This normal can have either unit or zero length. Instead, \textit{F}, \textit{V} and \textit{C} pinball normals always have unit length.

### 2.11.2 Implementation of the ASN algorithm

The implemented version of the ASN algorithm is as follows.

- **Build external element face normals**, i.e. build face normals \textit{only along external faces of the mesh}, see Figure 22 and comments at the end of this paragraph. These normals are oriented “outgoing” from the element and are \textit{not} normalized, i.e. the norm of the vector equals the length (in 2D) or area (in 3D) of the face.

- Assemble external element face normals at each element’s nodes, to obtain (directly) \textit{global external nodal normals}, see Figure 23. These are not (yet) normalized.

- Assemble nodal normals to obtain \textit{element normals}, see Figure 24. These are not normalized.

- Normalize the resulting element normals to unit length, to obtain \textit{normalized element normals}, see Figure 25.

- Normalize the nodal normals to unit length to obtain \textit{normalized nodal normals}, see Figure 26.

It is important to note that the normalization of the nodal normals is performed last, i.e. \textit{after} the use of these normals to obtain the element normals (which are also normalized after assembly). The example of Figure 27, consisting of a stand-alone triangular element, illustrates the two alternatives. With the algorithm as described above (top part of the Figure), the resulting element normal is zero (as expected, because the element is not connected to any other elements). Instead, if one would normalize the nodal normals before assembling them one would obtain a non-zero element normal (which seems unreasonable for a stand-alone element), as shown in the bottom part of the Figure.

Note that the fact that face normals are built only along the external faces, and not also along the internal ones like in the original implementation of the ASN algorithm by Belytschko and law (Section 2.1) is due to two reasons. First, external and internal faces are already identified for other purposes in the code and are readily available, so it would be a waste of CPU to compute and assemble normals at internal faces/nodes. Second, in the present implementation the ASN and in particular the
occurrence of a vanishing normal is not used as a means of detecting internal nodes or faces. Vanishing ASNs or, more precisely, vanishing pinball normals can still occur, but they indicate either a material point or an isolated (stand-alone) continuum element.

2.11.3 Penetration direction (contact normal)

Two pinballs \( A \), \( B \) penetrate each other if, according to (2):

\[
\|C_{AB}\| < R_A + R_B.
\]  

When this occurs, the penetration direction (or contact normal) \( \hat{n}_{AB} \) can in principle be computed as follows (but for the algorithm actually implemented in the code see Section 4.1):

- If \( \|n_A\| = \|n_B\| = 0 \), i.e. if both pinballs are (element) pinballs with an undefined associated normal, then the penetration direction (contact normal) is the line joining the two pinball centres (see Figure 29):

\[
\hat{n}_{AB} = \frac{C_{AB}}{\|C_{AB}\|}
\]  

- Else, if \( (\|n_A\| = 1 \text{ and } \|n_B\| = 0) \), or if \( (\|n_A\| = 0 \text{ and } \|n_B\| = 1) \), i.e. if only one of the two pinballs has a defined normal, then the penetration direction (contact normal) coincides with either \( \hat{n}_A \) or \( -\hat{n}_B \), respectively, see Figure 30. This can be treated either as a special case, or by applying the general equation (4), since the result is the same.

- Else \( \|\hat{n}_A\| = 1 \) and \( \|\hat{n}_B\| = 1 \), i.e. both pinballs have a defined (unit) normal. Then the types of the two pinballs must be considered:
  - If both pinballs are \( E \) pinballs, then the penetration direction (contact normal) is computed with the general expression (4):

\[
\hat{n}_{AB} = (n_A - n_B) / \|n_A - n_B\|
\]  

  - Else if one pinball is an \( E \) pinball and the other is not an \( E \) pinball, then the normal associated with the other pinball (\( F \), \( C \) or \( V \)) “has the precedence” over this one and determines the penetration direction (contact normal). This case occurs when the \( E \) pinball is a parent (level \( L = 0 \)) pinball associated either with a continuum or with a structural (shell/beam etc.) element (but not with a material point, since here the normal is assumed to be non-zero), while the other pinball (\( F \), \( C \) or \( V \)) is necessarily a descendent pinball, i.e. at a level \( L > 0 \). 

  - Else neither \( A \) nor \( B \) is an \( E \) pinball. Then:

* If both pinballs are face pinballs, then the penetration direction (contact normal) is computed with the general expression (4), see Figure 31.
* Else if one pinball is a face pinball and the other is a corner or vertex pinball, then the normal associated with the face pinball “has the precedence” over the other one and determines the penetration direction (contact normal), see Figure 32.

* Else if both pinballs are corner pinballs, then the penetration direction (contact normal) is computed with the general expression (4).

* Else if one pinball is a corner pinball and the other is a vertex pinball, then the normal associated with the corner pinball “has the precedence” over the other one and determines the penetration direction (contact normal).

* Else both pinballs are vertex pinballs. Then, the penetration direction (contact normal) is computed with the general expression (4).

The precedences are summarized in the following Table. In abscissa is represented the type of the first pinball and in ordinate the type of the second pinball (obviously, the table is symmetric with respect to the diagonal).

<table>
<thead>
<tr>
<th></th>
<th>E</th>
<th>V</th>
<th>C</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>E</td>
<td>V</td>
<td>C</td>
<td>F</td>
</tr>
<tr>
<td>V</td>
<td>V</td>
<td>V</td>
<td>C</td>
<td>F</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

Table 1 - Summary of precedences.

### 2.12 Implementation of the contact force calculation

The implementation of the penalty contact force evaluation is detailed hereafter. As anticipated in Section 2.8, physically the contact force is given by an expression of the general form (60):

$$F_p = \beta \sigma_h S$$  \hspace{1cm} (74)

where:

- $\beta$ is the penalty coefficient (which should be roughly of the order of 1),
- $\sigma_h$ is the normal stress generated by the penetration,
- $S$ is the contact surface.
By assuming for simplicity a linear elastic material, the normal stress generated by the penetration can be approximated as (see also eqs. 57 and 59):

\[ \sigma_h \approx K \varepsilon_h = \frac{Kp}{h} \]  

(75)

where:

- \( K \) is the material’s bulk modulus (close to Young’s modulus \( E \), see eq. 53),
- \( \varepsilon_h \) is the engineering strain in the penetration direction (i.e. along the contact normal),
- \( p \) is the penetration,
- \( h \) is the element’s height in the penetration direction (i.e. along the contact normal).

By combining (74) and (75) one gets (56):

\[ F_p = \beta KS \frac{h}{p} \]  

(76)

from which one sees that the force \( F_p \) will be expressed in N if \( \beta \) is a non-dimensional coefficient, \( K \) is expressed in Pa, \( S \) in m² and \( h \) and \( p \) in m.

2.12.1 3D continuum

In the case of contact between 3D continuum elements:

- \( S \) (contact surface) can be taken approximately equal to the pinball’s (parent or descendent) cross-section (with \( R \) the pinball’s radius):

\[ S \approx \pi R^2 \]  

(77)

- \( h \) (height of the element) can be taken approximately equal to the parent pinball’s diameter, i.e. twice the parent pinball’s radius \( R_0 \):

\[ h \approx 2R_0 \]  

(78)

Then (76) becomes:

\[ F_p = \beta K \frac{\pi R^2}{2R_0} p. \]  

(79)

Note that this formula is valid both for parent and for descendent pinballs and does not require any adjustment for the pinball level. In fact, the level is included in the \( S \) quantity, which depends upon the descendent’s radius \( R \).
If $L$ is the level of the descendent pinball ($L \geq 0$), then it is approximately (see 64):

$$ R \approx \frac{R_0}{2^L} \quad \text{(80)} $$

so that:

$$ R^2 \approx \frac{R_0^2}{2^{2L}} \quad \text{or} \quad \frac{R^2}{R_0^2} \approx \frac{1}{2^{2L}} \quad \text{(81)} $$

and (79) can also be written as:

$$ F_p = \beta K \frac{p R_0}{2^{2L}}. \quad \text{(82)} $$

Alternatively, one can express $R_0$ as a function of $R$ from (80):

$$ R_0 \approx 2^L R \quad \text{or} \quad \frac{R_0^2}{R^2} \approx \frac{2^L}{2^L R^2} = \frac{R}{2^L}, \quad \text{(83)} $$

so that (79) becomes also:

$$ F_p = \beta K \frac{p R}{2^{2L}} \quad \text{3D continuum} \quad \text{(84)} $$

The expression (84) is more efficient than (82) because it does not require the evaluation of the parent pinball’s radius $R_0$.

Note that these formulas (82) or (84) are valid for 3D continuum elements \textit{whatever the shape of their faces} (i.e. both for quadrilateral and for triangular faces).

### 2.12.2 2D continuum

In the case of contact between 2D continuum elements, one should distinguish between the plane stress / plane strain case and the axisymmetric case.

In the plane strain or plane stress case, let $t$ be the thickness of the element in the direction normal to the plane (usually it is taken $t = 1$). The contact force is still given by (74) and the stress is still given by (75), so that expression (76) is still valid.

However, now for the contact surface $S$ we have, in place of (77):

$$ S \approx 2Rt \quad \text{(85)} $$
with $R$ the pinball radius (parent or descendent). The height $h$ of the element is still approximately equal to the parent pinball’s diameter (see eq. 78), so that from (76) we obtain, in place of (79):

$$F_p = \beta K \frac{R_t}{R_0} p.$$  \hspace{1cm} (86)

This expression is dimensionally correct because, if $R$, $t$, $R_0$ and $p$ are expressed in m and $K$ is expressed in Pa, then $F_p$ results expressed in N.

The previous relations (80), (81) and (83) between $R$ and $R_0$ are valid also in 2D, so that eq. (86) can be re-written as:

$$F_p = \beta K \frac{t}{2} p \hspace{1cm} 2D \text{ plane stress / plane strain continuum}$$ \hspace{1cm} (87)

In the 2D axisymmetric case, the same expressions derived above for the plane strain / plane stress case are valid, if one replaces the (constant) element thickness $t$ by the (local) azimuthal thickness (“mean” radius) $r_M$, which is in this case given by the $x$-coordinate of the pinball’s center $x_C$:

$$F_p = \beta K \frac{x_C}{2} p \hspace{1cm} 2D \text{ axisymmetric continuum}$$ \hspace{1cm} (88)

### 2.13 New input directives

The ASN algorithm described in the present report is activated by a new option ASN:

OPTI PINS ... ASN

From the User’s manual: “The so-called “assembled surface normal” (ASN) algorithm of Belytschko and Law (1985) is used to compute a unique (normalized) normal to each external node of the mesh portion subjected to contact, and a unique (normalized) normal to each pinball (parent or descendent). The penetration direction (or contact normal) between contacting pinballs is then computed using the ASNs of the two pinballs according to a set of rules. This ameliorates the treatment of flat contact, especially in conjunction with a penalty formulation to compute the contact forces. This option cannot be used together with (is an alternative to) options FNOR, CNOR (and its sub-options), or SNOR.”

**Visualization directives**

New visualization commands in the built-in OpenGL graphical module of the code are provided in order to check the ASN quantities. These can either be activated interactively (right-click and then choose Geometry → Pinballs sub-menu), or in batch mode via the SCEN directive:

SCEN ... PINB ... PARE CDES ... NORM ... NASN PASN DASN
where PARE visualizes the parent pinballs, CDES the contacting descendent pinballs, NORM the contact normals, NASN the nodal ASNs (assembled surface normals), PASN the pinball ASNs for the parent pinballs, and DASN the ASNs for the contacting descendents.

**Color code for pinballs**

In order to distinguish the various types of pinballs and sub-pinballs the coloring code shown in Table 2 is adopted. Pinballs are represented as glass-like (semi-transparent) spheres of the color listed in the Table.

<table>
<thead>
<tr>
<th>Pinball type</th>
<th>Status or sub-type</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parent</td>
<td>active</td>
<td>GREEN</td>
</tr>
<tr>
<td></td>
<td>not active</td>
<td>RED</td>
</tr>
<tr>
<td>Descendent</td>
<td>Element</td>
<td>RED</td>
</tr>
<tr>
<td></td>
<td>Vertex</td>
<td>MAGENTA</td>
</tr>
<tr>
<td></td>
<td>Corner</td>
<td>CYAN</td>
</tr>
<tr>
<td></td>
<td>Face</td>
<td>BLUE</td>
</tr>
</tbody>
</table>

*Table 2 - Color code for the different pinball types.*
Figure 1 - Continuum elements mesh.

Figure 2 - Element face normals.
Figure 3 - Assembled element nodal normals.

Figure 4 - Assembled global nodal normals.
Figure 5 - Assembled global element normals.

Figure 6 - Normalized element normals.
Figure 7 - Normalized global nodal normals.

Figure 8 - Assembled global element normals (variant).

Figure 9 - Normalized global element normals (variant).
Figure 10 - Pathological case (insufficient discretization).

Figure 11 - Penetration between two pinballs.
\[ p = p \hat{n}_{AB} \]

\[ \dot{p} = \dot{p} \hat{n}_{AB} = v_{AB} \hat{n}_{AB} = [ (v_A - v_B) \cdot \hat{n}_{AB} ] \hat{n}_{AB} \]

Figure 12 - Rate of penetration between two pinballs.

Figure 13 - Variation of the penetration between two pinballs.
Figure 14 - Contacting hexahedra in 3D.

Figure 15 - Contact in 3D computed by zero-level pinballs.
Figure 16 - Contact in 3D computed by hierarchic pinballs.

Figure 17 - ASN with hierarchic pinballs on an isolated 2D quadrilateral.
Figure 18 - ASN with hierarchic pinballs on a simple mesh of 2D quadrilaterals.

![Diagram showing elemental face normal, nodal normal, and pinball normal]

Figure 19 - “Flat” aligned contact with hierarchic pinballs.

\[
\hat{n}_{AB} = \frac{n_A - n_B}{\|n_A - n_B\|}
\]

![Diagram showing internal contact and vertex contact with pinball normals and penetration normals]
Figure 20 - “Flat” misaligned contact with hierarchic pinballs.

Figure 21 - Vertex-to-face contact with hierarchic pinballs.
Figure 22 - Element external face normals (built only along external faces).

Figure 23 - Assembled global external nodal normals.
Figure 24 - Assembled global element normals.

Figure 25 - Normalized element normals.
Figure 26 - Normalized global external nodal normals.

Figure 27 - Example of stand-alone triangle.
Figure 28 - Pinball types.

- Pinballs for 3D continuum elements
- Pinballs for 2D continuum elements
- Pinballs for 2D shell/beam/bar and for 3D beam/bar elements
- Pinballs for 3D shell elements

Parent pinball: $E$

Descendent pinballs: $F$, $V$, $C$

(no descendents)
Figure 29 - Penetration direction (contact normal) between two “element” pinballs.

\[ \hat{n}_{AB} = \frac{C_{AB}}{\|C_{AB}\|} \]

Material point:
\[ \|n_B\| = 0 \]

Figure 30 - Penetration direction (contact normal) between two pinballs of which only one has a defined normal.

\[ \hat{n}_{AB} = \hat{n}_A \]

Material point:
\[ \|n_B\| = 0 \]

\[ \hat{n}_{AB} = -\hat{n}_B \]

Material point:
\[ \|n_A\| = 0 \]
Figure 31 - Penetration direction (contact normal) between two face pinballs.
Figure 32 - Penetration direction (contact normal) between a face pinball and a vertex pinball.
3. Contact algorithm for shell/beam/bar elements

We now consider the contacts involving (at least one) non-continuum elements, i.e. a shell, beam or bar element (but not a material point, which can be considered as a special case of continuum element).

3.1 2D case

Let us start from the simplest case, which occurs in 2D. In two spatial dimensions shell, beam and bar elements are all represented by a SEG2 element shape, i.e. a 2-node segment, see Figure 33. The element has two (mutually opposite) faces (with two nodes each) and two edges (with one node each). Other similar elements (in any number, including 0) may be attached to an edge, thus leading in general to element junctions, which are not possible with continuum elements and therefore have not been considered in the previous Chapter. Such elements are called adjacents.

When building element face normals for shell/beam/bar elements, only one of the two faces must be considered. We will consider only the first face, by convention. The second face is equal and opposite to the first one. Of course, unlike the case of continuum (bulky) elements, here the orientation of the normal is known only apart from the sign: one normal direction or the opposite one can be obtained just depending on connectivity, i.e. on the way in which the element nodes are listed in the geometry file.

Therefore, the question arises whether it makes sense or not to build nodal normals for such elements. In Figure 33 we consider the simplest possible case, just two SEG2 elements $e_1$, $e_2$ sharing a common node $I$. Obviously the assembled normal at the common node depends upon the orientation of the element normals, whose sign is arbitrary as mentioned. Therefore, the result would depend upon the specific numbering of the elements, see e.g. cases A and B in the Figure, and this is clearly unacceptable. A similar problem would also occur in the case of junctions, i.e. of three or more elements sharing the same node.

Therefore, it seems preferable to assume that nodal normals cannot be computed (and therefore are simply set at 0.0) for SEG2 elements.

It remains to establish what should be done upon pinball refinement, i.e. at levels $L > 0$. The following strategy is tentatively assumed.

**ASN for pinballs of shell/beam/bar elements in 2D**

- At the parent level, shell/beam/bar elements in 2D have one element normal each, which coincides (by convention) with the normal to the first face of the element (the second face is just opposite).
• The nodal normals are not assembled so they remain at 0.0.

• Upon refinement of a shell/beam/bar pinball in 2D, some \( V \) (vertex) sub-pinballs and some \( F \) (face) sub-pinballs are generated.

• The \( V \) pinballs are those “close” to an edge without any adjacents. All other descendent pinballs are \( F \) pinballs.

• The \( V \) pinballs are assigned a zero ASN, so that in contact calculations the centers-joining line algorithm will be preferably adopted.

• The \( F \) pinballs have the same ASN as the corresponding parent pinball (i.e. the normal to the first face of the parent element, which is always non-zero for these elements).

This algorithm is summarized in the example of Figure 34. For generality, a structure with several elements is considered. Nodes \( I \), \( M \) and \( N \) are edges without any adjacents. Node \( J \) is a regular node (at which two elements are joined), while node \( K \) is an example of junction (three or more elements meeting at a point).

Note that the \( V \) pinballs generated according to the above procedure are descendent pinballs with a zero associated ASN. This is contrary to the rule assumed in the previous Chapter for the case of continuum elements, and the corresponding cases will have to be treated appropriately in the general contact algorithm.

3.2 3D case

In 3D we must distinguish two cases depending upon the element shape: beams/bars (SEG2 shape), or shells (TRI3 or QUA4 shape).

3.2.1 Beam/bar elements in 3D

The shape of these elements (SEG2) is the same as for their corresponding 2D elements. However, the treatment is slightly different because it is impossible to define a normal to these elements in 3D (while it was possible in 2D). Only the plane containing the normal can be defined: this is the plane normal to the segment representing the element.

**ASNs for pinballs of beam/bar elements in 3D**

• Beam/bar elements in 3D have zero ASN both at the parent and at the descendent level.

• The nodal normals are not assembled so they remain at 0.0.

• Upon refinement of a beam/bar pinball in 3D, some \( V \) (vertex) sub-pinballs and some \( C \) (corner) sub-pinballs are generated.
• The $V$ pinballs are those “close” to an edge without any adjacents. All other descendent pinballs are $C$ pinballs.

• Both the $V$ pinballs and the $C$ are assigned a zero ASN.

This algorithm is summarized in the example of Figure 35, which is the equivalent in 3D of Figure 34. For generality, a structure with several elements is considered. Nodes $I$, $M$ and $N$ are edges without any adjacents. Node $J$ is a regular node (at which two elements are joined), while node $K$ is an example of junction (three or more elements meeting at a point).

Like for 2D SEG2-shaped elements, note that both the $V$ and the $C$ pinballs generated according to the above procedure are descendent pinballs with a zero associated ASN, contrary to the rule assumed in the previous Chapter for the case of continuum elements.

Finally, note that due to the occurrence of $C$ (corner) pinballs with an associated zero ASN, we may be faced with the contact between two such pinballs. Since the ASNs are zero, the general formula (4) cannot be used. In this case, instead of using as penetration direction the line connecting the pinball centers (see Figure 29 and eq. 96) it is more appropriate (and more accurate) to use the line joining the closest points on the two segments, as detailed in Appendix A.

3.2.2 Shell elements in 3D

The shape of shell elements in 3D is either TRI3 (triangle with three nodes) or QUA4 (quadrangle with four nodes). These elements have two (mutually opposite) faces with three or four nodes, respectively, and either three or four edges (respectively), each having the shape of a segment with two nodes.

A normal to the element is defined as the normal to the first face, by convention. For the triangle, this normal is geometrically well defined (apart from the sign of course) since the triangle lies in a plane. Instead, the quadrilateral may be warped so that its four nodes do not lie in a common plane. The normal is taken by convention as the perpendicular to the plane defined by the two medians of the quadrilateral: these two lines intersect in a point, the element centroid, and therefore they always lie on a plane.

The calculation of ASNs for the triangular and quadrilateral shells (considered as stand-alone elements) is shown in Figures 36 and 37, respectively, and is detailed hereafter. The case of two elements connected along an edge is shown in Figure 38.

**ASNs for pinballs of shell elements in 3D**

• At the parent level, pinballs are $E$ pinballs. The associated ASN is taken as the normal to the first face of the element, by convention.
• Nodal ASNs are not assembled so they remain at zero.

• Upon refinement of a shell element in 3D, some \( F \) (face) sub-pinballs, some \( C \) (corner) sub-pinballs and some \( V \) (vertex) sub-pinballs are generated.

• The \( V \) pinballs are close to the element vertices. They are assigned zero ASNs.

• The \( C \) pinballs are near the element corners without any adjacents. They are also assigned zero ASNs.

• The \( F \) pinballs are either in the interior of the element or near the element corners with adjacents. (see e.g. Figure 38). They have the same ASN as the parent pinball, i.e. the normal to the (first) element face.

3.3 Calculation of the contact normal

The calculation of the contact normal (or penetration direction) \( \vec{n}_{AB} \) for a couple of inter-penetrating pinballs \( A \) and \( B \) of which at least one is associated with a non-continuum (shell/beam/bar) element, requires some more care than with continuum elements (or material points).

In fact, for (a pinball associated with) a continuum element, the associated ASN (resulting from the ASN algorithm described in the previous Sections) should be “reasonably” oriented outwards the corresponding body, as shown in Figure 39. Therefore, one can use Eq. (4) to compute the contact normal\(^\dagger\) without any further checks

\[
\vec{n}_{AB} = \frac{(\vec{n}_A - \vec{n}_B)}{||\vec{n}_A - \vec{n}_B||} \tag{89}
\]

and the resulting \( \vec{n}_{AB} \) will be “reasonably” directed “from” \( A \) towards \( B \). In other words, we use the above expression by implicitly assuming that:

• \( \vec{n}_A \) points “reasonably” from \( A \) towards \( B \);  
• \( \vec{n}_B \) points “reasonably” from \( B \) towards \( A \).

By using the definition of the *centers-joining* (unit) vector \( \vec{c}_{AB} \) (see Eq. 1):

\[
\vec{c}_{AB} = \frac{\vec{C}_B - \vec{C}_A}{||\vec{C}_B - \vec{C}_A||} \tag{90}
\]

the above conditions can be expressed mathematically as:

\[
\vec{n}_A \cdot \vec{c}_{AB} > 0 \quad \vec{n}_B \cdot \vec{c}_{AB} < 0 \tag{91}
\]

\(^\dagger\) Note that in (89) we use the *normalized* pinball ASNs \( \hat{n}_A, \hat{n}_B \) instead of the non-normalized ASNs \( n_A, n_B \) of (4).
Instead, for a non-continuum element pinball, namely a shell in 2D or 3D, or a bar/beam in 2D‡, the associated ASN is defined only apart from the sign, since a shell without a topological thickness has two (mutually opposite) faces and, by convention, the normal to the first face is retained. In fact, a shell pinball can often be penetrated by two pinballs, laying on opposite sides of the shell, as shown in the example of Figure 40.

Thus, for a shell pinball $A$, the ASN to be considered in the calculation of the contact normal is either $\hat{n}_A$ or $-\hat{n}_A$. Both values are equally valid and the “right” one to choose depends on the position of the contacting (second) pinball $B$ with respect to the current $A$ pinball. In the example of Figure 40 we consider for generality contact of the shell ($A$) with another shell ($B$) on one side, and with a continuum ($C$) on the other side. In order to compute the first contact normal $\hat{n}_{AB}$ we proceed as follows:

- Choose the values of $\hat{n}_A$ and $\hat{n}_B$ with the “right” sign for this contact, which are indicated by $\hat{n}^*_A$ and $\hat{n}^*_B$

  \[
  \begin{align*}
  &\text{If } \hat{n}_A \cdot \hat{c}_{AB} > 0 \quad \text{then } \hat{n}^*_A = \hat{n}_A \quad \text{else } \hat{n}^*_A = -\hat{n}_A \quad (92) \\
  &\text{If } \hat{n}_B \cdot \hat{c}_{AB} < 0 \quad \text{then } \hat{n}^*_B = \hat{n}_B \quad \text{else } \hat{n}^*_B = -\hat{n}_{AB}
  \end{align*}
  \]

- Compute the contact normal by using the “star” version of the ASNs

  \[
  \hat{n}_{AB} = \frac{\hat{n}^*_A - \hat{n}^*_B}{\|\hat{n}^*_A - \hat{n}^*_B\|} \quad (93)
  \]

For the second contact, occurring between the shell pinball $A$ and the continuum pinball $C$, exactly the same procedure can be used, by noting that for continuum pinballs the “star” ASN always coincides with the ASN, irrespective of the position of the second (contacting) pinball:

  \[
  \hat{n}^*_C = \hat{n}_C \quad (94)
  \]

The resulting contact normals $\hat{n}_{AB}$ and $\hat{n}_{AC}$ should therefore be directed as shown in Figure 40.

‡: Recall that for a bar/beam in 3D the ASN is undefined (set to 0.0).
Figure 33 - A shell/beam/bar element in 2D (SEG2 element shape).

Figure 34 - ASNs for shell/beam/bar elements in 2D (SEG2 element shape).
Parent pinballs are all $E$ pinballs

$L = 0$
Element ASNs are zero
Nodal ASNs are zero

Descendent pinballs are either $V$ or $C$ pinballs

$L > 0$
$V$ pinballs are near the edges without adjacents and have zero ASNs
$C$ pinballs also have zero ASN

---

**Figure 35 - ASNs for beam/bar elements in 3D (SEG2 element shape).**

Parent pinballs are all $E$ pinballs

$L = 0$
Element ASNs are normal to 1st face of the element
Nodal ASNs are zero

Descendent pinballs are $F$, $V$ or $C$ pinballs

$L > 0$
$F$ pinballs are either internal or near an edge with adjacents. They have the same ASN as the parent
$C$ pinballs are near the edges without adjacents and have zero ASNs
$V$ pinballs are near the vertices and have zero ASNs

---

**Figure 36 - ASNs for a stand-alone shell element in 3D (TRI3 element shape).**
**Figure 37 - ASNs for a stand-alone shell element in 3D (QUA4 element shape).**

- Parent pinballs are all \( E \) pinballs
- \( L = 0 \) Element ASNs are normal to 1st face of the element
- Nodal ASNs are zero
- \( L > 0 \) Descendent pinballs are \( F, V \) or \( C \) pinballs
- \( F \) pinballs are either internal or near an edge with adjacents. They have the same ASN as the parent
- \( C \) pinballs are near the edges without adjacents and have zero ASNs
- \( V \) pinballs are near the vertices and have zero ASNs

**Figure 38 - ASNs for two shell element in 3D sharing a common edge.**

- \( L > 0 \) \( C \) pinballs near the shared edge become \( F \) pinballs
Figure 39 - Contact normal between pinballs associated with continuum elements.

\[
\hat{n}_{AB} = \frac{\hat{n}_A - \hat{n}_B}{\|\hat{n}_A - \hat{n}_B\|}
\]

Figure 40 - Contact normal between pinballs associated with shell elements (in 2D).

\[
\hat{n}_{AB} = \frac{\hat{n}_A^* - \hat{n}_B^*}{\|\hat{n}_A^* - \hat{n}_B^*\|}
\]

\[
\hat{n}_{AC} = \frac{\hat{n}_A^* - \hat{n}_C^*}{\|\hat{n}_A^* - \hat{n}_C^*\|}
\]
4. Implementation details

Some details of the implementation are presented hereafter for completeness.

In the module M_PINBALL, in the derived type TYPE PINBALL (which represents parent pinballs) the following description of the ASN is added:

```fortran
TYPE PINBALL ! "PARENT" PINBALL DEFINITION
  INTEGER :: ELEMENT     ! ELEMENT INDEX
  REAL(8) :: RADIUS      ! RADIUS OF THE PINBALL
  REAL(8) :: CENTER(3)   ! CENTER OF THE PINBALL
  INTEGER :: MAXLEV      ! MAXIMUM LEVEL OF DESCENDANT PINBALLS
  INTEGER :: SET         ! SET TO WHICH PINBALL BELONGS
    (!<0 IF SELF-CONTACT ENABLED)
  INTEGER :: HARDNESS    ! ASSOCIATED "HARDNESS" (ONLY FOR FLAT CONTACT)
  REAL(8) :: DTPINB      ! LIMIT TIME STEP ON THIS PINBALL'S ELEMENT AS DICTATED BY PCONTACTS
    (!0 IF NONE)
  LOGICAL :: IS_ACTIVE   ! ACTIVE PINBALL OR NOT
  REAL(8) :: ASN(4)      ! ASN OF THE PINBALL:
    (!1)  = LENGTH OF THE ASN BEFORE NORMALIZATION (>= 0.0)
    (!2:) = COMPONENTS OF THE NORMALIZED ASN (IDIM VALUES) OF LENGTH 1.0
    (IF ASN(1)>0) OR 0.0 (IF ASN(1)=0)
END TYPE PINBALL
```

Similarly, in the derived type TYPE DESCENDENT_PINBALL (which represents descendent pinballs) we add the same quantity:

```fortran
TYPE DESCENDENT_PINBALL ! "DESCENDENT" PINBALL DEFINITION
  INTEGER :: ANCESTOR    ! INDEX OF ANCESTOR (0-LEVEL) PINBALL
  INTEGER :: LEVEL       ! LEVEL OF THIS PINBALL (>= 0)
  REAL(8), POINTER :: XYZ(:, :) ! COORDS OF "NODES" OF PINBALL
  REAL(8) :: RADIUS      ! RADIUS OF THE PINBALL
  REAL(8) :: CENTER(3)   ! CENTER OF THE PINBALL
  INTEGER, POINTER :: IFACE(:) ! FACE INDEXES
    ! N>0=EXTERNAL, ON FACE N OF PARENT
    ! M<0=INTERNAL, SEES PAR. NEIGHBOUR M
    ! 0=INTERNAL (FROM CUT OF PARENT)
    ! FOR BEAM/SHELL ELEMENTS:
    ! SPECIAL CONVENTION (SEE REPORT)
  REAL(8) :: ASN(4)      ! ASN OF THE DESCENDENT PINBALL:
    (!1)  = LENGTH OF THE ASN BEFORE NORMALIZATION (ALWAYS > 0.0)
    (!2:) = COMPONENTS OF THE NORMALIZED ASN (IDIM VALUES) OF LENGTH 1.0
END TYPE DESCENDENT_PINBALL
```

In the same module M_PINBALL, a new possible value for the PINBALL_FNOR variable is added: the variable is set to 3 when the ASN optional keyword is read (OPTI PINS ASN):

```fortran
INTEGER :: PINBALL_FNOR  ! 0 : PINBALL VELOCITY CONSTRAINT IS WRITTEN ALONG THE DIRECTION OF THE LINE THAT JOINS THE CENTERS (DEFAULT)
  ! 1 : PINBALL VELOCITY CONSTRAINT IS WRITTEN ALONG A "MEAN" OF THE TWO FACE NORMALS N = (NA - NB)
  ! (OPTI PINS FNOR)
  ! (THIS REQUIRES OPTI PINS FACE
```

55
In module M_PINBALLS_DATA we add a new array NODAL_ASN (:,:) to contain the assembled normals at nodes:

\[
\text{REAL}(8), \text{POINTER} :: \text{NODAL\_ASN}(::)
\]

* for use with opti pinb asn:
* nodal_asn(1,i) = norm of nodal asn at node i (>= 0.0)
* nodal_asn(2:,i) = components of nodal asn at node i (idim values)
* normalized to length 1.0 or 0.0

If needed, the ASNs for pinballs and at nodes are computed in the subroutine M_PINBALLS::UPDATE_PINBALLS. The penalty contact forces are computed in M_PINBALLS::PENA_CONTACT_FORCES.

The contact normal is computed in M_PINBALLS_SPLIT::CONTACT_DATA_PARENTS in the case of contact between parent (0-level) pinballs, or in M_PINBALLS_SPLIT::CONTACT_DATA in the case of contact between descendent pinballs.

The pinball types for each couple of contacting pinballs (TYPEI, TYPEJ) are added to the derived type TYPE PCONTACT in module M_PCONTACT:

\[
\text{TYPE PCONTACT}
\]

INTEGER :: PA, PB ! INDEXES OF THE TWO (PARENT) PINBALLS
REAL(8) :: NAB(3) ! UNIT NORMAL FROM A TO B
! "JOINS CENTERS" IF PINBALL\_FNOR = 0,
! "MEAN" NORMAL IF PINBALL\_FNOR = 1,
! "COMMON" NORMAL IF PINBALL\_FNOR = 2,
! "ASN" BUILT NORMAL IF PINBALL\_FNOR = 3

REAL(8), POINTER :: SHA(:), SHB(:)
! SHAPE FUNCTIONS OF ELEMENT A AND B AT
! INTERACTION POINTS A' AND B'
REAL(8) :: CSIA(3), CSIB(3) ! NORMALIZED COORS OF CONTACT POINTS
! WITH RESPECT TO PARENT ELEMENTS
TYPE(DESCENDENT_PINBALL) :: DPI, DPJ ! DESCENDENT PINBALLS IN
! CONTACT (WITH LEVEL=0 IF PARENT)
! THESE ARE COPIES (NOT POINTERS)
LOGICAL :: HAS\_COMMON\_NORMAL ! IS ASSOCIATED NORMAL "COMMON"?
INTEGER :: NODE1, NODE2 ! NODES OF THE CONTACT (EACH MAY BE 0)
! N1 N2  --> NN CONSTRAINT
! 0 0  --> PP CONSTRAINT
! N1 0 OR
! 0 N2  --> NP CONSTRAINT
INTEGER :: ELEM1, ELEM2 ! ELEMENTS OF THE CONTACT (EACH MAY BE 0)
! 0 0  --> NN CONSTRAINT
! E1 E2  --> PP CONSTRAINT
! 0 E2 OR
! E1 0  --> NP CONSTRAINT
LOGICAL :: FLAG ! GENERIC FLAG FOR VARIOUS OPERATIONS
REAL(8) :: PENETR ! PENETRATION (ALONG NAB) IF USEFUL
REAL(8) :: PDOT ! PENETRATION RATE (ALONG NAB) IF USEFUL

INTEGER :: TYPEI, TYPEJ ! TYPE OF CONTACTING (SUB-)PINBALL:
! 0 = ELEMENT (PARENT) PINBALL
! 1 = VERTEX SUB-PINBALL
! 2 = CORNER SUB-PINBALL (3D ONLY)
! 3 = FACE SUB-PINBALL

INTEGER :: NODEI(5), NODEJ(5) ! N. OF CONTACTING SUB-PINBALL NODES,
! THEN LIST OF THESE NODES (GLOBAL INDEXES):
! 1 NODE FOR A VERTEX SUB-PINBALL
! 2 NODES FOR A CORNER SUB-PINBALL (3D ONLY)
! 2, 3 OR 4 NODES FOR A FACE SUB-PINBALL

END TYPE PCONTACT

In addition, the nodes NODEI (:), NODEJ (: ) related to the contacting sub-pinballs are also added, see above list of the derived type. Note the comment defining NAB (:), where it is explicitly stated that the normal is oriented “from” A “to” B.

A flowchart of the relevant code parts is shown in Figure 41.

---

**Figure 41 - Flowchart of pinball contact calculations.**
4.1 Implemented version of the contact normal (penetration direction)

The algorithm to compute the contact normal (penetration direction) actually implemented in the code is slightly different from the one listed in Section 2.11.3, because of the particular code structure (see Figure 41) and is reported below.

Two pinballs $A$, $B$ penetrate each other if, according to (2):

$$\|C_{AB}\| < R_A + R_B.$$  \hfill (95)

With reference to the flowchart in Figure 41, this is checked on the parent pinballs in the subroutine M_PINBALLS::PINBALL_PENETRATION. When this condition is satisfied, we distinguish between two cases:

- If no hierarchic pinballs are foreseen in the calculation, so that both (parent) pinballs are already at the maximum level of refinement prescribed by the user, then a new pinball contact (PCONTACT) between parent pinballs is created by calling the subroutine M_PINBALLS_SPLIT::CONTACT_DATA_PARENTS.

- Else, one or both pinballs are further refined (recursively) and penetration is then checked on their descendents, see M_PINBALLS_SPLIT::DESCEND_PINBALLS, until either no penetration is detected any more (so that the search terminates without generating any contact for the current couple of pinballs), or we arrive at the maximum level for both pinballs, and in the latter case a new pinball contact (PCONTACT) is created by calling the subroutine M_PINBALLS_SPLIT::CONTACT_DATA.

4.1.1 Contact normal between two parent pinballs

In the first case, both contacting pinballs are parent ($E$) pinballs. For these pinballs the ASN may or may not be defined. The penetration direction $\hat{n}_{AB}$ is computed as follows in the subroutine M_PINBALLS_SPLIT::CONTACT_DATA_PARENTS. In principle one would have, from the algorithm of Section 2.11.3:

- If $\|n_A\| = \|n_B\| = 0$, i.e. if both pinballs have an undefined associated normal, then the penetration direction is the line joining the two pinball centres (see Figure 29):

$$\hat{n}_{AB} = C_{AB}/\|C_{AB}\|.$$  \hfill (96)

- Else, if ( $\|n_A\| = 1$ and $\|n_B\| = 0$ ), or if ( $\|n_A\| = 0$ and $\|n_B\| = 1$ ), i.e. if only one of the two pinballs has a defined normal, then the penetration direction coincides with either $-\hat{n}_A$ or $\hat{n}_B$, respectively, see Figure 30. This can be treated either as a special case, or by applying the general equation (4), since the result is the same.
• Else $\|\vec{n}_A\| = 1$ and $\|\vec{n}_B\| = 1$, i.e. both pinballs have a defined (unit) normal. Then in principle the types of the two pinballs should be considered. However, in this case we know that both pinballs are $E$ pinballs. Then the penetration direction is computed with the general expression (4):

$$\vec{n}_{AB} = (\vec{n}_A - \vec{n}_B)/\|\vec{n}_A - \vec{n}_B\| \quad (97)$$

In practice, however, we always compute the penetration direction by the general expression (4) or (97) first. This gives the correct result in the last two cases above, while in the first one the result is undetermined because the denominator of (4) is zero. If the computed denominator is below a small tolerance, we use expression (96) to evaluate the penetration direction. This may occur either in the first case listed above (both ASNs are zero) or also with non-zero ASNs in some limit, degenerated case, where the two contacting pinballs have the same or almost the same ASN. In all these cases using (96) provides a better (probably the best possible) guess for the penetration direction.

4.1.2 Contact normal between two pinballs of which at least one is a descendent

In the other case, at least one of the two contacting pinballs is a descendent ($F$, $C$ or $V$) pinball. Then, the contact normal (penetration direction) $\vec{n}_{AB}$ could in principle be computed in the subroutine M_PINBALLS_SPLIT::CONTACT_DATA by following the algorithm sketched previously in Section 2.11.3.

However, that algorithm is relatively complex and further complication would be added by the fact that, in general, only face ($F$) pinballs are guaranteed to have a non-zero associated ASN, while for all other types of pinballs (element $E$, vertex $V$ and corner $C$) the ASN may be zero in some cases, detailed in Section 4.2.6 below. For $E$ pinballs, this happens when the associated element is either a material point, or a 3D beam/bar, or a continuum with insufficient discretization (see pathological case in Figure 10). For $V$- and $C$- pinballs, the ASN is zero in particular cases which are explained in detail in Section 4.2 below, where the calculation of the ASN for descendent pinballs is described.

However, by assigning numerical codes 0 to 3 in increasing level of priority to the pinball types $E$, $V$, $C$, $F$, respectively (as indicated in the description of the TYPEI and TYPEJ variables in TYPEPCONTACT at the beginning of this Section), the procedure can be simplified and is actually implemented as follows:

• If TYPEI is equal to TYPEJ†, then the general expression (4) or (97) is used to compute the contact normal, except in the case of a $C$-$C$ contact (TYPEI=TYPEJ=2), i.e. contact between two corner pinballs in 3D, in which a more accurate expression using the line joining the closest points on the two segments is adopted (see Section 3.2.1 and Appendix A).

†: Note that at this point of the procedure this can occur only if both pinballs are descendent pinballs, since here we can have at most one element (parent) pinball.
• Else the two types are different. The type with greater numerical index “has the precedence” over the other one and the contact normal is constructed by using only the ASN of the pinball with the greater type: if \( \text{TYPE}_I > \text{TYPE}_J \) then \( \hat{n}_{AB} = \hat{n}_A \), else \( \hat{n}_{AB} = -\hat{n}_B \).

• In any case we check whether the computed penetration vector (before normalization) is physically meaningful. If its length is below a small tolerance, we use expression (96) instead, i.e. the centers-joining line, to evaluate the contact normal.

• If contact occurs between a corner (\( C \)) pinball and a lower-priority (\( E \) or \( V \)) pinball, so that the corner ASN is taken due to the precedence rule, but this normal is zero, then the centers-joining line assumed as contact normal is corrected by projecting it onto the plane normal to the corner and then re-normalizing it. This makes sure that the final normal is perpendicular to the corner.

It should be noted that the “star” version of the pinball ASNs \( \hat{n}_A^*, \hat{n}_B^* \), i.e. choosing the “right” sign for the ASN of non-continuum pinballs, introduced in Section 3.3 and resulting from Eqs. (92) are preliminary computed and actually used in the above described procedure, in place of \( (\hat{n}_A, \hat{n}_B) \), in order to cope with the case of shell/beam/bar elements, for which the normal is defined only apart from the sign.

It is easily verified that this procedure gives the same result in the cases treated by the algorithm of Section 2.11.3, while being more general.

The precedence rules are summarized in the following Table. In abscissa is represented the type of the first pinball (both as a mnemonic letter and as a numeric code, see the \text{TYPE PCONTACT} at the beginning of this Section) and in ordinate the type of the second pinball. Obviously, the table is symmetric with respect to the diagonal.

\[
\begin{array}{cccc}
0 & 1 & 2 & 3 \\
E & - & V & C & F \\
V & - & C & F \\
C & C & - & F \\
F & F & F & - \\
\end{array}
\]

\textbf{Table 3 - Precedence rules in the contact between two pinballs.}
4.2 Calculation of the ASN for a descendent pinball

We now detail the calculation of the ASN for a descendent pinball, which is performed in subroutine M_PINBALLS_SPLIT::COMPUTE_DESC_ASN (see the flowchart in Figure 41). The routine also fills up the TYPEI/TYPEJ variable and the NODEI(:)/NODEJ(:) array for the PCONTACT, see beginning of this Section. Let DP be the descendent pinball and let IEL be the element to which it is associated (ancestor element of DP). The routine is only called for proper descendents (level $L > 0$), since for parent pinballs the ASN has been already computed (at the beginning of each time step) in another routine (M_PINBALLS::UPDATE_PINBALLS).

We distinguish several cases, according to the “class” of the element IEL to which DP is associated: material point, continuum element, shell element or beam/bar element.

4.2.1 The IFACE array

Before describing the calculation of ASNs for descendent pinballs, we briefly recall the definition of the IFACE(:) array from reference [13] (Figure 215 on page 269 and Sections 13.2.1 and 15.1 of that report), because this quantity is used in the following algorithm for the ASN.

The basic purpose of the IFACE(:) array is to provide information in order to “localize” a descendent pinball (of any level $L > 0$) with respect to its parent (0-level) pinball. The length of the IFACE(:) array is the number of faces of the element associated with the parent pinball. Since descendent pinballs are built from element portions of the same shape as the parent element (except in the case of the pyramid, which is somewhat special), this is also the number of faces of the element portion associated with the descendent. Furthermore, the orientation in space of descendent element portions is the same as that of the (parent) element.

Thus, we see from the data structure in Section 4 that the IFACE(:) array is a field of the derived type DESCENDENT_PINBALL and, as such, it is also available in the derived type PCONTACT through the DPI and DPJ fields of that type (which are of type DESCENDENT_PINBALL).

However, one should note that, despite its name, the type DESCENDENT_PINBALL can represent either a true descendent pinball or a parent pinball. In fact, the LEVEL of the DESCENDENT_PINBALL can be 0, as one can see from the definition of this type at the beginning of Section 4. While a true descendent pinball ($L > 0$) has always associated a (non-empty) IFACE table, a (fake) descendent pinball with $L = 0$ may or may not have associated a non-empty IFACE table (and in the latter case the table is not allocated, so it should not be used!) The table is allocated and filled up in complete analogy with (proper) descendent pinballs if the 0-level pinball is in contact (in the current PCONTACT being considered) with another pinball of level $L > 0$, i.e. with a proper
descendent pinball. Instead, if the second pinball involved in the PCONTACT is (also) of level \( L = 0 \), then the IFACE table is not allocated.

All this may be quite confusing. The confusion would be largely avoided if one would have used the name CONTACTING_PINBALL instead of DESCENDENT_PINBALL for the derived type. Then it would be clear that a CONTACTING_PINBALL can be at any level (including \( L = 0 \)). It would also be more natural for the derived type PCONTACT to contain two CONTACTING_PINBALLs (DPI and DPJ), instead of two DESCENDENT_PINBALLs.

From reference [13], the values contained in the IFACE table are as follows, see Figure 42.

- For (pinballs associated with) a \text{continuum} element: \(\text{IFACE}(I) = n \ (> 0)\) if the \( I \)-th face of the element portion associated with the pinball is external to the mesh and lies on face \( n \) of the parent element. \(\text{IFACE}(I) = m \ (< 0)\) if the \( I \)-th face of the element portion associated with the pinball is internal to the mesh and has (parent) element \(-m\) as a neighbour. \(\text{IFACE}(I) = 0\) if the \( I \)-th face of the element portion associated with the pinball is internal to the mesh and is derived from a cutting operation of the parent element.

- For \text{material point} elements, the IFACE table (when it is allocated) contains only one entry (because the PMAT elements have only one face), of fixed content by convention \(\text{IFACE}(1) = 1\).

- For the other types of elements (shells, beams, bars), the IFACE table (when it is allocated) contains two entries (because all these elements have two faces in EPX, including bar elements in 3D). The contents of the IFACE table (when it is allocated) is fixed to \((2,1)\) for parent pinballs, is fixed to \((1,0)\) for proper descendent pinballs lying on a face of the element (\(F\)-pinballs), is \((i, i)\) for proper descendent pinballs lying on the \(i\)-th (local) \text{vertex} of the element (\(V\)-pinballs) and is \((i, j)\), with \( j = \text{next}(i) \), for proper descendent pinballs lying on the \(i\)-th (local) \text{corner} of the element (\(C\)-pinballs).

Note that the IFACE code values described above have been chosen in such a way to be unique. In other words, from the contents of the IFACE table one can deduce the type of the associated element. The following examples should help clarify the definition of IFACE.

\text{ASNC01}

This example is shown in Figure 43. Two bodies, each composed of two 2D continuum elements, come into contact. Zero-level pinballs are employed. Two PCONTACTs are detected. Since each PCONTACT involves only parent (0-level) pinballs the IFACE tables are not allocated.

\[\text{‡. Here } i \text{ varies from } 1 \text{ to the number of nodes of the element (} n \text{), while } j = \text{next}(i) \text{ means that } j = i + 1 \text{ for } i = 1, \ldots, n - 1 \text{ and } j = 1 \text{ for } i = n.\]
**ASNC02**

This example is shown in Figure 44. It is similar to case ASNC01 but (proper descendent) pinballs at MLEV 1 are used for the contact instead of parent descendents. There are four PCONTACTs. The involved pinballs are proper descendents of continuum type, therefore the IFACE tables are generated, as shown in the right part of the Figure.

For example, the first PCONTACT involves the second from left (for local numbering reasons) couple of pinballs in the Figure. The DPI%IFACE table (referring to the lower pinball) contains the sequence (0, -2, 3, 0). This means that the first and last face of the element portion (for which IFACE is 0) are internal to the parent element and result from an ideal cutting operation of the parent element. This is correct, since these two faces are the lower and left ones of the element portion.

The value -2 on the second face (right face) means that this is an internal face of the mesh and that element 2 is the neighbour on the other side of the face. Finally, the value 3 on the third (upper) face means that this is an external face (portion) of the mesh laying on face 3 of the parent element. Similarly, the DPJ%IFACE (referring to the upper pinball) contains (1, -4, 0, 0): the first face (lower) is external, laying on face 1 of the parent element, the second face (right) is an internal face of the mesh, contiguous to element 4 of the mesh, and the last two faces (upper and left) are internal to the parent element and result from an ideal cutting operation of the parent element.

**ASNC03**

This example is similar to case ASNC02 but the level of pinballs used for the contact is MLEV 2, see Figure 45. There are 8 contacts detected and the construction of the IFACE tables is similar to the previous example.

**ASNC04**

This example is similar to case ASNC01 (parent pinballs used for the contact), but the mesh is more extended, see Figure 45. There are 2 contacts detected and the IFACE tables are not generated since, like in case ASNC01, the contacts involve only zero-level continuum pinballs.

**ASNC05**

This example uses bar elements in 2D (BARR), see Figure 46. Zero-level (parent) pinballs are used for the contact. Since only one body is defined in this example, the OPTI PINS VIDE option is activated in order to achieve (fake) contact between each pinball and a fictitious 0-pinball that encompasses the whole model. Therefore, five (fake) contacts are detected, between each one of the five bar elements and the 0-pinball. However, since all pinballs involved (including the 0-pinball) are zero-level (parent) pinballs, the IFACE tables are not generated.
ASNC13
This example is shown in Figure 48. It is similar to case ASNC03 but only the lower body uses MLEV 2, while the upper body uses MLEV 0 like in case ASNC01. The scope is to check what happens when pinballs at different levels come into contact. Eight contact are detected, one for each (proper) descendent pinball in the lower body. Each contact involves a pinball of level 2 and a pinball of level 0 (parent). In this case, since (at least) one of the two involved pinballs has $L > 0$, the IFACE tables are generated, as shown in the right part of the Figure. The DPI%IFACE are associated with level-2 pinballs and are therefore identical to those of case ASNC03. The DPJ%IFACE are associated with 0-level pinballs and are built like if the pinball would be at a higher level. Of course, in this case, none of the DPJ%IFACE values can be 0 since the (parent) element is not split.

ASNC06
This example is similar to ASNC05 but uses MLEV 1 pinballs (and the VIDE option), see Figure 49. Ten contacts are generated, each one involving a level-2 bar pinball and the 0-level (fake) pinball. Therefore, IFACE tables are generated. For the level-2 bar pinballs, the IFACE tables contain $(i, i)$ (with either $i = 1$ or $i = 2$) since all these are $V$-pinballs. Also the 0-pinball (fake) receives an IFACE table in this case, containing just one entry: DPJ%IFACE(1) = 0.

ASNC16
This test is similar to case ASNC06 but uses MLEV 2 pinballs in the bars, see Figure 50. The result in terms of IFACE tables is similar to that of case ASNC06, but we see appear some “face” pinballs (in the sense of IFACE, not of the ASN), which are characterized by the (fixed) code IFACE = 1, 0.

PMAT00
This example checks material points at level 0 and is illustrated in Figure 51. One contact between 0-level material point pinballs is detected. Since both involved pinballs are 0-level, the IFACE tables are not generated.

PMAT11
This example checks contact between a material point at level 0 and a bar (also at level 0) and is illustrated in Figure 52. One contact between 0-level pinballs is detected. Since both involved pinballs are 0-level, the IFACE tables are not generated.

PMAT12
This example is similar to case PMAT11 but level 1 is used for the bar. It is illustrated in Figure 53. Two contacts between a 1-level pinball (in the bar) and a 0-level pinball (in the material point) are
detected. The IFACE tables are generated. In this case, the table associated with the material point pinball is $\text{IFACE}(1)=1$, in accordance with Figure 42.

**PMAT13**

This example is similar to case PMAT11 but level 2 is used for the bar. It is illustrated in Figure 53. Four contacts between a 2-level pinball (in the bar) and a 0-level pinball (in the material point) are detected. The IFACE tables are generated similarly to case PMAT12 but, in addition, some “face” pinballs (in the sense of IFACE, not of the ASN) appear, which are characterized by the (fixed) code $\text{IFACE}=1,0$.

One may observe that material point pinballs (which are always at level 0) receive a (fixed) code $\text{IFACE}(1)=1$, if they are involved in a contact with another pinball of level $L > 0$.

### 4.2.2 Material point

After the description of the IFACE data structure, we return to the procedure for computing the ASNs in descendent pinballs. For a material point no descendent pinballs are admitted, so the routine COMPUTE_DESC_ASN should not be called. In any way, the ASN to a material point is always zero so it need not be computed.

### 4.2.3 Continuum element

For a continuum element, we consider the external (to the mesh) faces of the element $\text{IEL}$ “touched” (so to say) by $\text{DP}$, i.e. the external faces to which $\text{DP}$ is adjacent. Recall that descendent pinballs are built only along the (external) faces of continuum elements and not in the interior of such elements. This information can be readily extracted from the IFACE(:) array (see derived type DESCENDENT_PINBALL at the beginning of this Section), which is filled up elsewhere. Only the strictly positive values, indicating an external face to the mesh, are retained, because (for continuum elements) 0 values indicate an internal sub-face to the element and negative values an internal face to the mesh. The complete set of definitions of the IFACE array is shown in Figure 42, which is taken from reference [13] (Figure 215 on page 269 of that report) and edited to update the terminology: the “corner” ($C$) and “side” ($S$) proper descendent pinballs of reference [13] correspond respectively to vertex ($V$) and corner ($C$) pinballs in the present report, while face ($F$) pinballs are named the same in both reports. The parent $C$-pinballs of reference [13] correspond to “element” ($E$) pinballs here.

Then:

- If $\text{DP}$ touches only one external face of $\text{IEL}$, then $\text{DP}$ is a face ($F$) pinball, and its ASN coincides with the normal to the face concerned. The associated nodes list are simply the nodes of the face.
- If $\text{DP}$ touches two external faces of $\text{IEL}$, then we consider two sub-cases.
– If the space dimension is two, then $\mathcal{D}_P$ is a vertex ($V$) pinball, and its ASN coincides with the normal to the node concerned (the common node to the two faces). This normal is contained in the $\text{NODAL\_ASN(\cdot)}$ array for the node concerned. The associated nodes list contains only this node.

– Else the space dimension is three, $\mathcal{D}_P$ is a corner ($C$) pinball, the node list contains the two nodes common to both faces, and the ASN is the average of the normals to the concerned vertices ($\text{NODAL\_ASN(\cdot)}$), projected onto the plane normal to the corner.

• Finally, if $\mathcal{D}_P$ touches three external faces of $\text{IEL}$ (which can occur only in 3D), then $\mathcal{D}_P$ is a vertex ($V$) pinball, associated with the common node to the three faces. The nodes list contains only this node and the associated ASN is the corresponding nodal normal in $\text{NODAL\_ASN(\cdot)}$.

4.2.4 Shell element

If the element is a shell, then we distinguish two cases according to the space dimension.

The $\text{IFACE(\cdot)}$ array for a 2D shell contains the following information:

* (for the parent pinball (not used here): $\text{iface(\cdot)} = (2,1)$)
* for proper descendent pinballs: for vertex $i$, $\text{iface(\cdot)} = (i,i)$
* for a face pinball, $\text{iface(\cdot)} = (1,0)$

• If $\text{IFACE(2)}$ is zero, the descendent is a face ($F$) pinball having as ASN the same ASN as its ancestor (0-level) pinball and as associated node list the two nodes of the face.

• Else:
  – If the descendent is near an edge of the current shell element with no associated adjacent elements, then it is a vertex ($V$) pinball with associated zero ASN and with associated node list the node of the concerned edge.
  – Otherwise it is a face ($F$) pinball having as ASN the same ASN as its ancestor (0-level) pinball and as associated node list the two nodes of the face.

The $\text{IFACE(\cdot)}$ array for a 3D shell contains the following information:

* (for the parent pinball (not used here): $\text{iface(\cdot)} = (2,1)$)
* for proper descendent pinballs: for vertex $i$, $\text{iface(\cdot)} = (i,i)$
* for a face pinball, $\text{iface(\cdot)} = (1,0)$
* for edge (corner) of local nodes $j$-$k$, $\text{iface(\cdot)} = (j,k)$ with $k = \text{next}(j)$

• If $\text{IFACE(2)}$ is zero, the descendent is a face ($F$) pinball having as ASN the same ASN as its ancestor (0-level) pinball, i.e. the normal to the face, and as associated node list the three or four nodes of the face.

• Else:
– If the descendent is near a node of the current shell element, then we consider the two edges of the shell element that share this node: if both edges have no adjacent shell element, then the descendent is a vertex \((V)\) pinball with associated zero ASN and as associated node list the node concerned; else if one edge has adjacents and the other has no adjacents, then the descendent is a corner \((C)\) pinball with zero ASN and with associated nodes list the two nodes of the edge with no adjacents; else both edges have adjacents and the descendent is a face \((F)\) pinball having as ASN the same ASN as its ancestor (0-level) pinball, i.e. the normal to the face, and as associated node list the three or four nodes of the face.

– Else the descendent is near an edge of the current shell element and the element has no adjacents along this edge, then the descendent is a corner \((C)\) pinball with zero ASN and with associated nodes list the two nodes of the edge.

– Otherwise, the descendent is near an edge with adjacents, and then it is a face \((F)\) pinball having as ASN the same ASN as its ancestor (0-level) pinball, and as nodes list the three or four nodes of the face.

4.2.5 Beam/bar element

Finally, let us consider the case of a beam/bar element.

The IFACE(\(:\)) array for a 2D or 3D beam/bar is similar to that for a 2D shell and contains the following information:

* (for the parent pinball (not used here): \(\text{iface}(:) = (2,1)\))
* for proper descendent pinballs: for vertex \(i\), \(\text{iface}(:) = (i,i)\)
* for a face pinball, \(\text{iface}(:) = (1,0)\)

- If IFACE(2) is zero, then we distinguish two cases depending upon space dimension:
  - In 2D the descendent is a face \((F)\) pinball having as ASN the same ASN as its ancestor (0-level) pinball and as associated node list the two nodes of the face.
  - In 3D the descendent is a corner \((C)\) pinball having zero ASN and as associated node list the two nodes of the face.

- Else:
  - If the descendent is near an edge (node) of the current beam/bar element with no associated adjacent elements, then it is a vertex \((V)\) pinball with associated zero ASN and with associated node list the node of the concerned edge.
  - Otherwise the descendent is near an edge (node) with adjacents: then, if the space dimension is two (2D), the descendent is a face \((F)\) pinball having as ASN the same ASN as its ancestor (0-level) pinball and as associated node list the two nodes of the face.
– Else the space dimension is three (3D), and the descendent is a corner (C) pinball having zero
ASN and as associated node list the two nodes of the face.

4.2.6 Summary of cases in which the ASN of a pinball is zero

It is useful to summarize the cases in which the ASN of a pinball is zero, because this has an influence on the algorithm used to compute the contact normal between two pinballs:

• A parent element (E) pinball has zero ASN in the following cases:
  – If the element is a material point.
  – If the element is a 3D beam/bar.
  – If the element is a continuum with insufficient discretization, see e.g. the pathological example in Figure 10.

• A vertex (V) pinball has zero ASN has zero ASN in the following cases:
  – If the element is a 2D/3D shell/beam/bar and the pinball is near a node without adjacents.

• A corner (C) pinball has zero ASN has zero ASN in the following cases:
  – If the element is a 3D beam/bar.
  – If the element is a 3D shell and the pinball is in certain particular positions (see Section 4.2.4).

• A face (F) pinball has always a non-zero ASN.
Figure 42 - Definition of the IFACE(:) array for the various pinball types (edited from [13]).

Note: the F pinball for the 2D continuum case was erroneously indicated as a C pinball in the first edition [18].
Figure 43 - Contents of the IFACE(:) array in test case ASNC01.

<table>
<thead>
<tr>
<th>Assembled model</th>
<th>Lower body</th>
<th>Upper body</th>
<th>Pcontacts</th>
</tr>
</thead>
</table>

Figure 44 - Contents of the IFACE(:) array in test case ASNC02.
**Figure 45 - Contents of the IFACE(:) array in test case ASNC03.**

<table>
<thead>
<tr>
<th>pcontact #</th>
<th>IP</th>
<th>JP</th>
<th>DPI%IFACE</th>
<th>DPJ%IFACE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>0, -2, 3, 0</td>
<td>1, -4, 0, 0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>0, 0, 3, 0</td>
<td>1, 0, 0, 0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>3</td>
<td>0, 0, 3, 0</td>
<td>1, 0, 0, 0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>3</td>
<td>0, 0, 3, 4</td>
<td>1, 0, 0, 0</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>4</td>
<td>0, 2, 3, 0</td>
<td>1, 2, 0, 0</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>4</td>
<td>0, 0, 3, 0</td>
<td>1, 0, 0, 0</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>4</td>
<td>0, 0, 3, 0</td>
<td>1, 0, 0, 0</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>4</td>
<td>0, 0, 3, -1</td>
<td>1, 0, 0, -3</td>
</tr>
</tbody>
</table>

**Figure 46 - Contents of the IFACE(:) array in test case ASNC04.**

**Figure 47 - Contents of the IFACE(:) array in test case ASNC05.**

<table>
<thead>
<tr>
<th>pcontact #</th>
<th>IP</th>
<th>JP</th>
<th>DPI%IFACE</th>
<th>DPJ%IFACE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>0</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Figure 48 - Contents of the IFACE(:) array in test case ASNC13.

<table>
<thead>
<tr>
<th>pcontact</th>
<th>IP</th>
<th>JP</th>
<th>DPI%IFACE</th>
<th>DPJ%IFACE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>0, 2, 3, 0</td>
<td>1, -4, 3, 4</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>0, 0, 3, 0</td>
<td>1, -4, 3, 4</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>3</td>
<td>0, 0, 3, 4</td>
<td>1, -4, 3, 4</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>3</td>
<td>0, 0, 3, 4</td>
<td>1, -4, 3, 4</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>4</td>
<td>0, 2, 3, 0</td>
<td>1, 2, 3, -3</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>4</td>
<td>0, 0, 3, 0</td>
<td>1, 2, 3, -3</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>4</td>
<td>0, 0, 3, 0</td>
<td>1, 2, 3, -3</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>4</td>
<td>0, 0, 3, -1</td>
<td>1, 2, 3, -3</td>
</tr>
</tbody>
</table>

Figure 49 - Contents of the IFACE(:) array in test case ASNC06.

<table>
<thead>
<tr>
<th>pcontact</th>
<th>IP</th>
<th>JP</th>
<th>DPI%IFACE</th>
<th>DPJ%IFACE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1, 1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>2, 2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>0</td>
<td>1, 1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>0</td>
<td>2, 2</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>0</td>
<td>1, 1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>0</td>
<td>2, 2</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>0</td>
<td>1, 1</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>0</td>
<td>2, 2</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td>0</td>
<td>1, 1</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>0</td>
<td>2, 2</td>
<td>0</td>
</tr>
</tbody>
</table>
Figure 50 - Contents of the IFACE(:) array in test case ASNC16.

<table>
<thead>
<tr>
<th>pcontact #</th>
<th>IP</th>
<th>JP</th>
<th>DP%IFACE</th>
<th>DPJ%IFACE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1,1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1,0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1,0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>2,2</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>0</td>
<td>1,1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>0</td>
<td>1,0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 51 - Contents of the IFACE(:) array in test case PMAT00.
Figure 52 - Contents of the IFACE(;) array in test case PMAT11.

<table>
<thead>
<tr>
<th>pcontact #</th>
<th>IP</th>
<th>JP</th>
<th>DPI%IFACE</th>
<th>DPJ%IFACE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1, 1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2, 2</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 53 - Contents of the IFACE(;) array in test case PMAT12.
5. Numerical examples

The algorithms described in the previous Section are illustrated by means of numerical examples. All the input files of the examples proposed are listed in the Appendix.

5.1 Visualization of the ASNs

A first set of tests shows the resulting nodal ASNs, the parent pinball ASNs and the descendent pinball ASNS (where appropriate) in various cases. They are listed in Table 4 and described hereafter.

<table>
<thead>
<tr>
<th>Name</th>
<th>Mesh</th>
<th>Contact parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VIDE01</td>
<td>1 CAR1</td>
<td>MLEV 2</td>
<td>Single-element continuum in 2D</td>
</tr>
<tr>
<td>VIDE02</td>
<td>1 TRIA</td>
<td>MLEV 2</td>
<td>Single-element continuum in 2D</td>
</tr>
<tr>
<td>VIDE05</td>
<td>1 CUBE</td>
<td>MLEV 2</td>
<td>Single-element continuum in 3D</td>
</tr>
<tr>
<td>VIDE07</td>
<td>2 CAR1</td>
<td>MLEV 2</td>
<td>Two-element continuum in 2D</td>
</tr>
<tr>
<td>ASNO01</td>
<td>3 CAR1 2 TRIA</td>
<td>MLEV 0</td>
<td>Heterogeneous continuum mesh in 2D</td>
</tr>
<tr>
<td>ASNO02</td>
<td>4 CUB8</td>
<td>MLEV 2</td>
<td>Bar impact in 3D</td>
</tr>
<tr>
<td>ASNO03</td>
<td>400 CAR1</td>
<td>MLEV 0</td>
<td>Impacting bodies in 2D</td>
</tr>
<tr>
<td>ASNO04</td>
<td>360 CAR1</td>
<td>MLEV 0</td>
<td>Hollow body in 2D</td>
</tr>
<tr>
<td>FUN201</td>
<td>5 FUN2</td>
<td>MLEV 0</td>
<td>Bar in 2D</td>
</tr>
<tr>
<td>FUN202</td>
<td>5 FUN2</td>
<td>MLEV 1</td>
<td>Bar in 2D</td>
</tr>
<tr>
<td>FUN203</td>
<td>5 FUN2</td>
<td>MLEV 2</td>
<td>Bar in 2D</td>
</tr>
<tr>
<td>FUN301</td>
<td>5 FUN3</td>
<td>MLEV 0</td>
<td>Bar in 3D</td>
</tr>
<tr>
<td>FUN302</td>
<td>5 FUN3</td>
<td>MLEV 1</td>
<td>Bar in 3D</td>
</tr>
<tr>
<td>FUN303</td>
<td>5 FUN3</td>
<td>MLEV 2</td>
<td>Bar in 3D</td>
</tr>
</tbody>
</table>

Table 4 - Tests to show the nodal, parent pinball and descendent pinball ASNs.

VIDE01

This test shows the nodal ASNs and the (parent) pinball ASNs for a stand-alone 2D quadrilateral (CAR1). A hierarchic pinball of level 2 (MLEV 2) is embedded in the element and the option OPTI PINS VIDE is activated in order to consider all descendents in contact and to be able to visualize the final descendents and their ASNs.

The left part of Figure 54 shows the parent pinball (PASN) and the nodal ASNs (NASN). The (parent) ASN (PASN) is zero as expected in this case because this is a stand-alone element. The right part of the Figure shows the final (level 2) descendents and the associated ASNs (DASN).
This test shows the nodal ASNs and the (parent) pinball ASNs for a stand-alone 2D triangle (TRIA). A hierarchic pinball of level 2 (MLEV 2) is embedded in the element and the option OPTI PINS VIDE is activated in order to consider all descendents in contact and to be able to visualize the final descendents and their ASNs.

The left part of Figure 54 shows the parent pinball and the nodal ASNs. The (parent) ASN is zero as expected in this case because this is a stand-alone element. The right part of the Figure shows the final (level 2) descendents and the associated ASNs.

This test shows the nodal ASNs and the (parent) pinball ASNs for a stand-alone 3D hexahedron (CUBE). A hierarchic pinball of level 2 (MLEV 2) is embedded in the element and the option OPTI PINS VIDE is activated in order to consider all descendents in contact and to be able to visualize the final descendents and their ASNs.

The left part of Figure 56 shows the parent pinball and the nodal ASNs. The (parent) ASN is zero as expected in this case because this is a stand-alone element. The right part of the Figure shows the final (level 2) descendents and the associated ASNs.

This test shows the nodal ASNs and the (parent) pinball ASNs for a mesh of two 2D quadrilaterals (CAR1). Hierarchic pinballs of level 2 (MLEV 2) are embedded in the elements and the option OPTI PINS VIDE is activated in order to consider all descendents in contact and to be able to visualize the final descendents and their ASNs.

The left part of Figure 54 shows the parent pinballs, the nodal ASNs and the (parent) pinball ASNs. The (parent) ASNs are no longer zero, as expected in this case. The right part of the Figure shows the final (level 2) descendents and the associated ASNs.

This test considers the mesh of Figure 1, composed by three quadrilaterals and two triangles in 2D. Parent pinballs are embedded in all the elements.

The left part of Figure 58 shows the parent pinballs, the nodal ASNs and the (parent) pinball ASNs on the whole mesh. The right part of the Figure shows the same quantities but on a subset of the mesh (to check visualization implementation).
This test considers a simplified bar impact in 3D. The mesh is composed by two Cub8 hexahedral elements for each bar. Hierarchic pinballs of level 2 (MLEV 2) are embedded only in the elements which are likely to come into contact (one element for each bar).

The upper part of Figure 59 shows the parent pinballs, the nodal ASNs, the (parent) pinball ASNs, the contacting descendents and their ASNs. The bottom part of the Figure shows details of the descendental ASNs and of the resulting contact normal directions. Note that the contact normals (in green) are all vertical, i.e. in the direction of the impact, although the same is not true for the descendental ASNs.

This test considers the impact between two rectangular bodies in 2D. The mesh is composed by two hundred CAR1 quadrilateral elements for each rectangular body. Parent pinballs of level 0 are embedded only in the elements which are likely to come into contact, i.e. along the surface of the two bodies.

The left part of Figure 60 shows the parent pinballs, the right part shows the (parent) pinball ASNs and the nodal ASNs.

This test considers a hollow circular body in 2D. The mesh is composed by 360 CAR1 quadrilateral elements. Parent pinballs of level 0 are embedded only in the elements along the surface (both internal and external) of the body.

The left part of Figure 61 shows the parent pinballs, the right part shows the (parent) pinball ASNs and the nodal ASNs.

These tests visualize (with the VIDE option) pinballs in 2D or 3D bars, at different levels of refinement (MLEV 0, 1 or 2). As shown in Figure 62, at zero level no (parent) pinballs have any associated ASN. At higher level, some descendental pinballs (those not near to an “isolated” vertex) have non-zero ASNs in 2D. However, in 3D all ASNs are always zero because it is not possible to uniquely define the normal to a bar in 3D space.

Furthermore, note that in 2D the non-isolated descendents are face (F) pinballs while in 3D they are corner (C) pinballs.
Figure 54 - ASNs in case VIDE01.

Figure 55 - ASNs in case VIDE02.
Figure 56 - ASNs in case VIDE05.

Figure 57 - ASNs in case VIDE07.
Figure 58 - ASNs in case ASNO01.
Figure 59 - ASNs in case ASNO02.
Figure 60 - ASNs in case ASNO03.

Figure 61 - ASNs in case ASNO04.
Figure 62 - ASNs in cases FUN201, FUN202, FUN203, FUN301, FUN302, FUN303.
5.2 Visualization of the ASNs and of the (sub-)pinball types

The next set of tests checks the ASNs and the resulting (sub-)pinball types in simple cases for the different element types. They are listed in Table 5 and described hereafter.

<table>
<thead>
<tr>
<th>Name</th>
<th>Mesh</th>
<th>Contact parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASNC01</td>
<td>4 CAR1</td>
<td>2 BODY MLEV 0</td>
<td>Contact with continuum elements in 2D</td>
</tr>
<tr>
<td>ASNC02</td>
<td>4 CAR1</td>
<td>2 BODY MLEV 1</td>
<td>Contact with continuum elements in 2D</td>
</tr>
<tr>
<td>ASNC03</td>
<td>4 CAR1</td>
<td>2 BODY MLEV 2</td>
<td>Contact with continuum elements in 2D</td>
</tr>
<tr>
<td>ASNC04</td>
<td>8 CAR1</td>
<td>2 BODY MLEV 0</td>
<td>Contact with continuum elements in 2D</td>
</tr>
<tr>
<td>ASNC05</td>
<td>5 BARR</td>
<td>1 BODY MLEV 0</td>
<td>Contact with bar elements in 2D</td>
</tr>
<tr>
<td>ASNC06</td>
<td>5 BARR</td>
<td>1 BODY MLEV 1</td>
<td>Contact with bar elements in 2D</td>
</tr>
<tr>
<td>ASNC07</td>
<td>5 Q4GS</td>
<td>1 BODY MLEV 0</td>
<td>Contact with shell elements in 3D</td>
</tr>
<tr>
<td>ASNC08</td>
<td>5 Q4GS</td>
<td>1 BODY MLEV 1</td>
<td>Contact with shell elements in 3D</td>
</tr>
<tr>
<td>ASNC09</td>
<td>5 Q4GS</td>
<td>1 BODY MLEV 2</td>
<td>Contact with shell elements in 3D</td>
</tr>
<tr>
<td>ASNC10</td>
<td>5 Q4GS</td>
<td>1 BODY MLEV 3</td>
<td>Contact with shell elements in 3D</td>
</tr>
<tr>
<td>ASNC11</td>
<td>10 Q4GS</td>
<td>1 BODY MLEV 1</td>
<td>Contact with shell elements in 3D</td>
</tr>
<tr>
<td>ASNC12</td>
<td>10 Q4GS</td>
<td>1 BODY MLEV 3</td>
<td>Contact with shell elements in 3D</td>
</tr>
</tbody>
</table>

Table 5 - Tests to show the ASNs and the (sub-)pinball types.

**ASNC01**

Two blocks each formed by two CAR1 quadrilateral elements are in contact. Zero level (MLEV 0, by default) pinballs are specified for each block. Note that, since there is just one layer of continuum elements in each contacting body, we are in the pathological case of Figure 10. However, there are no element pinballs with zero normals here, since each body contains only two elements. Figure 63 shows the contacting pinballs, which in this case coincide with the parent pinballs (left), the nodal ASNs, parent ASNs and contact normals (right) and the descendent ASNs (which in this case coincide with the parent ASNs). Nodal ASNs have the expected directions. The types of contacting pinballs (TYPEI, TYPEJ) are 0 and the nodes list (NODEI (:), NODEJ (:)) are also 0, as expected since these quantities are defined only for sub-pinballs (at level L > 0).

**ASNC02**

This test is similar to ASNC01 but here we take MLEV 1 pinballs. Figure 64 shows the contacting pinballs (left), the nodal ASNs, parent ASNs and contact normals (right) and the descendent ASNs. Nodal ASNs have the expected directions. The types of contacting pinballs and the nodes list are as expected: magenta indicates $V$ pinballs and blue indicates $F$ pinballs.
ASNC03
This test is similar to ASNC01 but here we take MLEV 2 pinballs. Figure 65 shows the contacting pinballs (left), the nodal ASNs, parent ASNs and contact normals (right) and the descendent ASNs. Nodal ASNs have the expected directions. The types of contacting pinballs and the nodes list are as expected: magenta indicates $V$ pinballs and blue indicates $F$ pinballs.

ASNC04
This test is similar to ASNC01 but we have two layers of CAR1 elements in each contacting body, instead of just one, for a total of 8 CAR1 elements. This is to avoid “degenerated” parent normals (the horizontal blue arrows in Figure 63 for the ASNC01 case). Figure 66 shows the contacting pinballs (left), the nodal ASNs, parent ASNs and contact normals (right) and the descendent ASNs. Nodal ASNs have the expected directions. The types of contacting pinballs and the nodes list are as expected: magenta indicates $V$ pinballs and blue indicates $F$ pinballs. The parent normals now have a more physical direction.

ASNC05
This test checks ASNs in 2D bar elements (BARR). Zero-level pinballs (MLEV 0 by default) are specified. Figure 67 shows the parent pinballs, the parent ASNs and the nodal ASNs (which are 0 in this case) in the left part, the contacting descendents (of which there are none in this case, since there is only one BODY) in the right part.

ASNC06
This test is identical to ASNC05 but uses MLEV 1. The OPTI PINS VIDE option is added in order to activate contacts (with the 0-pinball) in all descendent pinballs, so that they can be visualized. Figure 68 shows the parent pinballs, the parent ASNs and the nodal ASNs (which are 0 in this case) in the left part, the contacting descendents and their ASNs in the right part.

As expected, contacting descendents are either of vertex ($V$) type, represented in magenta and with a 0 associated ASN in this case, or of face ($F$) type, represented in blue.

ASNC07
This test is the 3D version of case ASNC05, and uses Q4GS quadrilateral shell elements. Zero-level pinballs (MLEV 0 by default) are specified. Figure 69 shows the parent pinballs, the parent ASNs and the nodal ASNs (which are 0 in this case) in the left part, the contacting descendents (of which there are none in this case, since there is only one BODY) in the right part.
**ASNC08**

This test is identical to ASNC08 but uses MLEV 1. The OPTI PINS VIDE option is added in order to activate contacts (with the 0-pinball) in all descendent pinballs, so that they can be visualized. Figure 70 shows the parent pinballs, the parent ASNs and the nodal ASNs (which are 0 in this case) in the left part, the contacting descendents and their ASNs in the right part. As expected, contacting descendents are all of vertex \((V)\) type, represented in magenta and with a 0 associated ASN in this case.

**ASNC09**

This test is identical to ASNC08 but uses MLEV 2. The OPTI PINS VIDE option is added in order to activate contacts (with the 0-pinball) in all descendent pinballs, so that they can be visualized. Figure 71 shows the parent pinballs, the parent ASNs and the nodal ASNs (which are 0 in this case) in the left part, the contacting descendents and their ASNs in the right part. As expected, contacting descendents are either of vertex \((V)\) type, represented in magenta and with a 0 associated ASN, or of corner \((C)\) type, represented in cyan and with a 0 associated ASN, or of face \((F)\) type, represented in blue and with an associated non-zero ASN.

**ASNC10**

This test is identical to ASNC08 but uses MLEV 3. The OPTI PINS VIDE option is added in order to activate contacts (with the 0-pinball) in all descendent pinballs, so that they can be visualized. Figure 72 shows the parent pinballs, the parent ASNs and the nodal ASNs. As expected, contacting descendents are either of vertex \((V)\) type, represented in magenta and with a 0 associated ASN, or of corner \((C)\) type, represented in cyan and with a 0 associated ASN, or of face \((F)\) type, represented in blue and with an associated non-zero ASN.

**ASNC11**

This test is similar to ASNC08 but five more Q4GS elements are added so as to have some shell nodes completely surrounded by shell elements. The descendent pinballs (MLEV 1) near such nodes must be of the face \((F)\) type, as shown in Figure 73.

**ASNC12**

This case is similar to ASNC11 but uses MLEV 3. Results are shown in Figure 74.
Figure 63 - ASNs in case ASNC01.
contacting descendent pinballs

nodal ASNs, parent ASNs and contact normals

descendent ASNs

Figure 64 - ASNs in case ASNC02.
Figure 65 - ASNs in case ASNC03.
contacting pinballs (parent pinballs in this case)  nodal ASNs, parent ASNs and contact normals

descendent ASNs (parent ASNs in this case)

---

Figure 66 - ASNs in case ASNC04.
parent pinballs, parent ASNs and nodal ASNs (0) contacting descendents (none) and desc. ASNs

Figure 67 - ASNs in case ASNC05.

parent pinballs, parent ASNs and nodal ASNs (0) contacting descendents and descendent ASNs

Figure 68 - ASNs in case ASNC06.
Figure 69 - ASNs in case ASNC07.

Figure 70 - ASNs in case ASNC08.
parent pinballs, parent ASNs and nodal ASNs (0) contacting descendents and desc. ASNs

Figure 71 - ASNs in case ASNC09.

parent pinballs, parent ASNs and nodal ASNs (0) contacting descendents and desc. ASNs

Figure 72 - ASNs in case ASNC10.
Figure 73 - ASNs in case ASNC11.

Figure 74 - ASNs in case ASNC12.
5.3 Simple impact tests

The next set of tests shows simple impact simulations.

5.3.1 Impact between continuum elements

We start by considering impacts between continuum elements. They are listed in Table 6 and are described hereafter.

<table>
<thead>
<tr>
<th>Name</th>
<th>Mesh</th>
<th>Contact parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PENE01</td>
<td>2 CAR1</td>
<td>PENA MLEV 0 ASN</td>
<td>single-element impact in 2D</td>
</tr>
<tr>
<td>PENE02</td>
<td>4 CAR1</td>
<td>PENA MLEV 0 ASN</td>
<td>two plus two element impact in 2D</td>
</tr>
<tr>
<td>PENE03</td>
<td>4 CUBE</td>
<td>PENA MLEV 0 ASN</td>
<td>two plus two element impact in 3D</td>
</tr>
<tr>
<td>PENE04</td>
<td>4 CUBE</td>
<td>PENA MLEV 0</td>
<td>two plus two element impact in 3D</td>
</tr>
<tr>
<td>PENE05</td>
<td>4 CUBE</td>
<td>PENA MLEV 2 VIDE ASN</td>
<td>two plus two element impact in 3D</td>
</tr>
<tr>
<td>PENE06</td>
<td>4 CUBE</td>
<td>PENA MLEV 2 ASN</td>
<td>two plus two element impact in 3D</td>
</tr>
</tbody>
</table>

Table 6 - Simple impact tests between continuum elements.

PENE01

This test considers the impact between two quadrilaterals in 2D, represented by just one CAR1 element each. A parent pinball (level 0) is embedded in each of the elements. A zero initial gap exists between the two elements, leading to a large initial interpenetration of the parent pinballs.

The left part of Figure 75 shows the parent pinballs and the nodal ASNs. The right part of the Figure shows the nodal ASNs, the (parent) ASNs (which are zero in this case, because these are two stand-alone elements) and the contact normal (in green).

Note that in this case the general formula (25) for the contact normal (or for the penetration direction) would give an undefined direction because the two pinballs normals are zero. In such a case the code takes the centers-joining line (see eq. 72) as the best possible guess for the contact direction, as shown in the right part of Figure 75. This is just to avoid a fatal error message. However, it remains that the solution of this contact problem with such a coarse mesh is non-physical, especially if the contact forces are uniformly distributed over all the element’s nodes like in the present case. A finer mesh would be needed to obtain plausible results.

Note that in this impact problem (and in the following ones) we systematically specify the OPTI PINS NORB keyword in order to completely disable any special treatment of rebound. According to Section 2.9, such treatments are necessary when using the LM method, but not with the penalty method.
**PENE02**

This test is similar to PENE01 but uses twice longer impacting bars, each discretized by two CAR1 elements. The level of pinballs is 0 and they are embedded only in the two elements that come into contact. An initial gap of 0.42 m is assumed so that the pinballs are not in contact initially. Opposite initial velocities of 50 m/s are assumed.

The left part of Figure 76 shows the parent pinballs, which enter into contact at step 2. The right part of the Figure shows the nodal ASNs and the (parent) pinball ASNs.

Figure 77 shows the nodal displacements of two nodes on the opposite contacting surfaces and Figure 78 shows the corresponding contact forces.

**PENE03**

This test is the 3D version of PENE01 and uses two CUBE elements for each bar. The level of pinballs is 0 and they are embedded only in the two elements that come into contact. An initial gap of 0.75 m is assumed so that the pinballs are not in contact initially. Opposite initial velocities of 50 m/s are assumed.

The left part of Figure 79 shows the parent pinballs, which enter into contact at step 5. The right part of the Figure shows the nodal ASNs and the (parent) pinball ASNs.

Figure 80 shows the nodal displacements of two nodes on the opposite contacting surfaces and Figure 81 shows the corresponding contact forces.

**PENE04**

This test is similar to PENE03 but without the OPTI PINB ASN option.

The left part of Figure 82 shows the parent pinballs, which enter into contact at step 5. The right part of the Figure shows the nodal ASNs and the (parent) pinball ASNs, which of course are all zero in this case because the ASN model is not activated.

Figure 83 shows the nodal displacements of two nodes on the opposite contacting surfaces and Figure 84 shows the corresponding contact forces.

The results are identical to those of case PENE03 which used the ASN method. This is because the two methods give the same (perfectly vertical) contact normal in this particular case (perfectly aligned impact).

**PENE05**

This test is similar to PENE03 but with MLEV 2 and VIDE option to show all the descendent pinballs and the associated ASNs.
The left upper part of Figure 85 shows the parent pinballs, the nodal ASNs and the (parent) pinball ASNs. The right upper part of the Figure shows the descendent pinballs and the associated ASNs. The lower part of the Figure shows a detail of the descendent pinball ASNs.

**PENE06**

This test is similar to PENE05 but without the **VIDE** option, so that a real calculation is performed. The initial gap is 0.2 m.

Figure 86 shows the nodal displacements of two nodes on the opposite contacting surfaces and Figure 87 shows the corresponding contact forces.
Figure 75 - ASNs in case PENE01.

Figure 76 - ASNs in case PENE02.
Figure 77 - Displacements in case PENE02.

Figure 78 - Contact forces in case PENE02.
Figure 79 - ASNs in case PENE03.

Figure 80 - Displacements in case PENE03.
Figure 81 - Contact forces in case PENE03.

Figure 82 - ASNs in case PENE04.

parent pinballs

nodal and pinball ASNs
Figure 83 - Displacements in case PENE04.

Figure 84 - Contact forces in case PENE04.
parent pinball, nodal and pinball ASNs
descendent pinballs and their ASNs
detail of descendent pinball ASNs

Figure 85 - ASNs in case PENE05.
Figure 86 - Displacements in case PENE06.

Figure 87 - Contact forces in case PENE06.
5.3.2 Impacts between material points

An advantage of the penalty formulation with respect to the Lagrange Multipliers formulation is that it allows to treat impacts between material points, i.e. between particles represented by one-node “elements” in the same way as for continuum or shell elements. The same is not true for the Lagrange Multipliers method (in its standard form), in which a difficulty would arise in modeling the rebound between the impacting particles.

An example is considered in Table 7:

<table>
<thead>
<tr>
<th>Name</th>
<th>Mesh</th>
<th>Contact parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMAT01</td>
<td>2 PMAT</td>
<td>PENA DIAM 1.0 ASN</td>
<td>material point impact in 2D</td>
</tr>
</tbody>
</table>

Table 7 - Impact tests between material points.

**PMAT01**

Two particles hit each other at a certain initial velocity. The particles material is elastic, so after some interaction rebound occurs. Pinballs are associated to the particles, with a prescribed diameter (DIAM). These are parent (0-level) pinballs since the hierarchic pinball method cannot be applied to particles. A penalty method (PENA) is used. The ASN method is formally invoked: however, this has no influence in the present case since no ASN can be associated with a particle, and therefore the contact occurs always along the line joining the contacting particle centers.

A “thickness” (diameter, in this case) of 1.0 is assigned to the particles via the COMP EPAI directive. This is used for the visualization of the particles, but also for the calculation of their mass and critical time step. The latter two quantities are indeed computed because a material LINE (and not the more usual material MASS) is associated with the particles. The elastic properties of the material are used to compute the penalty forces during contact.

Note that the diameter of the pinballs associated with the particles is assigned independently in the PINB DIAM directive. Here the same value as the physical diameter of the particles (1.0 units) is chosen, which seems the only reasonable choice. However, be aware that the choice is left to the user and the code will use any value prescribed, without checking that it is equal to the value assigned to the particle diameter via the COMP EPAI directive.

Figure 88 shows the two particles (via the associated pinballs) during the contact phase, before rebound starts. Figures 89, 90 and 91 show the displacements, velocities and contact forces. As it can be observed, the rebound is elastic in that the rebound velocity is equal and opposite to the initial velocity, and the interaction is quite smooth (no oscillations).
Figure 88 - Contacting particles in test PMAT01.

Figure 89 - Displacements in case PMAT01.
Figure 90 - Velocities in case PMAT01.

Figure 91 - Contact forces in case PMAT01.
5.3.3 Impacts between material points and other element types

We now consider impact between material points and other element types, namely shell elements. Examples are considered in Table 7:

<table>
<thead>
<tr>
<th>Name</th>
<th>Mesh</th>
<th>Contact parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMAT02</td>
<td>1 PMAT, 81 Q4GS</td>
<td>PENASN</td>
<td>material point impacts 3D plate</td>
</tr>
<tr>
<td>PMAT03</td>
<td>1 PMAT, 100 Q4GS</td>
<td>PENASN</td>
<td>same as 02 but finer mesh</td>
</tr>
<tr>
<td>PMAT04</td>
<td>1 PMAT, 361 Q4GS</td>
<td>PENASN</td>
<td>even finer mesh</td>
</tr>
<tr>
<td>PMAT05</td>
<td>1 PMAT, 400 Q4GS</td>
<td>PENASN</td>
<td>even finer mesh</td>
</tr>
<tr>
<td>PMAT06</td>
<td>1 PMAT, 361 Q4GS</td>
<td>LINKCOUP ASN</td>
<td>same as 04 but LINK COUP</td>
</tr>
<tr>
<td>PMAT07</td>
<td>1 PMAT, 400 Q4GS</td>
<td>LINKCOUP ASN</td>
<td>same as 05 but LINK COUP</td>
</tr>
</tbody>
</table>

Table 8 - Impact tests between material points and other element types.

**PMAT02**

A material particle hits a square plate at a certain initial velocity. Both the particle and the plate have linear elastic material. The plate is clamped along its contour (all displacements and all rotations blocked). The plate is discretized by a regular grid of $9 \times 9$ shell elements Q4GS. The PENA method with ASN is used to describe the contact.

Figure 92 shows an example of contact occurring during the test, namely the secondary contact when the deformed plate starts bouncing back and hits the particle. Figures 93, 94 and 95 show the displacement, velocity and contact force on the particle. Contact occurs in two phases: first the particle hits the plate, which deforms and detaches from the particle. Then the plate bounces back and hits the particle. Contact is quite smooth, as already observed in the previous example.

**PMAT03, PMAT04 and PMAT05**

These tests are repetitions of case PMAT02 using finer and finer meshes for the plate, up to 400 elements, in order to check the convergence of the numerical solution to a stabilized result. The reason for using meshes differing by just one element along each spatial direction is that when the element number is odd the impact occurs at the centre of an element (face pinball) while when the number is even the impact occurs exactly at one node of the plate mesh. Results should not be sensitive to such details.

Figures 96, 97 and 98 compare solutions PMAT02 and PMAT03 (9 and 10 shell elements respectively), showing some significant differences, so that finer meshes are needed.
Figures 99, 100 and 101 compare solutions PMAT04 and PMAT05 (19 and 20 shell elements respectively), showing better convergence although some differences remain especially in the contact forces.

**PMAT06 and PMAT05**

These tests are repetitions of cases PMAT04 (19 elements) and PMAT05 (20 elements), respectively, by using Lagrange Multipliers (coupled links) instead of penalty method to describe contact. The ASN algorithm is kept and is used in combination with the Lagrange Multipliers method.

Results of these two calculations are compared with case PMAT05 in Figures 102, 103 and 104. While solution PMAT06 is relatively similar to PMAT05, solution PMAT07 has a quite different behavior. Contact forces tend to act over much longer periods in the solutions with Lagrange Multipliers, compared with those with penalty. This might indicate a problem in treating the rebound (which is necessary with the Lagrange Multipliers while it is redundant with the penalty method).

This problem will be better investigated in a subsequent Section, dealing with sliding-like contact between continuum bodies.
Figure 93 - Particle displacement in case PMAT02.

Figure 94 - Particle velocity in case PMAT02.
**Figure 95 - Particle contact force in case PMAT02.**

**Figure 96 - Particle displacement in cases PMAT02 and PMAT03.**
Figure 97 - Particle velocity in cases PMAT02 and PMAT03.

Figure 98 - Particle contact force in cases PMAT02 and PMAT03.
Figure 99 - Particle displacement in cases PMAT04 and PMAT05.

Figure 100 - Particle velocity in cases PMAT04 and PMAT05.
Figure 101 - Particle contact force in cases PMAT04 and PMAT05.

Figure 102 - Particle displacement in cases PMAT05, PMAT06 and PMAT07.
Figure 103 - Particle velocity in cases PMAT05, PMAT06 and PMAT07.

Figure 104 - Particle contact force in cases PMAT05, PMAT06 and PMAT07.
5.3.4 Impacts of “sliding” type between continuum bodies

Finally, we consider impact (or rather contact) of “sliding” type between continuum elements. This type of phenomenon is less violent than impact, but presents some particularities that make it a challenging test for the proposed algorithms.

The problem consists of two blocks of metal, a lower block of rectangular shape which is fixed at its base, and an upper block of square shape initially lying slightly above the first block and with an initial velocity causing a slight impact against (and a lot of sliding along) the fixed block. The horizontal component of the velocity is 100 m/s while the vertical component is -10 m/s. The lower block is discretized by $10 \times 2 = 20$ quadrilateral elements CAR1, while the upper block is discretized by $2 \times 2 = 4$ CAR1. The material is linear elastic with steel-like characteristics.

The executed calculations are listed in Table 7:

<table>
<thead>
<tr>
<th>Name</th>
<th>Mesh</th>
<th>Contact parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLID01</td>
<td>24 CAR1</td>
<td>PENA MLEV 4</td>
<td>Penalty solution without ASN</td>
</tr>
<tr>
<td>SLID02</td>
<td>24 CAR1</td>
<td>PENA MLEV 4 ASN</td>
<td>Penalty solution with ASN</td>
</tr>
<tr>
<td>SLID03</td>
<td>24 CAR1</td>
<td>LINK COUP MLEV 4</td>
<td>Lagrange mult. solution without ASN</td>
</tr>
<tr>
<td>SLID04</td>
<td>24 CAR1</td>
<td>LINK COUP MLEV 4 ASN</td>
<td>Lagrange mult. solution with ASN</td>
</tr>
<tr>
<td>SLID05</td>
<td>2 CAR1</td>
<td>LINK COUP MLEV 4</td>
<td>Lagrange mult. solution without ASN</td>
</tr>
<tr>
<td>SLID06</td>
<td>2 CAR1</td>
<td>LINK COUP MLEV 4 ASN</td>
<td>Lagrange mult. solution with ASN</td>
</tr>
<tr>
<td>SLID07</td>
<td>2 CAR1</td>
<td>LINK COUP MLEV 3 ASN</td>
<td>Lagrange mult. solution with ASN</td>
</tr>
<tr>
<td>SLID08</td>
<td>2 CAR1</td>
<td>LINK COUP MLEV 2 ASN</td>
<td>Lagrange mult. solution with ASN</td>
</tr>
<tr>
<td>SLID09</td>
<td>2 CAR1</td>
<td>LINK COUP MLEV 1 ASN</td>
<td>Lagrange mult. solution with ASN</td>
</tr>
<tr>
<td>SLID10</td>
<td>2 CAR1</td>
<td>LINK COUP MLEV 0 ASN</td>
<td>Lagrange mult. solution with ASN</td>
</tr>
</tbody>
</table>

Table 9 - Sliding-type contact between continuum elements.

**SLID01**

This solution uses the penalty (decoupled) approach to contact, without the ASN algorithm. Two time instants of the solution are presented in Figure 105: the initial configuration and the instant of first contact between the two blocks (step 85).

Figure 106 shows the vertical displacement of the upper block, indicating that rebound (although with some elastic oscillations) occurs regularly. The velocity components of the upper block are shown in Figure 107. The horizontal component should be conserved since there is no friction and the two contacting surfaces are parallel. A slight loss takes place due to the irregular (non-flat) shape of the pinballs: recall that in this case the ASN is not used so the contact occurs in a slightly non-ver-
tical direction. The vertical component indicates rebound with some oscillations. Finally, Figure 108 shows the contact force components: there is a small but non-negligible horizontal component (which should not be there as already mentioned) and this is the cause for dissipation of horizontal velocity.

**SLID02**

This solution is similar to the previous one but the ASN is activated. The contact direction should therefore much closer to the ideal one, which is perfectly vertical (apart some distortions induced by the elastic deformation of the bodies).

Figures 109, 110 and 111 confirm that the solution is better than the previous one: the horizontal velocity stays almost perfectly constant (see also finer comparison in Figure 112 between the two solutions) and the horizontal contact force is practically zero.

**SLID03**

This solution is similar to SLID01 (no ASN) but uses the Lagrange Multiplier algorithm (LINK COUP) instead of the penalty approach. The solution, shown in Figures 113, 114 and 115, is very similar to SLID01: slight loss of horizontal velocity due to a spurious (but small) horizontal contact force. The rebound appears correct.

**SLID04**

This solution is similar to SLID02 (ASN) but uses the Lagrange Multiplier algorithm (LINK COUP) instead of the penalty approach. It would be expected that this solution be better than SLID03 and similar to SLID02 obtained with the penalty method.

However, it is not so: as shown in Figure 116, at a certain moment of the solution the upper block seems to “stick” onto the lower one and starts rotating instead of sliding smoothly. This is confirmed from Figures 117, 118 and 119. The horizontal velocity is completely lost as an enormous horizontal component of the contact force suddenly appears.

The following tests are an attempt at understanding the problem. An obvious candidate for the observed mis-functioning is the rebound model, i.e. the so-called a-priori rebound model described in Section 8.2 of reference [13]. Recall in fact that a rebound model is necessary with the LM version of the contact model by pinballs (unlike in the penalty formulation) and that by default the a-priori rebound model is activated.

Another possible source of trouble is the fact that, as shown in Section 7 of reference [13], the Lagrangian Multipliers approach suffers from redundant constraints which render the links matrix singular: for the configuration of the simplified debugging test SLID05 and following ones (see
next) the matrix is singular already from a hierarchy level of 2, as shown in Section 7.1 (pages 77-79) of [13].

The further debugging tests are done on a simplified version of the sliding contact problem: only one element is used for the lower block and one element for the upper block, in order to simplify the checks.

**SLID05**

This solution is similar to SLID03 (LINK COUP without ASN) but uses the reduced model with only two elements. The solution is acceptable, see Figure 121. The contact forces are not vertical, but this is normal since ASN is not activated in this case. Contact forces are repulsive (or zero), as suggested by the physics of the problem, so the a-priori rebound algorithm seems to work in this case.

**SLID06**

This solution is similar to SLID05 (2-element mesh) but uses the ASN (like in case SLID04). The solution is obviously unacceptable, as appears from Figure 121. Until step 4 the contact forces are repulsive, but from step 5 onwards they become attractive, which is clearly unphysical. The contact forces are vertical as expected, thanks to the ASN method, but the sign is obviously wrong.

Figure 122 shows the contact configurations in steps 0 to 7 for this calculation.

**SLID07 to SLID10**

These solutions are repetitions of case SLID06 (which uses MLEV 4) but with smaller hierarchy levels equal to 3, 2, 1 and 0, respectively. The scope is to see whether the inversion of contact forces occurs also at lower hierarchy levels.

Figure 123 shows the contacts in case SLID07 (MLEV 3). Contact force inversion occurs at step 4 instead of 5, but for the rest the solution is similar to SLID06. The same happens in SLID08 (MLEV 2) as shown in Figure 124.

The result of case SLID09 (MLEV 1) is strange and is shown in Figure 125. At this level there should be no constraint redundancy. However, strangely the contact forces become non-vertical from step 1 onwards, like if the ASN condition would not be respected. The reason for this can be seen in Figure 126: at step 1 the resulting ASN is indeed not vertical due to the particular contact configuration. It seems difficult to avoid such problems in general.

Finally, the case SLID10 (MLEV 0) is a limit case, as expected. The contact occurs only at step 0, (with vertical forces) which is sufficient to completely block the penetration. From step 1 onwards no contact forces occur and perfect sliding takes place, see Figure 127.
Figure 105 - Sliding contact test SLID01.

Figure 106 - Displacement of the upper block center in case SLID01.
Figure 107 - Velocity components of the upper block center in case SLID01.

Figure 108 - Contact force components of the upper block center in case SLID01.
Figure 109 - Displacement of the upper block center in case SLID02.

Figure 110 - Velocity components of the upper block center in case SLID02.
Figure 111 - Contact force components of the upper block center in case SLID02.

Figure 112 - Horizontal velocity components of the upper block center in cases SLID01 and SLI02.
Figure 113 - Displacement of the upper block center in case SLID03.

Figure 114 - Velocity components of the upper block center in case SLID03.
Figure 115 - Contact force components of the upper block center in case SLID03.

Figure 116 - Sliding contact test SLID04.
Figure 117 - Displacement of the upper block center in case SLID04.

Figure 118 - Velocity components of the upper block center in case SLID04.
Figure 119 - Contact force components of the upper block center in case SLID04.

Figure 120 - Sliding contact test SLID05.
Figure 121 - Sliding contact test SLID06.

Step 4 (contact forces are repulsive)

Step 5 (contact forces become attractive)
Figure 122 - Contact configurations in test SLID06 (steps 0 to 7).
Figure 123 - Sliding contact test SLID07.

Step 3 (contact forces are repulsive)

Step 4 (contact forces become attractive)
Figure 124 - Sliding contact test SLID08.
Figure 125 - Sliding contact test SLID09.

Step 0 (contact forces are repulsive and vertical)

Step 1 (contact forces become weird)
Figure 126 - Sliding contact test SLID09, details on contact conditions
Figure 127 - Sliding contact test SLID10.
5.4 Tube array crash test

The next set of tests shows a realistic contact problem, the crushing of an array of tubes (courtesy of Onera Lille), already considered in reference [15]. The initial configuration of the tubes array is shown in Figure 128.

In reference [15] the problem has been studied by a variety of approaches, including either linear-displacement or parabolic-displacement elements and either Lagrangian or ALE formulations (the latter justified by the very large local deformations of the structure). In all cases, pinballs with a Lagrange Multipliers method (LINK COUP) were employed for the contact.

Some of these simulations are repeated by adding the new ASN technique, and also a penalty-based solution (rather than LM) is attempted. We choose as a reference the test CARA07 of reference [15], which is an ALE simulation using the Q42 element, although this solution was an over-stiff one (the Q42 element is a fully integrated linear-displacement quadrilateral).
The new simulations are listed in Table 10 and are described hereafter.

<table>
<thead>
<tr>
<th>Name</th>
<th>Mesh</th>
<th>Notes</th>
<th>Steps</th>
<th>CPU [s]</th>
<th>Els*step</th>
</tr>
</thead>
<tbody>
<tr>
<td>CARN07</td>
<td>4080</td>
<td>Q42 LINK DECO PINB PENA SELF DMIN 0.1</td>
<td>89,289</td>
<td>1,681</td>
<td>3.64×10^8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PINS GRID EQVF ASN NORB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAR007</td>
<td>4080</td>
<td>Q42 LINK COUP PINB SELF DMIN 0.1</td>
<td>89,950</td>
<td>1,543</td>
<td>3.67×10^8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PINS GRID EQVF ASN NORB</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 10 - Tube array crash tests.

**CARN07**

This test is a repetition of CARA07 of reference [15], but uses de-coupled boundary conditions instead of coupled (LM) ones. This means that the blockages and symmetries are imposed via LINK DECO BLOQ (instead of LINK COUP BLOQ), and that the contact is treated by LINK DECO PINB PENA instead of LINK COUP PINB. A penalty coefficient SFAC 1.0 is chosen. In addition, the ASN algorithm is activated by adding the option OPTI PINS ASN NORB. The latter keyword disables the treatment of a-priori rebound, which would be active by default, since normally with the penalty method the contact force should be applied also during the rebound phase, as long as there is penetration.

Figure 129 shows the initial configuration with nodal ASNs (in magenta) and parent pinball ASNs (in blue). Figure 130 shows the nodal ASNs in an intermediate configuration, at time 0.615 s. Figures 131 and 132 show the contact normal directions and the contact forces, respectively, for some contacting descendent pinballs near one of the folds at 0.76 s. Note that the contact forces are directed along the normal, and the normal is very reasonably perpendicular to the local contact surface (which is not always the case with standard pinballs without the ASN algorithm).

Figures 133 and 134 show the contacts and the equivalent plastic strains in the final configuration, i.e. at time 1.0 s.

Figure 135 shows the upper and lower plate crushing forces. As already noted in the simulations of reference [15] the two curves are practically identical, except for a very short initial period where some slight dynamic effects are present. Figure 136 compares the crushing forces in solutions CARA07 (no ASN, coupled links and in particular LM formulation for the contacts), shown in green, and CARN07 (ASN, de-coupled links and in particular penalty formulation for the contacts), shown in black. The penalty/ASN solution shows a reduction of the crushing force in the final part of the transient. The final value of the force passes from 6,100 to 5,400, a reduction of 11%, and should therefore be in better agreement with the experimental result.
**CARO07**

This test is a repetition of CARN07 but uses the coupled (LM) formulation for the links (LINK COUP BLOQ and LINK COUP PINB). The difference with respect to case CARA07 is therefore only in the ASN algorithm for the determination of the contact normal. The scope is to see whether the differences observed in the previous solution with respect to CARA07 are due to the penalty formulation, or to the ASN (or to both).

Figures 137 and 138 show the contacts and the equivalent plastic strains in the final configuration, i.e. at time 1.0 s.

Figure 139 compares the crushing forces in solutions CARA07 (in green), CARN07 (in red) and CARO07 (in black). The latter solution shows only a very slight reduction of final crushing force with respect to case CARA07. Therefore, most of the reduction in force observed in the previous solution comes from the penalty formulation (vs. the LM formulation) and the ASN seems to have only a minor effect in this particular example.

**Figure 129 - Nodal ASNs and parent pinball ASNs in case CARN07 (initial configuration).**
Figure 130 - Nodal ASNs in case CARN07 at 0.615 s.

Figure 131 - Contact normals in case CARN07 at 0.76 s.
Figure 132 - Contact forces in case CARN07 at 0.76 s.

Figure 133 - Final contacts in case CARN07.
Figure 134 - Final plastic strains in case CARN07.

Figure 135 - Upper and lower crushing forces in case CARN07.
Figure 136 - Crushing forces in cases CARA07 and CARN07.

Figure 137 - Final contacts in case CARO07.
Figure 138 - Final plastic strains in case CARO07.

Figure 139 - Crushing forces in cases CARA07, CARN07 and CARO07.
6. References


[13] Casadei F. A General Impact-Contact Algorithm Based on Hierarchic Pinballs for the EURO-


7. Appendix A - Closest points on two segments

To treat contact between 3D bars, it is useful to compute the closest points on the bars. Let us first consider the problem of finding the closest points on two (infinite) lines \( L_1, L_2 \), shown in Figure 140. We assume for simplicity that the two lines are not parallel.

If \( P_0, Q_0 \) are points on the lines and \( u, v \) are vectors directed along the lines, and \( s, t \) are (scalar) parameters, then the two lines can be represented by the following parametric vector equations:

\[
L_1: \quad P(s) = P_0 + su \\
L_2: \quad Q(t) = Q_0 + tv.
\]  

(98)

Let \( w(s, t) = P(s) - Q(t) \) be a generic vector between points on the two lines. If we denote \( P(s_c) \) and \( Q(t_c) \) the points on the two lines which are closest to each other, then the vector \( w_c(s_c, t_c) \) has the minimum length for all \( s \) and \( t \), and is perpendicular to both lines. This vector is unique, i.e. no other vector \( w(s, t) \) has this property. This condition is expressed by the system of equations:

\[
\begin{align*}
  u \cdot w_c &= 0 \\
  v \cdot w_c &= 0.
\end{align*}
\]  

(99)

By replacing the expression of \( w_c \) :

\[
w_c = w(s_c, t_c) = P(s_c) - Q(t_c) = P_0 + s_c u - Q_0 - t_c v = w_0 + s_c u - t_c v
\]  

(100)

(where \( w_0 = P_0 - Q_0 \)) into (99) one gets:

\[
\begin{align*}
  (u \cdot u)s_c - (u \cdot v)t_c &= -u \cdot w_0 \\
  (v \cdot u)s_c - (v \cdot v)t_c &= -v \cdot w_0.
\end{align*}
\]  

(101)

Then by letting the known terms:

\[
\begin{align*}
  a &= u \cdot u \\
  b &= u \cdot v = v \cdot u \\
  c &= v \cdot v \\
  d &= u \cdot w_0 \\
  e &= v \cdot w_0
\end{align*}
\]  

(102)

we obtain the two linear algebraic equations in the two unknown scalars \( s_c, t_c \):

\[
\begin{align*}
  as_c - bt_c &= -d \\
  bs_c - ct_c &= -e.
\end{align*}
\]  

(103)
By solving we obtain:

\[ s_c = \frac{bt_c - d}{a} \]

\[ \frac{b^2t_c - bd}{a} - ct_c = -e \]

\[ b^2t_c - bd - ac_t = -ae \]

\[ ac_t - b^2t_c + bd = ae \]

\[ (ac - b^2)t_c = ae - bd \]

\[ t_c = \frac{ae - bd}{ac - b^2} \]

\[ s_c = \frac{b(ae - bd)}{a(ac - b^2)} \quad \frac{d}{a} = \frac{b(ae - bd) - d(ac - b^2)}{a(ac - b^2)} = \frac{abe - b^2d - acd + b^2d}{a(ac - b^2)} = \frac{be - cd}{ac - b^2} \]

Summarizing, it is:

\[ s_c = \frac{be - cd}{ac - b^2} \]

\[ t_c = \frac{ae - bd}{ac - b^2} \]

whenever the denominator \( ac - b^2 \neq 0 \). One may note that, from (102):

\[ ac - b = (u \cdot u)(v \cdot v) - (u \cdot v) = \]

\[ = \|u\|^2\|v\|^2 - (\|u\|\|v\|\cos \theta)^2 = \]

\[ = \|u\|^2\|v\|^2 - \|u\|^2\|v\|^2\cos^2 \theta = \]

\[ = \|u\|^2\|v\|^2(1 - \cos^2 \theta) = \]

\[ = \|u\|^2\|v\|^2\sin^2 \theta = \]

\[ = (\|u\|\|v\|\sin \theta)^2 \geq 0 \]

where \( \theta \) is the angle between the vectors \( u \) and \( v \), i.e. the denominator is non-negative. The denominator is zero when \( \sin \theta = 0 \), i.e. for \( \theta = 0 \). In this case, the two equations are linearly dependent, the two lines are parallel (contrary to the assumption made above), the distance between the lines is constant and there are infinite couples of points with the minimum distance. One can find the distance by fixing the value of one of the parameters and by using any of the equations to find the other one. For example, by selecting \( s_c = 0 \) we get from (103):

\[ t_c = \frac{d}{b} = \frac{c}{e}. \]
Having solved for \( s_c \) and \( t_c \) we have the points \( P(s_c) \) and \( Q(t_c) \) where the two lines \( L_1 \) and \( L_2 \) are closest. Then the (minimum) distance between the lines is, by using (100):

\[
D(L_1, L_2) = \|w_c\| = \\
= \|P(s_c) - Q(t_c)\| = \\
= \|w_0 + s_c u - t_c v\| = \\
= \left\| (P_0 - Q_0) + \frac{(be - cd)u - (ae - bd)v}{ac - b^2} \right\|
\]

(108)

![Figure 140 - Closest points on two lines.](image)

**Version of the algorithm for use in the computation of the contact normal between two corners**

In order to use the above procedure for the computation of the contact normal between two corner pinballs, some adaptations are needed.

The most important one is that in this case we deal with segments (the corners) and not with infinite lines. Therefore, the resulting “closest” points may not be the closest ones in absolute for the two lines, but should be the closest ones located within the segments.

In input the procedure receives four points \( P_0, P_1, Q_0, Q_1 \) which define the two segments. The \( u \) vector is defined as \( u = \overrightarrow{P_0P_1} = P_1 - P_0 \) and the \( v \) vector as \( v = \overrightarrow{Q_0Q_1} = Q_1 - Q_0 \). After computing \( s_c \) and \( t_c \) by (105), we must make sure that the returned “closest” points \( P(s_c) \) and \( Q(t_c) \) lie within the given segments, since these points represent the centres of pinballs attached to the segments themselves. Therefore we limit the obtained values of \( s_c \) and \( t_c \) to lie within the interval \( 0.0 — 1.0 \) by:

\[
s_c = \min(s_c, 1.0) \quad ; \quad s_c = \max(s_c, 0.0) \\
t_c = \min(t_c, 1.0) \quad ; \quad t_c = \max(t_c, 0.0)
\]

(109)
Obviously, if these operations change the previously obtained values of $s_c$ or $t_c$, then the obtained “minimum distance” segment will not any more be perpendicular to the two segments, in general.
8. Appendix B - Unpublished Material on Pinballs

The following unpublished paper gives more details on the hierarchic pinball contact-impact algorithm with Lagrange Multipliers. The paper is incomplete: Sections 8 (Extension to domain decomposition and to spatial partitioning) and 10 (Summary and conclusions) are missing, and Section 9 (Numerical examples) is incomplete.
Contact-impact in explicit fast transient dynamics
by the hierarchic pinball method
with Lagrange multipliers

Folco Casadei¹, Vincent Faucher²

¹ Joint Research Center, Institute for the Protection and Security of the Citizen
T.P. 480, I-21020 Ispra, Italy
² Commissariat à l’Energie Atomique, Saclay, France
e-mail: folco.casadei@jrc.it, vincent.faucher@cea.fr

SUMMARY

The pinball algorithm first proposed two decades ago by Belytschko and co-workers is a simple, general and robust method for contact detection. If needed, spatial accuracy may be increased almost at will by splitting the element-based parent pinballs into a hierarchy of descendents. However, when using an implicit Lagrange multiplier technique to enforce impenetrability, redundant constraints are generated in situations of smooth or flat contact and the problem becomes ill-conditioned. A general implementation of the hierarchic pinball method is presented and techniques are proposed to improve geometrical aspects of contact enforcement, while at the same time removing the above mentioned redundancies. Extension to parallel domain decomposition and to spatial partitioning is also considered, and several numerical examples are shown.

KEY WORDS: Contact-impact, Pinball algorithm, Explicit, Fast Transient Dynamics.

1 INTRODUCTION

Contact-impact phenomena between solids occur quite often in engineering applications, ranging from metal forming, to automobile crash, to hyper-velocity impact with perforation. Although all these cases involve the notion of contact, the dominating phenomena are quite different physically. In the first case loading is slow, contact is typically smooth, deformations are relatively moderate and friction usually plays a very important role, while in the latter case loading is much faster, reaching complete failure of some parts of the body (fragmentation). As a matter of fact, the range of contact problems is so wide that a variety of approaches and of corresponding algorithms has been developed in the literature to cover the different cases.

In this paper, we focus in particular on transient dynamic applications such as crashes, impacts and explosions occurring as a consequence of natural disasters or of deliberate actions such as terrorist attacks. Loading is typically fast or very fast, friction is usually not a major issue and plastic deformations range from moderate to severe, sometimes up to complete failure and fragmentation of some structural components. Both so-called self-contact (Fig. 1) and body-to-body contact (Fig. 2) are considered.

For this class of problems Finite Element spatial discretization with explicit time integration is typically used. Two major strategies have been proposed in the literature to deal with
the contact phenomena occurring in this context: the so-called slide line (2D) or slide surface (3D) algorithms pioneered by Hallquist, Benson et al. [1-2], and the so-called pinball algorithm first proposed by Belytschko and co-workers [3-7].

Both algorithms consist of two phases: a contact detection (or penetration check) phase, which involves mostly geometrical calculations, followed by a contact enforcement phase whereby suitable contact forces are computed to prevent (further) interpenetration. The first phase is radically different in the two approaches: slide surface algorithms use so-called “slave” nodes and “master” surfaces, while the pinball approach uses simple spheres embedded in the elements. The second phase is quite similar if not identical in the two methods. In fact, a variety of techniques may be used in both cases to impose the contact constraints: e.g. penalty methods, Lagrangian multipliers, or augmented Lagrangian forms.

In general, slide surface algorithms are quite adequate for smooth contact problems, e.g. in metal forming applications. However, in geometrically more complex situations such as crash or perforation the complexity of geometrical calculations grows rapidly and several ambiguous penetration detection cases may occur, especially in the presence of thin shells. In reference [7] Belytschko illustrates some examples (see Fig. 3) and some others are shown by Winkelmuller [8], see Fig. 4. Because only interpenetration between nodes and element faces are checked, some surface-to-surface or edge-to-surface contacts cannot be detected, not to talk about contact between thin (1D) structural members such as beams and rods. Since in complex simulations the edge-to-surface contacts which can develop are not obvious, several simulations may be needed before all contacts are correctly identified. Solution of industrial problems thus often requires a trial-and-error process.

The pinball method, proposed by Belytschko [6] to overcome all the mentioned shortcomings and originally targeted mostly at impact problems with perforation, is attractive because it considerably simplifies geometric contact detection, reducing it to simple interpenetration checks between couples of spheres (pinballs). By using a pinball splitting technique [7] to better fit smaller and smaller pinballs within slender or distorted elements, the accuracy of contact detection may be increased potentially at will. The resulting method, as far as concerns the contact detection phase, is extremely robust because all the ambiguous cases mentioned above are avoided. Furthermore, a single and unified approach may be applied—with obvious benefits in terms of generality, robustness and computational speed—to all types of finite elements used in realistic discretizations, ranging from bulky continuum-like elements to structural members (shells, beams, bars) and even to material points (e.g. for coupling between FE and particle-based methods).

Yet another advantage of the pinball algorithm which, perhaps somewhat strangely, is not emphasized by Belytschko in the mentioned references is the fact that, unlike the slide surface technique, the algorithm is inherently symmetric. There is no need, at least in principle, to artificially distinguish between a master and a slave body, since pinballs are embedded in both bodies in exactly the same way. Thus input data is substantially simpler and safer and, most importantly, a unique solution for the contact problem is obtained. This is not always the case with sliding surfaces, where the results change, sometimes even considerably, if the definition of master and slave bodies is swapped.

For the above reasons, especially for its generality and robustness in contact detection, the pinball method is an ideal candidate for implementation in a fast transient explicit code, for the treatment of a large variety of contact applications. This has been realized in the EUROPLEXUS code, developed in collaboration by the Joint Research Center (JRC) of the European Commission and by the French Commissariat à l’Energie Atomique (CEA). The main method chosen for contact enforcement is based upon an implicit treatment via La-
grange multipliers, but recently also a fully explicit version with penalty is under testing. Several difficulties have emerged in the implementation via Lagrange multipliers, mainly related to the presence of flat or smooth contacts. The nature of these difficulties is examined in detail below and solutions are proposed, suggested by the comparison with slide surface algorithms, which behave better in these particular situations.

The paper is organized as follows. Section 2 briefly recalls the basic equations of dynamics and the transient explicit solution scheme, into which the contact algorithm is embedded. Section 3 summarizes the chosen Lagrange multipliers approach. The geometrical aspects of contact detection via a hierarchic pinball method are detailed in Section 4. Section 5 deals with the treatment of rebound and Section 6 with the elimination of redundant constraints, which appear to be an inevitable consequence of hierarchic pinball methods. Section 7 briefly discusses the implications of contact over the choice of time integration steps and the optimization of contact search. Section 8 presents the extension of the contact algorithm to domain decomposition and spatial partitioning. Finally, some conclusions are given in Section 9.

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

2 TRANSIENT EXPLICIT SOLUTION SCHEME

Before introducing the contact algorithm, it is useful to briefly recall the governing equations at the base of the transient explicit formulation. We assume for simplicity a Lagrangian description, suitable for the treatment of purely structural applications.

2.1 Governing equation

The governing equation is the conservation of momentum. By expressing equilibrium in the current configuration and by introducing a spatial semi-discretization based on Finite Elements, the following set of discrete differential equations in time may be obtained (bold-face symbols indicate non-scalar quantities):

$$Ma = F^{\text{ext}} - F^{\text{int}},$$

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

$$F^{\text{int}} = \sum_e \int_{V_e} B^T \sigma dV.$$  

In (2) the summation symbol represents the ordinary assembly operator over all elements $e$ of the mesh, $V_e$ is the element volume in the current configuration, $B$ is the matrix of shape function derivatives, of which a superposed $T$ indicates the transpose, and $\sigma$ is the Cauchy or “true” stress tensor. The nodal accelerations are formally obtained from (1) as:

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

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

$$a_j = (F^{\text{ext}}_j - F^{\text{int}}_j)/m_j.$$
2.2  Explicit time integration

Time integration of (1) under the form (4) is achieved via the so-called central difference (CD) scheme, which is implemented as follows in a typical explicit code. Assume that a complete solution, i.e. all discretized quantities, are known at time $t^n$. First, an intermediate velocity at mid-step is introduced:

$$v_{n+1/2}^n = v^n + (\Delta t / 2)a^n.$$  

(5)

This is the constant velocity that would transform configuration $n$ into $n+1$ over a time interval $\Delta t = t^{n+1} - t^n$. Then the new displacements are given by:

$$d^{n+1} = d^n + \Delta t v_{n+1/2}^{n+1}.$$  

(6)

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

$$x_{n+1}^{n+1} = x^n + \Delta t v_{n+1/2}^{n+1} = x^0 + d^{n+1},$$  

(7)

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

$$v^{n+1} = v^n + (\Delta t / 2)(a^n + a^{n+1}).$$  

(8)

The CD 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 except possibly for the enforcement of essential boundary conditions, in case an implicit treatment via Lagrange multipliers is chosen.

3  TREATMENT OF CONSTRAINTS

Most essential boundary conditions may be formulated as linear constraints of the form:

$$Cv = b,$$  

(9)

where $C$ is a matrix of known coefficients, $v$ is the vector (subset) of linked degrees of freedom (dofs), typically the velocities, and $b$ is a known vector. In general, both $C$ and $b$ may be function of time. A variety of methods may be used to impose such constraints, among which perhaps the most popular are the penalty method (explicit) and the Lagrange multipliers method (implicit).

In the EUROPLEXUS code Lagrange multipliers have been chosen by CEA [9] as the default technique because of their generality. The implicit formulation ensures proper combination of all constraints and unique solutions, irrespective of the order in which the constraints are given. There are in fact no specific input parameters to be “tuned”, unlike in the penalty method. This simplifies also input preparation and input error detection. In fact, if the imposed conditions are physically incompatible (as may be not readily evident in large complex applications) the resulting linear system becomes singular and the problem is promptly detected. The price to be paid is the computational cost of the implicit solution of a linear system in an otherwise fully explicit environment. However, the CPU overhead with respect to an explicit technique is usually negligible, if the subset of constrained dofs is relatively small.

3.1  Lagrange multipliers

With reference to the time integration scheme of Section 2.2, suppose that a configuration $n+1$ at $t^{n+1}$ has been reached. The velocity and acceleration corresponding to this configura-
tion are not known yet, but the internal forces and the external loads (natural boundary conditions) are known since they depend only upon the current configuration and upon time. For simplicity, all quantities expressed at time $t^{n+1}$, i.e. in the current configuration, are indicated without the superscript $n+1$ in the following discussion.

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

$$ m\alpha = f_{\text{ext}} - f_{\text{int}} + r = f + r, $$

where $m$ is the mass matrix, $\alpha$ is the vector of accelerations, $f_{\text{ext}}$ and $f_{\text{int}}$ are the vectors of external loads and of internal forces, respectively, $r$ are the unknown reactions and we have posed $f = f_{\text{ext}} - f_{\text{int}}$ for simplicity. Note that similar, but distinct, symbols have been used here with respect to eq. (1)—e.g. $m$ instead of $M$ for the mass matrix, etc.—to stress the fact that these equations involve just a (usually small) subset of the degrees of freedom in eq. (1).

Now assume that essential boundary conditions are expressed by a linear set of constraints of the form (9) on the velocities or, more precisely, on the next mid-step ($t^{n+3/2}$) velocities:

$$ C\upsilon_{n+3/2} = b. $$

To solve equation (10) for the accelerations $\alpha$, one should first determine the unknown reaction forces $r$. To this end, use is made of Lagrange multipliers associated with the constraint (11). Without loss of generality, the unknown reactions can in fact be expressed as:

$$ r = C^T\lambda, $$

where $\lambda$ is the vector of Lagrange multipliers. Substituting into (10) yields:

$$ m\alpha = f + C^T\lambda. $$

Before introducing into this equation the constraint (11), which is based upon the velocities, one should first transform (11) into the equivalent form expressed on the accelerations. This is achieved by exploiting the time integration scheme described in Section 2.2.

The central difference scheme for the velocity, see eqs. (5) and (8), gives:

$$ \upsilon_{n+3/2} = \upsilon_{n+1/2} + (\Delta t^n + \Delta t^{n+1})\alpha / 2 = \upsilon_{n+1/2} + \gamma\alpha, $$

where again a different symbol $\upsilon$ instead of $v$ is used for the subset of linked dofs and $\gamma$ indicates the (known) coefficient $\gamma = (\Delta t^n + \Delta t^{n+1})/2$. With this the constraint (11) becomes:

$$ C\upsilon_{n+3/2} + C\gamma\alpha = b \quad \text{or} \quad C\gamma\alpha = b - C\upsilon_{n+1/2}. $$

Equations (15) may be interpreted as equivalent forms of the constraint (11), expressed on the accelerations rather than on the velocities. Note in fact that the old velocities $\upsilon^{n+1/2}$ are known and therefore the right-hand side of the second of (15) is a known vector. Multiplying the equilibrium equation in the form (13) by $C\gamma m^{-1}$ and rearranging gives:

$$ C\gamma m^{-1}C^T\lambda = C\gamma\alpha - C\gamma m^{-1}f \quad \text{or} \quad D\lambda = w. $$

By solving this linear system one finds the Lagrange multipliers $\lambda$, and then the reactions $r$ are obtained from (12). Finally, the accelerations $\alpha$ along the constrained dofs are explicitly computed from (10)—in fact recall from Section 2.2 that the mass matrix $m$ is lumped—and the time integration procedure may go on.

In eqs. (16) the quantity $D$ is a square symmetric matrix, called the matrix of connections, and $w$ is a vector. Both $D$ and $w$ are known, as appears from their definitions, but in general they must be evaluated at each step, since the coefficients $C$ and $b$ usually vary with time.
Remark 1. One might wonder why it is chosen to impose the constraint (11) on the mid-step velocity $v^{n+3/2}$ rather than, say, on the full-step velocity $v^{n+1}$:

$$Cu^{n+1} = b.$$  \hspace{1cm} (17)

The reason is that the fundamental quantity of the CD time integration scheme is the mid-step velocity as given by (5). Note also, incidentally, that using the full-step velocity would be inconsistent at the zero-step (initial time) of the transient calculation i.e. for $v^{0} = v^0$. Initial velocities $v^0$ may in fact be freely prescribed and should not be modified by the time integration algorithm. Results are usually quite similar for permanent constraints such as node blockages. However, as shown in Fig. 8 for a simple academic 1D bar impact test case, the algorithm based upon eq. (11) gives much smoother numerical results than eq. (17) in the presence of non-permanent (e.g. contact) constraints.

4 CONTACT DETECTION BY PINBALLS

The pinball algorithm in its basic form was first introduced by Belytschko and Neal in references [4] and [6]. The major goal with respect to the more conventional slide line and surface techniques was to simplify interpenetration checks, eliminate many conditional branches and thus provide a readily vectorizable procedure, well suited for some computer architectures in use at that time. The original target was the simulation of high-speed impact and penetration phenomena, in conjunction with suitable element erosion procedures.

4.1 Basic pinball method

According to [6], the main idea of the pinball algorithm is to enforce the impenetrability condition and to define the interpenetration via a set of spheres, or pinballs, embedded in the finite elements (just one pinball per element) as shown in Fig. 6a. Interpenetration between the contacting bodies is approximated by interpenetration between the pinballs.

In [6] the pinball centre is the average of its element’s nodes while the radius is such that the pinball volume equals the element volume (equivalent radius). Alternatively, one might use a larger radius, encompassing all the element’s nodes (encompassing radius). While the center is evaluated at every time step, the radius is kept constant, thus assuming that element deformation is not too large and occurs (plastically) at constant volume.

If $R_1$, $R_2$ are the radii of two pinballs, and $d_{12}$ the distance between their centers $C_1$, $C_2$, then contact (or more precisely interpenetration) has occurred when:

$$d_{12} = \|C_1 - C_2\| < R_1 + R_2$$ \hspace{1cm} (18)

where $\| \|$ designates the length of a vector.

Once detected the contact, one must introduce suitable contact forces preventing (further) interpenetration of the contacting bodies. Ref. [6] presents two implementations, one based upon penalty methods, which is also retained in subsequent work [7], and the other based on Lagrange multipliers. Whatever implementation is chosen, contact forces should ensure that:

$$\left( v_A - v_B \right) \cdot \hat{n} \leq 0$$ \hspace{1cm} (19)

where $v_A$, $v_B$ are the velocities of the two bodies at some contact point(s), and $\hat{n}$ represents a suitable normal direction to the contact surface (hats stand for unit vectors). The inequality sign accounts for possible rebound (see below). To complete the algorithm one must choose how to compute the contact point(s), i.e. how to express $v_A$, $v_B$ in terms of the nodal velocities of the elements containing the impacting pinballs, and how to compute the normal $\hat{n}$.
In ref. [6] two alternatives are considered for the contact point(s): either the pinballs centres (corresponding to the associated elements’ centroids), or the centres of the contacting element faces. In the first case, the contact force is equally distributed over the 8 nodes of the hexahedral elements used in [6] while in the second case the force is equally distributed over the 4 nodes of the contacting face. How to determine the contacting face is not specified.

As concerns the normal \( \hat{n} \), the following expression is proposed in [6]:

\[
\hat{n} = \frac{(\hat{n}_2 - \hat{n}_1)}{\|\hat{n}_2 - \hat{n}_1\|},
\]

where \( \hat{n}_1 \) and \( \hat{n}_2 \) are “the” normals associated with the two contacting pinballs. Thus, in this approach each pinball must have a unique, well-defined normal. According to [6] this normal is defined by the so-called assembled surface normal algorithm of Belytschko and Lin [3], which assembles an approximate normal to outside surfaces. The normal is non-zero only on outside surface nodes, and pinballs are placed only in elements with at least one node with a non-zero normal, see Fig. 6b.

Unfortunately, it is not specified how to pass from the assembled normals at nodes to the normals at pinballs, which are the ones needed in expression (20). Maybe simple averaging was used in [6] and it is possible that the actual expression adopted is not too important for the perforation calculations with erosion considered in that reference. As observed by the authors, in any case the piecewise spherical contact surface resulting from pinballs is much smoother than the piecewise linear irregular surface resulting from element erosion.

Although eq. (20) works relatively well for bulky continuum-like bodies, the same may not be said for thin shell or beam elements. As a matter of fact, in the hierarchic pinball algorithm [7], which deals also with shell elements, Belytschko and Yeh use a simpler approach: the contact force (they use a penalty approach) is always exerted in the direction joining the pinball (or descendent pinball, in case of hierarchy) centers, i.e.:

\[
\hat{n} = \frac{(C_1 - C_2)}{\|C_1 - C_2\|} = \frac{(C_1 - C_2)}{d_{12}}.
\]

Expressions (20) or (21) are adequate for problems where sliding and friction are not crucial, such as penetration and crashworthiness. However, in the “smooth” sliding of two relatively flat bodies, spurious oscillations may occur in the contact normal and thus in the direction of contact forces, due to the pinballs curvature. These effects were not studied in [6].

Contact detection by the basic pinball algorithm is extremely robust (conservative) and avoids all pathological situations discussed in Section 1. However, spatial resolution is clearly insufficient in cases of large practical importance (see Fig. 7), i.e.: a) with very irregular or distorted (as a consequence of large deformation) continuum elements; and b) with beam, plate or shell elements having small (or zero) topological thickness.

4.2 Hierarchic pinball method

Belytschko and co-workers recognized this deficiency and proposed an improved algorithm, based on a (hierarchic) splitting procedure, see [7]. The idea is quite simple: to improve spatial resolution of contact, a hierarchy of pinballs is constructed. A unique pinball is first and constantly associated with each element like in the basic algorithm. These are the so-called parent, or 0-level pinballs. Whenever two such pinballs are found to interpenetrate, they are (recursively) split into smaller (higher-level) descendent pinballs, which are further checked for penetration until either penetration is no longer detected or some prescribed maximum level (or minimum size) of these descendent pinballs is reached. Normally the size of pinballs is roughly halved at each splitting operation.
Thus the interpenetration of parent pinballs becomes a necessary, but not sufficient, condition for contact. It might therefore be safe and convenient to take both the parent pinballs and the descendents up to the fore-last level of the hierarchy as large as needed to ensure detection of all contacts (“encompassing” radius as described in Section 4.1). For the last level of the hierarchy an “equivalent” radius in the sense of Section 4.1 is probably a better choice.

Fig. 8a illustrates an example of hierarchic pinball generation in 2D for a regular quadrilateral element. The parent (zero-level) pinball has the same volume as the element. This pinball is subdivided into 1-level pinballs, each encompassing one fourth of the original element (represented by the dotted lines). These are further subdivided into 2-level pinballs and so on. Note that only “external” descendent pinballs, i.e. belonging to the external parts (faces $F_2$, $F_3$), of the element, need be generated, because interpenetration necessarily initiates in these parts of the contacting bodies. In addition, sub-pinballs are (recursively) generated only from pinballs that are found to be actually in contact (in Fig. 8 we assume that all “outer” pinballs are contacting an external body, not shown in the figure for simplicity).

A compromise should be found between two opposite requirements: pinballs should be kept small to increase spatial resolution of contact detection, but not too small, in order to limit the reduction of explicit integration time steps (see Section 7). Fig. 8b illustrates the hierarchic pinball splitting for a shell element with zero topological thickness, an application of large practical importance. In this case there is a physical criterion which indicates (or rather limits) the minimum size of descendent pinballs and thus the maximum depth of the hierarchy: the final pinball diameter should be of the same order as the physical thickness of the associated elements. This accounts automatically—within the contact algorithm itself—for the shell thickness which is not represented topologically, a difficult task with slide surface contact-impact methods. Fig. 8c illustrates the hierarchic pinball splitting for a 3D hexahedron element. One may proceed similarly for the other element shapes. In the present work, the whole family of linear-displacement finite elements is considered.

### 4.3 Implementation

The hierarchic pinball algorithm has been implemented in the explicit FE code. The basic algorithm can be obtained as a special case, by specifying zero depth of the hierarchy. By default, contact enforcement is achieved by the implicit Lagrange multiplier technique of Section 3.1, applied to the constraints (19). An alternative fully explicit penalty-based implementation is under testing.

### 5 TREATMENT OF REBOUND

In contact-impact one must deal with rebound, i.e. with the unilateral nature of contact conditions. The contacting bodies must be free to detach whenever appropriate conditions are met. In impact problems, rebound occurs when a tensile stress wave—e.g. generated by reflection of the main compressive impact wave at a free boundary—reaches the contact interface.

Consider first the simple case of a single contact constraint resulting e.g. from localized contact between two bodies. A possible strategy is to provisionally impose the constraint (19) with the equals sign, i.e. as a bilateral constraint, compute the corresponding Lagrange multiplier and reaction force, and then retain this force only if it tends to keep the two bodies apart. This technique, that one might denote *a posteriori* treatment of rebound, works well e.g. with simple node-to-node contact algorithms, because these indeed produce mutually independent constraints. However, it is not adequate in general with slide surface techniques and perhaps even more with hierarchic pinball methods, because the resulting constraints are (sometimes heavily) coupled, e.g. in situations of flat contact.
Thus, in the general case the problem of rebound would in principle require a fully implicit (e.g. iterative) solution. However, this is hard to justify in the chosen computational scheme which, as shown above, is fully explicit except for the solution of the constraints system (16). Therefore, an approximate explicit \textit{a priori} treatment of rebound is adopted. As an indication of incipient rebound we consider the sign of the rate of change \( \delta \) of the (oriented) distance \( \delta \) between centers of contacting (sub-)pinballs \( A \) and \( B \), see Fig. 9. This is defined as:

\[
\delta = (B - A) \cdot n_{AB} \quad \text{with} \quad n_{AB} = (B - A) / \|B - A\| \tag{22}
\]

and is a positive scalar in the current configuration, i.e. time station \( n + 1 \) in Fig. 9a. To estimate the rate of change of \( \delta \) we compute the “free” positions of pinball centers at the next time station \( n + 2 \), i.e. by neglecting the pinball contact forces (and any other constraints):

\[
A^* = A + v_A^* \Delta t = A + v_A^{n+1/2} \Delta t + a_A^* (\Delta t)^2 \tag{23}
\]

where \( v_A^* \) is the “free” velocity at \( n + 3/2 \) and \( a_A^* \) the “free” acceleration at \( n + 1 \), which can be computed via eq. (3) or (4) by neglecting all constraints. Thus in analogy with (22) the “free” inter-centers distance at \( n + 2 \) would be:

\[
\delta^* = (B^* - A^*) \cdot n_{AB}^* \quad \text{with} \quad n_{AB}^* = (B^* - A^*) / \|B^* - A^*\| \tag{24}
\]

This is again a positive scalar quantity, which may not be directly compared with \( \delta \) because the orientation of \( n_{AB}^* \) is different in general from that of \( n_{AB} \), see Fig. 9. Therefore, we compute first the scalar product \( s = n_{AB}^* \cdot n_{AB} \), and then:

\[
\delta^{**} = \delta^* \text{sign}(s) \tag{25}
\]

This guarantees that, in case the \( A \) pinball would “overtake” the \( B \) pinball in the next free configuration, a consistently signed (negative) value is obtained for \( \delta^{**} \). Finally, the rate of change of inter-centers distance is estimated by:

\[
\dot{\delta} = (\delta^{**} - \delta) / \Delta t \tag{26}
\]

The pinball contact constraint under consideration is retained only if \( \dot{\delta} < 0 \).

6 ELIMINATION OF REDUNDANT CONSTRAINTS

One major difficulty that has emerged during the implementation of the hierarchic pinball method is the onset of redundant contact conditions. This fact had not been investigated (nor mentioned) in [7], possibly because Belytschko and Yeh adopt a penalty method to compute the contact forces. However, in conjunction with the Lagrange multipliers method chosen here, the presence of redundant constraints renders the connections matrix \( D \) of (16) singular, and thus the solution of the system becomes impossible or at least more laborious.

For example, consider Fig. 10a. Two elements \( A \) and \( B \) come into “flat” contact, i.e. along a whole side. At level 0 (parent pinballs) there is just one contact constraint, which may be written e.g. by linking the velocities at pinball centers (such constraints are represented by thick segments in Figs. 10 to 13). Passing to level 1, we obtain two constraints. In both cases, no redundancies are observed. However, at level 2 of the hierarchy four constraints would be obtained, of which two are redundant, i.e. the \( D \) matrix is twice singular. The figure presents also the case corresponding to an intermediate level 1.5 whereby three constraints would be obtained, of which one is redundant. This case is fictitious because the actual hierarchic splitting procedure generates only “integer” levels, but it is useful for illustration purposes.
A qualitative explanation of redundancies is that when several contacts are detected between descendent pinballs sharing the same couple of parents, i.e. associated with the same couple of finite elements, the corresponding constraints involve the same set of nodes, i.e. the same degrees of freedom. Intuitively, only a limited number of constraints may be written independently, more precisely one constraint for each node of a contacting face. This means at most two constraints in 2D, and three or four constraints in 3D, when using linear-displacement continuum FE. Any extra constraints are redundant and render the matrix singular. This explains why redundancies are observed neither with the basic pinball method, nor with slide surface techniques. In the former case, there is at most one constraint between each FE couple, while in the latter case at most one constraint is written for each slave node.

Fig. 10b illustrates another example of redundant constraints, due to flat contact between two neighboring element couples (this may be generalized to the case of two long flat bodies in smooth contact). Here we obtain one redundancy already at hierarchy level 1, due to neighboring elements contributions to the common contacting nodes.

6.1 The “Common Normal” algorithm

It is of course desirable to eliminate redundant constraints before attempting to solve the system (16), both for efficiency and for accuracy reasons. Many redundancies may be removed by applying the criterion mentioned above which limits the number of constraints to be retained for each contacting element couple. Fig. 11 illustrates the process in the simple 2D case of Fig. 10a. Assuming a hierarchy level 1.5 (just one redundant constraint) for simplicity, there are three alternative ways of removing the singularity, as sketched in the figure, i.e. by removing the right, the central or the left constraint. Analytically, results are indeed equivalent. However, numerically it is easily verified that removing the central constraint is the best solution, since the resulting constraints matrix is better conditioned. Note that, however, this technique does not remove all redundancies: e.g., it has no effect in the case of Fig. 10b.

The search for redundant constraints according to the technique described above is a computational burden specific to hierarchic pinballs, but it may be exploited to improve model behavior in situations such as those depicted in Fig. 12. Case 1 illustrates the perfect, ideal flat contact between two elements with level 2 pinballs. Of the four detected contacts two are redundant, so the above algorithm would retain only the two “external” contacts as indicated by the thick segments. Assuming that contact constraints are written along the lines joining the pinballs centres, one would obtain the two normal directions $\hat{n}_1$ and $\hat{n}_4$, which in this case are coincident and perpendicular to “the” common interface of the two contacting bodies.

However, when any imperfections are present as shown in Cases 2 to 4 in Fig. 12, due e.g. to misalignment, size mismatch or deformation, one would end up with normals that are no longer coincident (Cases 3 and 4) and anyway are not perpendicular to what may be considered as the “common interface” of the two contacting bodies. This is contrary to physical intuition, and the resulting contact force components tangential to the interface, although usually small, would produce spurious sticking or friction-like effects that act against the free relative sliding of the two bodies (inviscid contact is assumed here). For large relative sliding, an oscillating spurious tangential force would result, qualitatively indicated in the inset of Fig. 13, whose amplitude and period would depend on the size of the contacting pinballs.

In all four cases illustrated in Fig. 12, the desired behavior is that of Case 1, i.e. only the two “external” constraints should be retained, and these should have coincident normals $\hat{n}_1 \equiv \hat{n}_4$, both perpendicular to “the” contact interface. This may be obtained by the following “Common Normal” (CN) algorithm (see Fig. 13). Considering the case of 2D continuum elements for simplicity (the 3D situation is analogous but is not presented for brevity) one must
accept at most two contacts between descendent pinballs derived from the same couple of parents. If there are more than two, we construct the “mid-points” $M_i$ defined as the central points of the common zones (grayed in Fig. 13) of segments joining the sub-pinball centers. For each couple of mid-points $i$, $j$ we compute their distance $d_{ij}$. We retain only the two extreme contacts $I$, $J$ such that $d_{ij} = \max(d_{ij})$. Both contacts are assigned the same (common) normal $\hat{n}_c$, given by the direction perpendicular to segment $M_iM_j$.

If one (at least) of the contacting elements is a material point (1-node particle) or a bar/beam element (2-node segment), the above CN algorithm requires some simple adaptations, not discussed here for brevity. Interested readers may find further details in ref. [11].

6.2 Improving contact points by constraint collapse technique

As mentioned in Section 4.1 and in ref. [6], the pinball method offers a certain freedom as concerns the points at which discrete contact conditions are imposed. In the present implementation, by default, such points are the pinball (or sub-pinball) centers. However, in principle any point of the pinball may be chosen, since contact locations in the discrete model are determined only with a certain approximation, of the order of the (sub-)pinball radius.

When writing down the constraints (19), if the assumed discrete contact points $A$, $B$ are internal to the parent element, then resulting expressions via element shape functions involve all nodes (dofs) of the element, thus leading to relatively complex and coupled constraints. This is not the case with slide surface contact algorithms, whereby at most one constraint for each node of the slave surface is generated, thus avoiding any redundancies.

To simplify the constraints and to reduce the risk of redundancies in the hierarchic pinball method it seems therefore better, whenever appropriate, to choose discrete contact points that lie on the outer surface of the parent element or that, even better, coincide with a node of the parent. In the first case the constraint involves just the nodes of the contacting face, in the second just the contacting node. In this way, the expressions obtained from pinballs are quite similar to those that would result from node-to-surface contact algorithms.

In the following, we will refer to this technique as constraint “collapsing” on the nodes (or on the element surface), or simply as Nodal Collapse (NC). It is therefore convenient to classify descendent pinballs in three categories, as shown in Fig. 14: corner pinballs (close to nodes), side pinballs (along element sides) or face pinballs (3D only). With this nomenclature, the NC algorithm is as follows:

- **NC Rule 1.** For continuum elements: a corner pinball collapses onto the nearest node, a side pinball collapses onto the nearest point along the side and a face pinball (3D only) collapses onto the nearest point on the face.

- **NC Rule 2.** For 2D/3D beam elements and 2D shells (which have 2 nodes and 2 faces): a corner pinball collapses onto the nearest node, a side pinball does not collapse at all because its center lies already on the side. Furthermore, there are no face pinballs.

- **NC Rule 3.** For 3D shell elements (which have 3 or 4 nodes and 2 faces): a corner pinball collapses onto the nearest node, a side pinball collapses onto the nearest point along the side and a face pinball does not collapse at all because its center lies already on the face.

For example, consider the simple flat contact between two quadrilateral elements depicted in Fig. 15. The left part of the Figure shows the four “raw” contacts detected by a hierarchic pinball method of the second order (level 2 descendents). By applying first the CN algorithm of Section 6.1, as shown in the central part of the Figure, only the two extreme contacts ($C_1$, $C_4$) are retained, both acting along the common normal direction $\hat{n}_c$. Then, according to the NC algorithm, the contact points collapse onto the nodes, see the right part of the Figure,
since all involved sub-pinballs are corner pinballs. The normal directions are not affected by
this procedure and remain those resulting from the CN algorithm ($\hat{\mathbf{n}}_i$).

The two resulting constraints are thus completely independent from each other, since each
of them involves just one node of each element. The resulting constraints coincide in this case
with those obtained from a node-to-node contact method, which seems quite appropriate in
the present example.

As another example of the beneficial effects of the combined CN/NC algorithms, consider
the misaligned flat contact case of Fig. 16. Of the four sub-pinballs which remain after the CN
procedure, two collapse on the nodes as before, while the other two collapse on the element
side. The resulting constraints are again very similar to those obtained from a slave node /
master face technique, the only difference being that the discrete contact points on the two
facing elements do not coincide perfectly in general. However, the maximum distance be-
tween such points is of the order of the contact resolution (sub-pinball radius) and may ther-
fore, at least in principle, be rendered as small as desired by increasing the hierarchy depth.

6.3 Eliminating residual redundancies

As already mentioned, the CN algorithm of Section 6.1 may eliminate most, but in general not
all, the redundancies which result from straightforward application of a hierarchic pinball
method in flat contact conditions. Therefore, after application of the CN and of the NC algo-
rithms, a final Residual Constraint (RC) elimination algorithm may be needed.

Consider for example the simple case of 2D flat aligned contact between two couple of el-
ements, shown in Fig. 10b. Assume a level 2 hierarchy, so that there are 8 “raw” contacts be-
tween descendent pinballs as indicated in Fig. 17a. After application of the CN algorithm,
only the four constraints $C_1, C_4, C_5, C_8$ remain, see Fig. 17b. All contacting sub-pinballs are
corner pinballs, so that the NC algorithm gives the situation of Fig. 17c. It is clear that of the
four remaining constraints, one is redundant. More precisely, $C_4$ and $C_5$ are the same con-
straint and thus one of the two must be eliminated. The final result, shown in Fig. 17d, is iden-
tical to the one that would result from a node-to-node contact algorithm or from a slide
surface algorithm: three node-to-node constraints remain, all having the same normal in this
simple ideal case.

Identification of the redundancy is simple in this case because the two constraints $C_4$ and
$C_5$ are of the same type (node-to-node constraints) and involve the same couple of nodes. Of
course, in geometrically more complex situations and especially in 3D detecting all the re-
dundancies may become a more complicated task. An empirical procedure (RC algorithm) has
been set up which treats all possible cases by first subdividing the constraints into groups of
node-to-node constraints (like in the simple example considered), or node-to-point con-
straints, or point-to-point constraints. The full procedure may not be listed here for brevity,
but interested readers may find all details in ref. [11].

6.4 Résumé of the proposed hierarchic pinball algorithm

It is perhaps worthwhile to give a short résumé of the general-purpose hierarchic pinball algo-
rithm. It consists of the following phases:

- Detect all “raw” contacts between sub-pinballs, as described in Section 4.2.
- Apply the common normal (CN) algorithm of Section 6.1. This produces “better” normals
in flat-like contacts and eliminates most redundancies.
- Collapse each constraint onto the nearest node (NC), side or face, as described in Section
6.2. This simplifies the form of the constraints, rendering them more similar to those that
would result from more traditional slave node / master surface algorithms. It also facilitates the following final removal of surviving redundancies.

- Eliminate any surviving redundant constraints (RC), as described in Section 6.3.
- Apply the a priori rebound detection algorithm, as described in Section 5, and reject any rebounding constraints.
- Add the remaining contact constraints to the other links, solve the linear system and compute the contact forces, as described in Section 3.1.

The algorithm requires no tuning parameters, unlike e.g. penalty-based methods, and has been applied successfully in a large variety of contact situations. Since the elimination of redundant constraints is a relatively expensive operation, the user may decide whether or not to activate it. This is actually necessary only in cases where flat contact may occur.

7 STEP CONTROL AND CONTACT SEARCH OPTIMIZATION

In explicit schemes such as the one adopted here the size of the time increment $\Delta t$ is governed by the stability (Courant) condition:

$$\Delta t = \varphi \Delta t^{\text{stab}} = \varphi \Delta x/c$$

(27)

where $\Delta t^{\text{stab}}$ is the stability step which, in sub-sonic problems, may be estimated as the smallest element length $\Delta x$ divided by the sound speed $c$, and $\varphi$ is a safety factor ($\varphi < 1$).

The pinball algorithm introduces additional constraints on $\Delta t$. In fact, for the method to work properly, pinball penetration must be detected before it exceeds a limit value, namely before the centers of the two impacting pinballs lie on opposite sides with respect to their positions just before impact. Consider the example in Fig. 18a. Let $v_A$, $v_B$ represent the pinball velocities just before contact, say at time station $n+1$. Then, if the chosen time increment $\Delta t_2 = t^{n+2} - t^{n+1}$ is too large, at the following time station ($n+2$) one might obtain the situation shown in Fig. 18b: the centre of pinball $A$ has “overtaken” the centre of pinball $B$ with respect to the positions they occupied at the previous step. In other words, the pinballs unit normal $\mathbf{n}_{AB}$ (oriented segment joining the two centres) has “changed sign” in just one step. In configuration $n+1$ no penetration occurs and thus no contact forces are generated. In configuration $n+2$ there is penetration, but the two pinballs appear to be detaching from each other (apparent rebound) so that, again, no contact forces are generated. In conclusion, pinball $A$ “passes” undisturbed across pinball $B$.

This situation is unlikely to occur, at least in sub-sonic problems, in the basic pinball algorithm because pinballs are relatively large. However, it becomes more and more likely in hierarchic methods as the size of descendents decreases, unless appropriate measures are taken. Such time step limitations are needed not only with pinballs, but also in slide surface algorithms. The advantage of the pinball method is that it offers a natural and effective way of computing step limitations as part of the contact checking algorithm itself. In fact, these may be obtained at little extra cost along with normal penetration checks. For each couple of parent pinballs $A$, $B$ eq. (18) is checked, i.e. $d_{AB} < (R_A + R_B)$. At the same time, we compute the (oriented) relative velocity of the two pinballs:

$$v_{AB} = (v_A - v_B) \cdot \mathbf{n}_{AB}$$

(28)

and then, if this quantity is positive ($v_{AB} > 0$, i.e. the two pinballs are approaching each other) we impose the following limitation over the time increments of both elements $A$ and $B$:

$$\Delta t_2 = t^{n+2} - t^{n+1} < \left( d_{AB} / v_{AB} \right)$$

(29)
Keeping the pinballs relatively large in size (encompassing radius) down to the fore-last of hierarchy level ensures not only safe contact capturing, but also early detection of time increment limitations and thus effective and gradual step reduction before actual contact occurs.

Another important practical aspect is optimization of contact search operations. Checking each (parent) pinball against each other is an $O(N^2)$ algorithm which becomes prohibitive for large numbers of parent pinballs $N$. Grouping pinballs into (user-defined) “sets” each representing a separate body helps, because distance checks are skipped for pinball couples belonging to the same set, but this technique may not be applied to self-contact problems. In the present implementation a standard search optimization algorithm based on bucket sorting technique is used. Space is subdivided into a regular grid of cells and pinballs in each cell are checked only against those in the same cell or in direct neighbor cells.

8 Extension to Domain Decomposition and to Spatial Partitioning
(To be written).

9 Numerical Examples

9.1 Cable wrapping

This test involves two elastic cables (see Fig. 19), discretized by 20 and 4 two-node cable-like elements, respectively. These elements react in traction, but not in compression (nor in bending). The shorter cable is fixed at both ends, while the longer cable is fixed at one end and has a 100 kg mass attached at the other end, represented by a material point. The mass has an initial velocity of 100 m/s and this produces the wrapping of the long cable around the short one.

One parent pinball is embedded in each of the cable elements, see Fig. 19a. A hierarchy of level 2 with CN algorithm is used to detect the contacts. First contact occurs at 56 ms and the wrapping process terminates at 833 ms, after 4 complete turns. At this time, due to elastic oscillations of the cables, some rebound (un-winding) starts, see final result at 900 ms. The star-shaped structure appearing in the zoomed-up picture is nothing else than the (coarse) discrete cable elements: obviously a 2-node-element may not “bend” upon itself. This problem shows the versatility of pinball contact detection and would be very difficult if not impossible to treat with node-to-surface techniques (cables have no surface). Another added value of the pinball method is that the contacts themselves (i.e. the contacting descendents and the assumed contact points) are easily visualized for inspection, as shown in the pictures of Fig. 19.

Note that two “bodies” are declared in this problem, each one corresponding to one cable. Self-contact, i.e. contact between (sub-)pinballs belonging to the same body, is not activated. This is why the long cable wraps up by remaining strictly in the $x-z$ plane: the cable may not impact upon itself. Moreover, no pinball is attached to the lumped mass.

9.2 Sphere indentation

This test simulates displacement-driven penetration of a rigid spherical indenter of radius $R$ into a half-space of elastic perfectly-plastic metal, see Fig. 20. An approximate static analytical solution is given by Johnson in [12], assuming that contact pressure $p_m = 3\sigma_y$ is constant and that the contact zone radius $r = \sqrt{(R\delta/0.368)}$, $\delta$ being the penetration depth. The resulting approximated penetration force is thus linear with $\delta$.

In the numerical simulation, only a cylinder of radius and height $3R$ is modelled instead of the full half space. The rigid indenter is represented by a material point with an associated parent pinball of radius $R$. This pinball is never splitted. One fourth of the cylinder (by symmetry) is discretized with 8192 cube elements and 360 prism elements (on the axis). Hierar-
chic pinballs of level 4 with CN and NC algorithms are associated with the relevant surface of the cylinder. A linear displacement in time is imposed to the penetrator, from 0 to $R$ in 50 ms. Although this is not static loading, it has been verified that dynamic effects are negligible. The computed penetration force agrees well with the approximate linear analytical solution, see Fig. 20b. The final shape of the solid is shown in Fig. 20c for the cylindrical 3D model, and in Fig. 20d for a cubes-only model, which is not exactly axisymmetric. Colors indicate the displacement norm. (More examples to be added here)

10 SUMMARY AND CONCLUSIONS

(To be written).

REFERENCES


Table I. Uniform-step CD explicit time integration algorithm (Lagrangian description).

<table>
<thead>
<tr>
<th>Algorithm UCD</th>
</tr>
</thead>
<tbody>
<tr>
<td>0. Set initial conditions: $n \leftarrow 0$ (step counter), $t \leftarrow t^0$, $x \leftarrow x_0$, $\sigma \leftarrow \sigma_0$, $v \leftarrow v_0$, $W^{\text{int}} \leftarrow 0$, $W^{\text{ext}} \leftarrow W^{\text{kin}}$</td>
</tr>
<tr>
<td>1. GO TO 4.</td>
</tr>
<tr>
<td>2. End of the calculation.</td>
</tr>
</tbody>
</table>

Table III. Numerical solutions for the 1D step wave propagation.
<table>
<thead>
<tr>
<th>Solution</th>
<th>Mesh/Elements</th>
<th>Time step</th>
<th>Steps</th>
<th>Cycles</th>
<th>Max level frequency</th>
<th>Elements × cycles</th>
<th>CPU (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step wave 1</td>
<td>Uniform/100</td>
<td>Uniform</td>
<td>250</td>
<td>—</td>
<td>— 2.5 × 10⁷</td>
<td>—</td>
<td>0.2</td>
</tr>
<tr>
<td>Step wave 4</td>
<td>Graded/111</td>
<td>Partition</td>
<td>252</td>
<td>2.016</td>
<td>8 4.7 × 10⁶</td>
<td>—</td>
<td>0.3</td>
</tr>
</tbody>
</table>

Figure 1. Crash of a metallic tube, simulation and experiment (Courtesy of CEA)

Figure 2. Simulation of bird strike on a jet fan by the SPH method (Courtesy of Snecma/CEA)

Figure 3. Examples of pathological contact detection by the sliding surface method (from [7])

Figure 4. Further examples of pathological contact detection cases (from [8])
Figure 5. Bar impact test by the basic pinball method

- a) FE mesh with initial velocities and pinballs shown
- b) contact force obtained with the first algorithm
- c) contact force obtained with the second algorithm

Figure 6. The basic pinball method (from [6])

- a) the pinball concept shown in 2D
- b) assembled surface normals
- c) interpenetrating pinballs

Figure 7. Shortcomings (incorrect contact detection) of the basic pinball method

- a) slender or distorted continuum elements
- b) beam or shell elements

Mesh node
- Other geometric point
Figure 8. The hierarchic pinball method

(a) Pinball splitting for a 2D quadrilateral

(b) Pinball splitting for a 2D beam or shell

c) Pinball splitting for a 3D hexahedron

Figure 9. A priori detection of rebound

(a) Configuration at time station \( n + 1 \)

(b) “Free” configuration at time station \( n + 2 \)

Figure 10. Example of redundant constraints generated by the hierarchic pinball method

(a) “Flat” contact between a couple of elements

(b) “Flat” contact between two element couples
Folco Casadei and Vincent Faucher

Figure 11. Removing redundant constraints

Figure 12. Examples of redundant constraints in flat contacts with hierarchic pinballs

Figure 13. The common normal algorithm in a 2D case

Figure 14. Classification of descendent pinballs
Figure 15. Nodal collapse of corner descendents in perfectly aligned flat contact

Figure 16. Nodal collapse of corner and side descendents in largely misaligned flat contact
Figure 17. Redundancy elimination in perfectly aligned multiple flat contact

Figure 18. Adaptation of the time step

Figure 19. Cable wrapping example
Folco Casadei and Vincent Faucher

Figure 20. Sphere indentation example

a) geometry and material

\[ R = 0.5 \text{ m} \]
\[ E = 2.1 \times 10^{11} \text{ Pa} \]
\[ v = 0.3 \]
\[ \sigma_T = 5.0 \times 10^7 \text{ Pa} \]
\[ E_P = 0 \]

b) comparison of numerical and analytical solutions

Approximate analytical solution

2D axisymmetric solution

3D axisymmetric solution

3D hexahedral mesh solution

TEST - 14
TIME: 5.000000E-02 STEP: 79300

INDE - 13
Time: 5.000000E-02 Step: 10562

Max: +5.21E-04

Min: +1.00E-04

Displacement norm [m]

c) final deformation

d) solution with all-hexahedral mesh
9. Appendix C - Input Files

Sample input files

This Section contains, in alphabetical file order, the listings of all input files related to the examples asnc01.epx

**asnc01.epx**

```
ASNC01
ECHO

CONV win
CPFA LAG

GEM LSR PON 12 CAR 4 TERM
8 0 1 2 0 1 1 2 1 2 2 2
1 1 1 2 1 0 2 1 2 2
1 2 2 1
3 4 5
7 9 11 11

COMP COUL VERT LECT tous TERM
MATE LINE RO 8000.0 YOUN 1.E12 NU 0.0
COMP COUL VERT LECT tous TERM

VITE 2 -50 LECT 7 PAS 1 12 TERM
MATE LINE RO 8000.0 YOUN 1.E12 NU 0.0

COMP COUL VERT LECT tous TERM
MATE LINE RO 8000.0 YOUN 1.E12 NU 0.0
COMP COUL VERT LECT tous TERM

CPLA LAGR
ECHO

ECRI COOR DEPL VITE ACCE FINT FEXT FLIA CONT ECRO FREQ 1

LINK DECO PINB PENA SFAC 1.0
FACE HFRO

PINB DASN
FACE HFRO

PINB NASN PASN NORM
FACE HFRO

PINB CDES JOIN
FACE HFRO

FEW 891 21 1
781 11 0
2365
1254
01 11 21 02 12 22
00 10 20 01 11 21
00 10 11 00 10 20
23 6
0 23 6
0
5 1 1
0

CONV win

LOG 1
CSTA 0.5E0

CPLA LAGR
ECHO

ECRI COOR DEPL VITE ACCE FINT FEXT FLIA CONT ECRO FREQ 1

LINK DECO PINB PENA SFAC 1.0
FACE HFRO

PINB DASN
FACE HFRO

PINB NASN PASN NORM
FACE HFRO

PINB CDES JOIN
FACE HFRO
```

**asnc02.epx**

```
ASNC02
ECHO

CONV win
CPFA LAG

GEM LSR PON 12 CAR 4 TERM
8 0 1 2 0 1 1 2 1 2 2 2
1 1 1 2 1 0 2 1 2 2
1 2 2 1
3 4 5
7 9 11 11

COMP COUL VERT LECT tous TERM
MATE LINE RO 8000.0 YOUN 1.E12 NU 0.0
COMP COUL VERT LECT tous TERM

VITE 2 -50 LECT 7 PAS 1 12 TERM
MATE LINE RO 8000.0 YOUN 1.E12 NU 0.0

COMP COUL VERT LECT tous TERM
MATE LINE RO 8000.0 YOUN 1.E12 NU 0.0
COMP COUL VERT LECT tous TERM

CPLA LAGR
ECHO

ECRI COOR DEPL VITE ACCE FINT FEXT FLIA CONT ECRO FREQ 1

LINK DECO PINB PENA SFAC 1.0
FACE HFRO

PINB DASN
FACE HFRO

PINB NASN PASN NORM
FACE HFRO

PINB CDES JOIN
FACE HFRO
```

**asnc03.epx**

```
ASNC03
ECHO

CONV win
CPFA LAG

GEM LSR PON 12 CAR 4 TERM
8 0 1 2 0 1 1 2 1 2 2 2
1 1 1 2 1 0 2 1 2 2
1 2 2 1
3 4 5
7 9 11 11

COMP COUL VERT LECT tous TERM
MATE LINE RO 8000.0 YOUN 1.E12 NU 0.0
COMP COUL VERT LECT tous TERM

VITE 2 -50 LECT 7 PAS 1 12 TERM
MATE LINE RO 8000.0 YOUN 1.E12 NU 0.0

COMP COUL VERT LECT tous TERM
MATE LINE RO 8000.0 YOUN 1.E12 NU 0.0
COMP COUL VERT LECT tous TERM

CPLA LAGR
ECHO

ECRI COOR DEPL VITE ACCE FINT FEXT FLIA CONT ECRO FREQ 1

LINK DECO PINB PENA SFAC 1.0
FACE HFRO

PINB DASN
FACE HFRO

PINB NASN PASN NORM
FACE HFRO

PINB CDES JOIN
FACE HFRO
```
bar202a.epx

bar203.epx

bar204.dgibi

Page 180
**bar204.epx**

**BAR204**

**END**

**CAST mesh**

**CPLA LASM**

**GEOM CMB mesh TERM**

**COMP COUL. VERT LECT tous TERM**

**MATE LINE BS 000.0 VUM 1.812 NU 0.0**

**COUR 8 'vx_p6' VITE COMP 1 NOEU LECT p6 TERM**

**COUR 7 'vx_p4' VITE COMP 1 NOEU LECT p4 TERM**

**COUR 5 'vx_p0' VITE COMP 1 NOEU LECT p0 TERM**

**COUR 4 'dx_p6' DEPL COMP 1 NOEU LECT p6 TERM**

**COUR 3 'dx_p4' DEPL COMP 1 NOEU LECT p4 TERM**

**COUR 2 'dx_p2' DEPL COMP 1 NOEU LECT p2 TERM**

**COUR 1 'dx_p0' DEPL COMP 1 NOEU LECT p0 TERM**

**AXTE 1.0 'Time [s]'**

**SORT GRAP**

**RESU ALIC GARD PSCR**

**ECHO**

**SUIT**

**END**

**PLAY**

**CALC TINI 0. TEND 0.4E-3**

**OPTI NOTE**

**ECRI COOR DEPL VITE ACCE FINT FEXT FLIA CONT ECRO TFRE 0.4E-3**

**INIT VITE 1 50 LECT bar1 TERM**

**LINK DECO PINB PENA SFAC 1.0**

**OPTI PINS STAT**

**MATE LINE RO 8000.0 YOUN 1.E12 NU 0.0**

**COMP COUL VERT LECT tous TERM**

**GEOM CAR1 mesh TERM**

**CAST mesh**

**ECHO**

**FIN**

---

**bar205.epx**

**BAR205**

**END**

**CAST mesh**

**CPLA LASM**

**GEOM CMB mesh TERM**

**COMP COUL. VERT LECT tous TERM**

**MATE LINE BS 000.0 VUM 1.812 NU 0.0**

**COUR 8 'vx_p6' VITE COMP 1 NOEU LECT p6 TERM**

**COUR 7 'vx_p4' VITE COMP 1 NOEU LECT p4 TERM**

**COUR 5 'vx_p0' VITE COMP 1 NOEU LECT p0 TERM**

**COUR 4 'dx_p6' DEPL COMP 1 NOEU LECT p6 TERM**

**COUR 3 'dx_p4' DEPL COMP 1 NOEU LECT p4 TERM**

**COUR 2 'dx_p2' DEPL COMP 1 NOEU LECT p2 TERM**

**COUR 1 'dx_p0' DEPL COMP 1 NOEU LECT p0 TERM**

**AXTE 1.0 'Time [s]'**

**SORT GRAP**

**RESU ALIC GARD PSCR**

**ECHO**

**SUIT**

**END**

**PLAY**

**CALC TINI 0. TEND 0.4E-3**

**OPTI NOTE**

**ECRI COOR DEPL VITE ACCE FINT FEXT FLIA CONT ECRO TFRE 0.4E-3**

**INIT VITE 1 50 LECT bar1 TERM**

**LINK DECO PINB PENA SFAC 1.0**

**OPTI PINS STAT**

**MATE LINE RO 8000.0 YOUN 1.E12 NU 0.0**

**COMP COUL VERT LECT tous TERM**

**GEOM CAR1 mesh TERM**

**CAST mesh**

**ECHO**

**FIN**

---

**bar204a.epx**

**BAR204 Post-treatment (animation from alice file)**

**END**

**RESU ALIC 'bar204a.AI' GARD PSCH**

**SORT VIEW RPT0 1**

**PLAY**

**CALC 1 EYE 0.000000E+00 2.101500E+00 1.212345E+01**

**O 0.000000E+00 0.000000E+00 0.000000E+00**

**VIEW 0.000000E+00 0.000000E+00 0.000000E+00**

**UP 0.000000E+00 1.000000E+00 0.000000E+00**

**VIEW 0.000000E+00 0.000000E+00 -1.000000E+00**

**POT 2.444601E+01**

**PEQ 2**

**GO**

**SCEN GEOM NAVI FREE**

**FICH ALIC FREQ 1**

**VITE 1 -50 LECT bar2 TERM**

**ASN**

**LECT tous TERM**

**BODY MLEV 2 LECT cond TERM**

** BODY MLEV 2 LECT cond TERM**

**INIT VITE 1 50 LECT bar2 TERM**

**ECHO**

**SCEN GEOM NAVI FREE**

**FICH AVI CONT REND**

**GO**

**END**

---

**bar205.dgibi**

**opti echo ;**

**opti zoom 2 elem qua ;**

**opti zoom form 'bar205.mesh' ;**

**opti trac per ftra 'bar205.mesh.pe' ;**

**pt = 0.0 ;**

**p1 = 0.01 ;**

**p2 = 1.0 ;**

**p3 = 1.00450 ;**

**p4 = 2.00450 ;**

**s1 = 0.0 ;**

**s2 = 1.0 ;**

**tol = 1.0 ;**

**cl = 0.0045 ;**

**bar = 2.00450 ;**

**tras tol (bars et p0 et p1 et p2 et p3 et p4 et p5 et p6 et p7) ;**

**con = bar1 elem cont p2 ;**

**mesh = bars ;**

**taxe mesh ;**

**sauv form mesh ;**

**trac qual mesh ;**

**trac qual (con1 et con2) ;**

**fin ;**

---
baru205a.epx

```
BAROIS Post-treatment (animation from alice file) ENCO RESU ALIC 'baro02_alix' GARD PCCH SORT VISU NFTO 1
*=================================================================
CALC TINI 0. TEND 0.4E-3
ECRI COOR DEPL VITE ACCE FINT FEXT FLIA CONT ECRO TFRE 0.4E-3
INIT VITE 1 50 LECT bar1 TERM
LINK DECO PINB PENA SFAC 1.0
OPTI PINS STAT
GEOM CAR1 mesh TERM
CPLA LAGR
CAST mesh
CONV win
ECHO
```

```
**=================================================================
ENDPLAY
TRAC OFFS FICH AVI CONT REND
GO
GOTR LOOP 999 OFFS FICH AVI CONT NOCL REND
FREQ 1
TRAC OFFS FICH AVI NOCL NFTO 1001 FPS 25 KFRE 10 COMP -1 REND
```

```
SLER CAM1 1 NFRA 1
SCEN GEOM NAVI FREE
! Q 1.00000E+00 0.00000E+00 0.00000E+00
CAME 1 EYE 3.00000E+00 3.20711E+00 2.90257E+01
*=================================================================
SORT VISU NSTO 1
ECHO
```

```
BAROIS Post-treatment (time curves from alice file) ENCO RESU ALIC 'baro02_alix' GARD PCCH SORT VISU NS70 1
PLAY
```

```
**=================================================================
ENDPLAY
TRAC OFFS FICH AVI CONT REND
GO
GOTR LOOP 999 OFFS FICH AVI CONT NOCL REND
FREQ 1
TRAC OFFS FICH AVI NOCL NFTO 1001 FPS 25 KFRE 10 COMP -1 REND
```

```
SLER CAM1 1 NFRA 1
SCEN GEOM NAVI FREE
! Q 1.00000E+00 0.00000E+00 0.00000E+00
CAME 1 EYE 3.00000E+00 3.20711E+00 2.90257E+01
*=================================================================
SORT VISU NSTO 1
ECHO
```

```
bar206a.epx

```
BAR206 Post-treatment (animation from alice file) ENCO RESU ALIC 'bar206_alix' GARD PCCH SORT VISU NFTO 1
***=================================================================
CALC TINI 0. TEND 0.4E-3
ECRI COOR DEPL VITE ACCE FINT FEXT FLIA CONT ECRO TFRE 0.4E-3
PLAY
```

```
**=================================================================
ENDPLAY
TRAC OFFS FICH AVI CONT REND
GO
GOTR LOOP 999 OFFS FICH AVI CONT NOCL REND
FREQ 1
TRAC OFFS FICH AVI NOCL NFTO 1001 FPS 25 KFRE 10 COMP -1 REND
```

```
SLER CAM1 1 NFRA 1
SCEN GEOM NAVI FREE
! Q 1.00000E+00 0.00000E+00 0.00000E+00
CAME 1 EYE 3.00000E+00 3.20711E+00 2.90257E+01
*=================================================================
SORT VISU NSTO 1
ECHO
```

```
carn07.epx

```
CARB07 ENCO (C038 VIN C039 VIN C03D VIN)
DIMA DIME 1 MARE 3140 TERM
GEOM q42 elem TERM
COMP R1 1 LCT 1.0E+00 TERM
MATE VM23 RO 0.00825 YOUN 197600.0 NU 0.29 ELAS 222.35
CAME 1 EYE 3.00000E+00 3.20711E+00 2.90257E+01
*=================================================================
ENDPLAY
TRAC OFFS FICH AVI CONT REND
GO
GOTR LOOP 999 OFFS FICH AVI CONT NOCL REND
FREQ 1
TRAC OFFS FICH AVI NOCL NFTO 1001 FPS 25 KFRE 10 COMP -1 REND
```

```
SLER CAM1 1 NFRA 1
SCEN GEOM NAVI FREE
! Q 1.00000E+00 0.00000E+00 0.00000E+00
CAME 1 EYE 3.00000E+00 3.20711E+00 2.90257E+01
*=================================================================
SORT VISU NSTO 1
ECHO
```

```
bar206.epx

```
BAR206 ENCO (C038 WIN C039 WIN C03D WIN)
CAST mesh
CPLA LAG
GEOM CAR1 mesh TERM
COMP C039 VERT LCT tout TERM
MATE LIME 8000 0 VOLUM 1.212 1.0 0.0
LOC 0.0
OPTI FINS STAT
ASE
LINE DEC0 FERS INERA SFAC 1.0
BODY MLVR 4 LCT cool TERM
BODY MLVR 4 LCT cool TERM
INIT VITE 1 50 LCT bar1 TERM
VITE 1 -50 LCT bar2 TERM
ENS CODE DEPL VITE ACCE FINT PEXT PLIA CONT BARO TERM TRAC 0.4E-3
ECHI CODE DEPL VITE ACCE FINT PEXT PLIA CONT BARO TERM TRAC 0.4E-3
ECHI CODE DEPL VITE ACCE FINT PEXT PLIA CONT BARO TERM TRAC 0.4E-3
ECHI CODE DEPL VITE ACCE FINT PEXT PLIA CONT BARO TERM TRAC 0.4E-3
```

```
ECHO
**=================================================================
ENDPLAY
TRAC OFFS FICH AVI CONT REND
GO
```

```
sltc00.epx

```
SLTR 0.00625 TERM
```

```
LACE 2 LACER LOG 1.0E+00 TERM
CPLA 4 LACER 0.0E+00 TERM
```

```
**=================================================================
ENDPLAY
```

cphin02.epx

19 December 2016 7:12 am

FIN DE ALIC FREQU 1
OPTI NOTE
CSTA 0.5E0
LOG 1

COLO PAPE
TEXT NODE
FOV 2.48819E+01
UP 0.00000E+00 1.00000E+00 0.00000E+00
VIEW 0.00000E+00 0.00000E+00 -1.00000E+00
RIGI 1.00000E+00 0.00000E+00 0.00000E+00

FOV 2.48819E+01
UP 0.00000E+00 1.00000E+00 0.00000E+00
VIEW 0.00000E+00 0.00000E+00 -1.00000E+00
RIGI 1.00000E+00 0.00000E+00 0.00000E+00

FOV 2.48819E+01
UP 0.00000E+00 1.00000E+00 0.00000E+00
VIEW 0.00000E+00 0.00000E+00 -1.00000E+00
RIGI 1.00000E+00 0.00000E+00 0.00000E+00

FOV 2.48819E+01
UP 0.00000E+00 1.00000E+00 0.00000E+00
VIEW 0.00000E+00 0.00000E+00 -1.00000E+00
RIGI 1.00000E+00 0.00000E+00 0.00000E+00

FOV 2.48819E+01
UP 0.00000E+00 1.00000E+00 0.00000E+00
VIEW 0.00000E+00 0.00000E+00 -1.00000E+00
RIGI 1.00000E+00 0.00000E+00 0.00000E+00

FOV 2.48819E+01
UP 0.00000E+00 1.00000E+00 0.00000E+00
VIEW 0.00000E+00 0.00000E+00 -1.00000E+00
RIGI 1.00000E+00 0.00000E+00 0.00000E+00
equal = 3.00000E+00 3.20711E+00 2.90257E+01
CAME 1 EYE 3.00000E+00 3.20711E+00 2.90257E+01
PLAY
*=================================================================
SORT VISU NSTO 1
ECHO
DRO201 Post-treatment (animation from alice file)
dro201a.epx
SCEN GEOM NAVI FREE
FREQ 2
! Q 1.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
CAME 1 EYE 5.00000E-01 2.21000E+00 1.21286E+01
PLAY
fin
CALCUL TINI 0. TEND 10.E0 NMAX 1000000
OPTI NOTE
ECRI COOR DEPL VITE ACCE FINT FEXT FLIA CONT ECRO TFRE 1
CHAR CONS GRAV 0 -9.80665D0 LECT squa1 rect1 squa2 TERM
LINK COUP BLOQ 12 LECT lext TERM
OPTI PINS STAT
MATE LINE RO 8000.0 YOUN 1.E12 NU 0.0
GEOM CAR1 mesh TERM
CPLA LAGR
CAST mesh
!CONV win
ECHO
DRO202
dro202.epx
fin;
trac qual (pbox et psqua1 et prect1 et psqua2);
trac qual mesh;
tass mesh;
psqua2 = squa2 elem appu larg (cont squa2);
prect1 = rect1 elem appu larg (cont rect1);
psqua1 = squa1 elem appu larg (cont squa1);
taxe mesh;
sauv form mesh;
trac quali (pbox et pequa1 et prec1 et pequa2);
fin;

*=================================================================
ENDPLAY
FREQ 30
TRAC OFFS FICH BMP REND
SLER CAM1 1 NFRA 1
SCEN GEOM NAVI FREE
TRAC OFFS FICH BMP REND
SLER CAM1 1 NFRA 1
SCEN GEOM NAVI FREE
SLER CAM1 1 NFRA 1
SCEN GEOM NAVI FREE
GO
FREQ 2
GO
SCEN GEOM HAVI FREE
PACE NFRO
FIRE PARE
CPLA PAGE
SLER CAM1 1 NRA 1
TRAC OFFS FICH BMP REND
SCEN GEOM HAVI FREE
PACE NFRO
FIRE MAIN PARE
CPLA PAGE
SLER CAM1 1 NRA 1
TRAC OFFS FICH BMP REND
SCEN GEOM HAVI FREE
PACE NFRO
FIRE HBON
CPLA PAGE
SLER CAM1 1 NRA 1
TRAC OFFS FICH BMP REND
FREQ 10
GO
REDISPLAY
*--------------------------*
PLAY
CAME 1 EYE 3.00000E-02 2.18000E+00 1.21800E+01
! Q 1.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
VIEW 0.00000E+00 0.00000E+00 -1.00000E+00
RIND 0.00000E+00 0.00000E+00 0.00000E+00
UP 0.00000E+00 1.00000E+00 0.00000E+00
PV 2.48818E+01
FREQ 2
GO
SCEN GEOM HAVI FREE
PACE NFRO
FIRE PARE
CPLA PAGE
SLER CAM1 1 NRA 1
TRAC OFFS FICH BMP REND
*--------------------------*
FIN
*=================================================================
ENDPLAY
TRAC OFFS FICH AVI CONT REND
GOTR LOOP 999 OFFS FICH AVI CONT NOCL REND
FREQ 1
TRAC OFFS FICH AVI NOCL NFTO 1001 FPS 25 KFRE 10 COMP -1 REND
SLER CAM1 1 NFRA 1
SCEN GEOM NAVI FREE
PACE NFRO
FIRE PARE
CPLA PAGE
SLER CAM1 1 NRA 1
TRAC OFFS FICH BMP REND
*--------------------------*
PLAY
CAME 1 EYE 3.00000E-01 2.18000E+00 1.21800E+01
! Q 1.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
VIEW 0.00000E+00 0.00000E+00 -1.00000E+00
RIND 0.00000E+00 0.00000E+00 0.00000E+00
UP 0.00000E+00 1.00000E+00 0.00000E+00
PV 2.48818E+01
FREQ 2
GO
SCEN GEOM HAVI FREE
PACE NFRO
FIRE PARE
CPLA PAGE
SLER CAM1 1 NRA 1
TRAC OFFS FICH BMP REND
*--------------------------*
LIST 11 12 AXES 1.0 'FEET [M]' YZEK

ENDPLAY

FIN

*=================================================================

**Post-treatment (animation from alice file)**

**ECHO**

**REU ALIC 'dro204.ali' GRID PLOT**

**SORT VISU MFO 1**

**PLAY**

**CASE** 1 EYE 3.00000E+00 2.70138E+00 2.93227E+00 |
| Q 1.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 |
| VYEK 1.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 |
| RES 1.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 |
| UP 0.00000E+00 1.00000E+00 0.00000E+00 |
| FV 2.49810E+01 |

**SCEN GEOM NAVI FREE**

**COLO PAPE**

**SLS LCM: 1 NFRA 1**

**TRAC OFFS FICH AVI ANT MOCL MFO 1010 PPS 25 KFRE 10 COMP -1 REND**

**SORT OFFS FICH ANTI REND**

**ENDPLAY**

**FIN**

*=================================================================

**Post-treatment (animation from alice file)**

**ECHO**

**REU ALIC 'dro205.ali' GRID PLOT**

**SORT VISU MFO 1**

**PLAY**

**CASE** 1 EYE 3.00000E+00 2.70138E+00 2.93227E+00 |
| Q 1.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 |
| VYEK 1.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 |
| RES 1.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 |
| UP 0.00000E+00 1.00000E+00 0.00000E+00 |
| FV 2.49810E+01 |

**SCEN GEOM NAVI FREE**

**COLO PAPE**

**SLS LCM: 1 NFRA 1**

**TRAC OFFS FICH AVI ANT MOCL MFO 1010 PPS 25 KFRE 10 COMP -1 REND**

**SORT OFFS FICH ANTI REND**

**ENDPLAY**

**FIN**

*=================================================================

**Post-treatment (animation from alice file)**

**ECHO**

**REU ALIC 'dro205.ali' GRID PLOT**

**SORT VISU MFO 1**

**PLAY**

**CASE** 1 EYE 3.00000E+00 2.70138E+00 2.93227E+00 |
| Q 1.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 |
| VYEK 1.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 |
| RES 1.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 |
| UP 0.00000E+00 1.00000E+00 0.00000E+00 |
| FV 2.49810E+01 |

**SCEN GEOM NAVI FREE**

**COLO PAPE**

**SLS LCM: 1 NFRA 1**

**TRAC OFFS FICH AVI ANT MOCL MFO 1010 PPS 25 KFRE 10 COMP -1 REND**

**SORT OFFS FICH ANTI REND**

**ENDPLAY**

**FIN**

*=================================================================
**dro208a.epx**

Post-treatment (time curves from alice file)

**SUIT**

Post-treatment (animation from alice file)

**FIN**

**dro209.epx**

Post-treatment (animation from alice file)

**FIN**

**dro209a.epx**

Post-treatment (time curves from alice file)

**FIN**
**dro210.dgibi**

- **opti echo 1;**
- **opti dize 2 elem qua;**
- **opti sauve form 'dro211.dgibi';**
- **opti trac per ftra 'dro212_mesh.ps';**
- **p1 = -1;**
- **p2 = 0;**
- **p3 = 4.5;**
- **p4 = 7.0;**
- **p5 = -1.4;**
- **p6 = 8.5;**
- **p7 = 4.2;**
- **p8 = 7.2;**
- **p9 = 1.4;**
- **p10 = 7.2;**
- **n1 = 2;**
- **n2 = 12;**
- **tol = 1.83;**
- **c1 = p1 d n1 p2;**
- **box1 = c1 tran n2 p2;**
- **box2 = box plus (7) 0;**
- **c2 = p1 d n1 p6;**
- **box2 = c1 tran n1 n2 0;**
- **box = box1 box2 box et box3 et box4 et box5 et box6 et box7 et box8 et box9 et box10;**
- **elim tol (box et p1 et p2 et p3 et p4 et p5 et p6 et p7 et p8 et p9 et p10);**
- **c1 = p2 plus (0 0) d (n1 + n1) 0 0;**
- **equal c1 tran (n1 + n1) 0 0;**
- **depl equal plus (1 0);**
- **depl equal recti tour 45 (bary recti);**
- **equal equal plus (2 5 0 4 4 0);**
- **recti = et tran n1 0 0;**
- **depl recti plus (1 0 3);**
- **depl equal recti tour 15 (bary recti);**
- **equal equal plus (2 5 0 4 4 0);**
- **mesh = box et equal et recti et equal;**
- **lint = de g d n2 p2 d n3 p2 (n1 + n2 p3);**
- **elim tol (lint et box);**
- **lext = p9 d n2 p3 d n3 p6 d n1 p4 d n2 p10;**
- **elim tol (lext et box);**
- **pbox = box elem appu larg lint;**
- **pequal = equal elem appu larg (cont equal);**
- **precti = recti elem appu larg (cont recti);**
- **pequal = equal elem appu larg (cont equal);**
- **taas mesh;**
- **sauv form mesh;**
- **trac qual mesh;**
- **trac qual (pbox et pequal et precti et pequal);**
- **fin;**

**dro210.epx**

- **dro210**
- **END**
- **!COMP win**
- **CART mesh**
- **CPFL LASS**
- **GEOM GIBI mesh TERM**
- **COMP COUL VERT LECT tous TERM**
- **MATE LNIN SU 8000.0 VOM 1.812 NU 0.0**
- **LEFT term TERM**
- **OPT PINS STAT**
- **AXE**
- **NOBB**
- **LINK GIBI BLO 12 LECT left TERM**
- **LINK DROG FINE FROM SPACE 1.0**
- **BODY MVIU 2 LECT pbox TERM**
- **BODY MVIU 2 LECT pbox TERM**
- **BODY MVIU 2 LECT pbox TERM**
- **BODY MVIU 2 LECT pbox TERM**
- **CHAR CURVE CURVE 0 -3.64000 LECT equalrest equalterm**
- **ECHI DEPL DEPL ACCE FINE FINE PLIA CONT MCOU TVRE 1**
- **FINE ALLTIC 1.5 2**
- **OPTI MUTE**
- **CER 0.0 LVR**
- **LDS 10000**
- **CALCUL TERM 0. TERM 10.0000 HMAX 1000000**
- **fin**

**FIN**

**dro210a.epx**

- **dro210 Post-treatment (animation from alice file)**
- **END**
- **!COMP win**
- **CART mesh**
- **CPFL LASS**
- **GEOM GIBI mesh TERM**
- **COMP COUL VERT LECT tous TERM**
- **MATE LNIN SU 8000.0 VOM 1.812 NU 0.0**
- **LEFT term TERM**
- **OPT PINS STAT**
- **AXE**
- **NOBB**
- **LINK GIBI BLO 12 LECT left TERM**
- **LINK DROG FINE FROM SPACE 1.0**
- **BODY MVIU 2 LECT pbox TERM**
- **BODY MVIU 2 LECT pbox TERM**
- **BODY MVIU 2 LECT pbox TERM**
- **BODY MVIU 2 LECT pbox TERM**
- **CHAR CURVE CURVE 0 -3.64000 LECT equalrest equalterm**
- **ECHI DEPL DEPL ACCE FINE FINE PLIA CONT MCOU TVRE 1**
- **FINE ALLTIC 1.5 2**
- **OPTI MUTE**
- **CER 0.0 LVR**
- **LDS 10000**
- **CALCUL TERM 0. TERM 10.0000 HMAX 1000000**
- **fin**

**dro211.dgibi**

- **opti echo 1;**
- **opti dize 2 elem qua;**
- **opti sauve form 'dro211.dgibi';**
- **opti trac per ftra 'dro212_mesh.ps';**
- **p1 = -1.0;**
- **p2 = 0.0;**
- **p3 = 1.0;**
- **p4 = 2.0;**
- **p5 = 3.0;**
- **p6 = 4.0;**
- **p7 = 5.0;**
- **p8 = 6.0;**
- **p9 = 7.0;**
- **p10 = 8.0;**
- **n1 = 2;**
- **n2 = 12;**
- **tol = 1.83;**
- **c1 = p1 d n1 p2;**
- **box1 = c1 tran n2 p2;**
- **box2 = box plus (7) 0;**
- **c2 = p1 d n1 p6;**
- **box2 = c1 tran n1 n2 0;**
- **box = box1 box2 box et box3 et box4 et box5 et box6 et box7 et box8 et box9 et box10;**
- **elim tol (box et p1 et p2 et p3 et p4 et p5 et p6 et p7 et p8 et p9 et p10);**
- **c1 = p2 plus (0 0) d (n1 + n1) 0 0;**
- **equal c1 tran (n1 + n1) 0 0;**
- **depl equal plus (1 0);**
- **depl equal recti tour 45 (bary recti);**
- **equal equal plus (2 5 0 4 4 0);**
- **recti = et tran n1 0 0;**
- **depl recti plus (1 0 3);**
- **depl equal recti tour 15 (bary recti);**
- **equal equal plus (2 5 0 4 4 0);**
- **mesh = box et equal et recti et equal;**
- **lint = de g d n2 p2 d n3 p2 (n1 + n2 p3);**
- **elim tol (lint et box);**
- **lext = p9 d n2 p3 d n3 p6 d n1 p4 d n2 p10;**
- **elim tol (lext et box);**
- **pbox = box elem appu larg lint;**
- **pequal = equal elem appu larg (cont equal);**
- **precti = recti elem appu larg (cont recti);**
- **pequal = equal elem appu larg (cont equal);**
- **taas mesh;**
- **sauv form mesh;**
- **trac qual mesh;**
- **trac qual (pbox et pequal et precti et pequal);**
- **fin;**
ED0111.exp

ED011

CONV win

CPLA LAGS

GEOM LIBR POIN 6 ED01 4 TERM

0 1 2 0 3 0

1 1 1 2

1 2

3 4

COMP BRAZ 0.1 LECT tout TERM

CUDL VERT LECT tout TERM

MATE VM03 RG 8000.0 VOM 2.8111 NU 0.3 ELAS 2.68

TRAC 2 2.04E 1.8 3 3.04 1.001E0

LEFT tout TERM

OPTI PINS :dump

@stat

@vide

@EQL

@EQU

@EUV

@EVP

LINE CUDU ELOG 12 LECT 1 4 TERM

LINE DECO PINE PAN

BODY MLV5 5 LECT 1 2 3 TERM

BODY MLV5 5 LECT 4 TERM

INIT VITE 2 -100.0 LECT 5 6 TERM

ECRI VITE TPRK 1.0

FICH ALIC FREQ 1

OPTI NOTE

CSTA 0.5

LOG 1

CALC TINT 0. TEND 1.0 INNX 300

FIN

*-----*

PLAY

GAME 1 EVE 1.000000E0 2.86594E-01 7.54573E-02

| Q | 1.000000E0 0.000000E0 0.000000E0 0.000000E0

VIEW 0.000000E0 0.000000E0 -1.00000E0

RIGH 1.00000E0 0.00000E0 0.00000E0

UP 0.00000E0 1.00000E0 0.00000E0

FOV 2.48819E+01

NAVIGATION MODE: ROTATING CAMERA

CENTER : 1.000000E0 2.86594E-01 0.00000E0

EQUALL 1.000000E0

EQUALL 0.00000E0

EQUALL 0.00000E0

EQUALL 0.00000E0

FOV 2.48819E+01

PINB BODY MLEV 0 LECT 1 2 3 TERM

FIN

*-----*

ED0122.exp

ED012

CONV win

CPLA LAGS

GEOM LIBR POIN 4 ED01 4 TERM

0 1 2 0 3 0

1 1 1 2

1 2

3 4

COMP BRAZ 0.1 LECT tout TERM

CUDL VERT LECT tout TERM

MATE VM03 RG 8000.0 VOM 2.8111 NU 0.3 ELAS 2.68

TRAC 2 2.04E 1.8 3 3.04 1.001E0

LEFT tout TERM

OPTI PINS :dump

@stat

@vide

@EQL

@EQU

@EUV

@EVP

LINE CUDU ELOG 12 LECT 1 4 TERM

LINE DECO PINE PAN

BODY MLV5 5 LECT 1 2 3 TERM

BODY MLV5 5 LECT 4 TERM

INIT VITE 2 -100.0 LECT 5 6 TERM

ECRI VITE TPRK 1.0

FICH ALIC FREQ 1

OPTI NOTE

CSTA 0.5

LOG 1

CALC TINT 0. TEND 1.0 INNX 300

FIN

*-----*
pene02.epx

**pene02.epx**

**pene03.epx**

**pene04.epx**
PENE05.epx

PRED3
END

*/CONV win

PENE06.epx

PRED3
END

*/CONV win

PENE07.epx

PRED3
END

*/CONV win
**pene08.dgibi**

```plaintext
opti echo 1;
opti dise 3 elem cub;
opti sauv form "pene08_mesh.ps";
opti trac per ftrc "pene08_mesh.ps";
p1 = 0 0 0;
p2 = 1 0 0;
p3 = 1 1 0;
p4 = 0 0 1;
p5 = 0 1 0;
p6 = 1 1 1;
p7 = 0 1 1;
e1 = nomu cub p1 p2 p3 p4 p5 p6 p7 p8;
e2 = e1 plus (0 1 2);
be2 = bary e2;
be2p = be2 plus (0 0 0);
depl e2 tour 45 be2 be2p;
```

**pene08.epx**

```plaintext
VIEW 0.00000E+00 0.00000E+00 1.00000E+00
SIZE 1.00000E+00 0.00000E+00 0.00000E+00
UP 0.00000E+00 1.00000E+00 0.00000E+00
POV 2.48339E+01
FREQ 2
GO
SCGN GEOM NAVI FREE
FACE FREE
PINFREE
COLD PAPE
SLER CAM1 1 NFRA 1
TRAC OFFS FICH BMP REND
SCGN GEOM NAVI FREE
FACE FREE
PINFREE
COLD PAPE
SLER CAM1 1 NFRA 1
TRAC OFFS FICH BMP REND
SCGN GEOM NAVI FREE
FACE FREE
PINFREE
COLD PAPE
SLER CAM1 1 NFRA 1
TRAC OFFS FICH BMP REND
FREQ 30
GO
ENDPLAY
```

**pene09.dgibi**

```plaintext
opti echo 1;
opti dise 3 elem cub;
opti sauv form "pene09_mesh.ps";
opti trac per ftrc "pene09_mesh.ps";
p1 = 0 0 0;
p2 = 1 0 0;
p3 = 1 1 0;
p4 = 0 0 1;
p5 = 0 1 0;
p6 = 1 1 1;
p7 = 0 1 1;
e1 = nomu cub p1 p2 p3 p4 p5 p6 p7 p8;
e2 = e1 plus (0 0 1.425);
be2 = bary e2;
be2p = be2 plus (0 0 0);
depl e2 tour 45 be2 be2p;
```

**pene09.epx**

```plaintext
VIEW 0.00000E+00 0.00000E+00 1.00000E+00
SIZE 1.00000E+00 0.00000E+00 0.00000E+00
UP 0.00000E+00 1.00000E+00 0.00000E+00
POV 2.48339E+01
FREQ 2
GO
SCGN GEOM NAVI FREE
FACE FREE
PINFREE
COLD PAPE
SLER CAM1 1 NFRA 1
TRAC OFFS FICH BMP REND
SCGN GEOM NAVI FREE
FACE FREE
PINFREE
COLD PAPE
SLER CAM1 1 NFRA 1
TRAC OFFS FICH BMP REND
FREQ 30
GO
ENDPLAY
```

---

**pene09.dgibi**

```plaintext
opti echo 1;
opti dise 3 elem cub;
opti sauv form "pene09_mesh.ps";
opti trac per ftrc "pene09_mesh.ps";
p1 = 0 0 0;
p2 = 1 0 0;
p3 = 1 1 0;
p4 = 0 0 1;
p5 = 0 1 0;
p6 = 1 1 1;
p7 = 0 1 1;
e1 = nomu cub p1 p2 p3 p4 p5 p6 p7 p8;
e2 = e1 plus (0 0 1.425);
be2 = bary e2;
be2p = be2 plus (0 0 0);
depl e2 tour 45 be2 be2p;
```

**pene09.epx**

```plaintext
VIEW 0.00000E+00 0.00000E+00 1.00000E+00
SIZE 1.00000E+00 0.00000E+00 0.00000E+00
UP 0.00000E+00 1.00000E+00 0.00000E+00
POV 2.48339E+01
FREQ 2
GO
SCGN GEOM NAVI FREE
FACE FREE
PINFREE
COLD PAPE
SLER CAM1 1 NFRA 1
TRAC OFFS FICH BMP REND
SCGN GEOM NAVI FREE
FACE FREE
PINFREE
COLD PAPE
SLER CAM1 1 NFRA 1
TRAC OFFS FICH BMP REND
FREQ 30
GO
ENDPLAY
```
### pene10.dgibi

```plaintext
**PENET**

END

CONV win

**ECHO**

PMAT11

pmat00.epx

PLAY

**CALC TINI 0. TEND 100.E-3 NMAX 0**

**OPTI NOTE**

ECRI COOR DEPL VITE ACCE FINT FEXT FLIA FDEC CONT ECRO FREQ 50

**OPTI PINS DUMP**

MATE LINE RO 8000.0 YOUN 1.E12 NU 0.0

COMP EPAI 1.0 LECT tous TERM

**GEOM LIBR POIN 2 PMAT 2 TERM**

CPLA LAGR

ECHO

PMAT11

pmat00.epx

**FIN**
```

### pene10.epx

```plaintext
**PENET**

END

**CONV win**

**ECHO**

CPLA LAGR

**GEOM LIBR POIN 2 PMAT 2 TERM**

CPLA LAGR

ECHO

CONV win

**FIN**
```

### pmat01.epx

```plaintext
**PENET**

END

**CONV win**

**ECHO**

CPLA LAGR

**GEOM LIBR POIN 2 PMAT 2 TERM**

CPLA LAGR

ECHO

**FIN**
```

### pene01.epx

```plaintext
**PENET**

END

**CONV win**

**ECHO**

CPLA LAGR

**GEOM LIBR POIN 2 PMAT 2 TERM**

CPLA LAGR

ECHO

**FIN**
```
**size01.epx**

```plaintext
SIZE 1.00000E+00 0.00000E+00 0.00000E+00
CP 0.00000E+00 1.00000E+00 0.00000E+00
POV 2.48418E+01

FREQ 2
GO
SCEN Geom NAVI FREE
FACE NFPO
FIRE FREE
CULO RACE
SLER CM1 1 HIFA 1
TRAC OFFS FICH BNP REND
SCEN Geom NAVI FREE
FACE NFPO
FIRE NFPO
CULO RACE
SLER CM1 1 HIFA 1
TRAC OFFS FICH BNP REND

FREQ 2
GO
EDIT

*----------------------------------------------------------------------

SUIT
Post-treatment (time curves from alice file)
END

**size02.epx**

```plaintext
SIZE 20

ECHO

(COPY win)

CULO LAGE

SCEN Geom NAVI FREE
FACE NFPO
FIRE FREE
CULO RACE
SLER CM1 1 HIFA 1
TRAC OFFS FICH BNP REND

FREQ 20
GO
EDIT

*----------------------------------------------------------------------

SUIT
Post-treatment (time curves from alice file)
END

**size03.epx**

```plaintext
SIZE 30

ECHO

(COPY win)

CULO LAGE

SCEN Geom NAVI FREE
FACE NFPO
FIRE FREE
CULO RACE
SLER CM1 1 HIFA 1
TRAC OFFS FICH BNP REND

FREQ 30
GO
EDIT

*----------------------------------------------------------------------

SUIT
Post-treatment (time curves from alice file)
END

```
The text appears to be a combination of code and comments, possibly from a computer simulation or modeling program. The content includes various commands and parameters, such as simulation settings, output instructions, and graphical settings. It is not clear what the specific purpose of the code is without additional context or a clear explanation of the simulation or model being described.
```plaintext
SCEN GEOM NAVI FREE
FAR : 8.77809E+00
NEAR : 4.07554E+00
ASPECT : 1.00000E+00
RSPHERE: 1.56752E+00
CENTER : 5.00000E-01 0.00000E+00 0.00000E+00

SCEN GEOM NAVI FREE
FAR : 5.01605E+00
NEAR : 3.13503E-01
ASPECT : 1.33333E+00
RADIUS : 1.88102E+00
RSPHERE: 1.56752E+00
CENTER : 5.00000E-01 0.00000E+00 0.00000E+00
NAVIGATION MODE: ROTATING CAMERA

Q 1.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
CAME 1 EYE 5.00000E-01 0.00000E+00 5.64306E+00
PLAY

Post-treatment from Alice file
ENDPLAY

GO
GOTR LOOP 9 OFFS FICH AVI CONT REND
TRAC OFFS FICH AVI CONT NOCL REND
FREQ 1
TRAC OFFS FICH AVI NOCL NFTO 11 FPS 5 KFRE 5 COMP -1 REND
SLER CAM1 1 NFRA 1

FOV 2.48819E+01
UP 0.00000E+00 1.00000E+00 0.00000E+00
RIGH 1.00000E+00 0.00000E+00 0.00000E+00
VIEW 0.00000E+00 0.00000E+00 -1.00000E+00

Play

Post-treatment from Alice file
ENDPLAY

GO
GOTR LOOP 9 OFFS SIZE 800 600 FICH BMP REND
TRAC OFFS SIZE 800 600 FICH BMP REND
FREQ 1
TRAC OFFS SIZE 800 600 FICH BMP REND
SLER CAM1 1 NFRA 1

FOV 2.48819E+01
UP 0.00000E+00 1.00000E+00 0.00000E+00
RIGH 1.00000E+00 0.00000E+00 0.00000E+00
VIEW 0.00000E+00 0.00000E+00 -1.00000E+00

Play

Post-treatment from Alice file
ENDPLAY

GO
GOTR LOOP 9 OFFS SIZE 800 600 FICH BMP REND
TRAC OFFS SIZE 800 600 FICH BMP REND
FREQ 1
TRAC OFFS SIZE 800 600 FICH BMP REND
SLER CAM1 1 NFRA 1

FOV 2.48819E+01
UP 0.00000E+00 1.00000E+00 0.00000E+00
RIGH 1.00000E+00 0.00000E+00 0.00000E+00
VIEW 0.00000E+00 0.00000E+00 -1.00000E+00

Play

Post-treatment from Alice file
ENDPLAY

GO
GOTR LOOP 9 OFFS SIZE 800 600 FICH BMP REND
TRAC OFFS SIZE 800 600 FICH BMP REND
FREQ 1
TRAC OFFS SIZE 800 600 FICH BMP REND
SLER CAM1 1 NFRA 1
```

**vide01.epx**

**vide03.epx**

**vide02.epx**
List of input files

This Section contains, in alphabetical file order, the listings of all input files related to the examples asnc01.epx ............................. 173
asnc02.epx .................................. 173
asnc03.epx .................................. 173
asnc04.epx .................................. 174
asnc05.epx .................................. 174
asnc06.epx .................................. 174
asnc07.epx .................................. 175
asnc08.epx .................................. 175
asnc09.epx .................................. 175
asnc10.epx .................................. 175
asnc11.epx .................................. 176
asnc12.epx .................................. 176
asnc13.epx .................................. 176
asnc16.epx .................................. 177
asnc00.epx .................................. 177
asnc01.epx .................................. 177
asnc02.epx .................................. 177
asnc03.epx .................................. 178
asnc04.epx .................................. 178
asnc04.dgibi ................................. 178
asnc05.epx .................................. 179
asnc05.dgibi ................................. 179
bar201.epx .................................. 179
bar201.a.epx ................................ 179
bar201.dgibi ................................. 179
bar202.epx .................................. 179
dro201.epx .................................. 179
dro201.a.epx ................................ 179
dro201.dgibi ................................. 179
dro202.epx .................................. 179
dro202.a.epx ................................ 179
dro202.dgibi ................................. 179
dro203.epx .................................. 179
dro203.a.epx ................................ 179
dro203.dgibi ................................. 179
dro205.epx .................................. 179
dro205.dgibi ................................. 179
dro206.epx .................................. 179
dro206.a.epx ................................ 179
dro206.dgibi ................................. 179
dro207.epx .................................. 179
dro207.dgibi ................................. 179
dro208.epx .................................. 179
dro208.dgibi ................................. 179
dro209.epx .................................. 179
dro209.dgibi ................................. 179
dro210.epx .................................. 179
dro210.dgibi ................................. 179
dro211.epx .................................. 179
dro211.dgibi ................................. 179
dro212.epx .................................. 179
ed0101.epx ................................ 182
ed0102.epx ................................ 182
ed0111.epx ................................ 182
ed0112.epx ................................ 182
ed0121.epx ................................ 182
ed0122.epx ................................ 182
carn07.epx ................................ 182
carn07a.epx ................................ 183
carn07b.epx ................................ 183
carn07c.epx ................................ 183
carn07d.epx ................................ 183
carn07e.epx ................................ 183
carn07f.epx ................................ 183
carn07g.epx ................................ 183
carn07h.epx ................................ 183
carn07i.epx ................................ 183
carn07l.epx ................................ 183
carn07m.epx ................................ 183
carn07n.epx ................................ 183
carn07o.epx ................................ 183
carn07p.epx ................................ 183
carn07q.epx ................................ 183
carn07r.epx ................................ 183
carn07s.epx ................................ 183
carn07t.epx ................................ 183
carn07u.epx ................................ 183
carn07v.epx ................................ 183
carn07w.epx ................................ 183
carn07x.epx ................................ 183
carn07y.epx ................................ 183
carn07z.epx ................................ 183
carro07a.epx ................................ 184
carro07b.epx ................................ 184
carro07c.epx ................................ 184
carro07d.epx ................................ 184
carro07e.epx ................................ 184
carro07f.epx ................................ 184
carro07g.epx ................................ 184
carro07h.epx ................................ 184
carro07i.epx ................................ 184
carro07j.epx ................................ 184
carro07k.epx ................................ 184
carro07l.epx ................................ 184
carro07m.epx ................................ 184
carro07n.epx ................................ 184
carro07o.epx ................................ 184
carro07p.epx ................................ 184
carro07q.epx ................................ 184
carro07r.epx ................................ 184
carro07s.epx ................................ 184
carro07t.epx ................................ 184
carro07u.epx ................................ 184
carro07v.epx ................................ 184
carro07w.epx ................................ 184
carro07x.epx ................................ 184
carro07y.epx ................................ 184
carro07z.epx ................................ 184
GETTING IN TOUCH WITH THE EU

In person
All over the European Union there are hundreds of Europe Direct information centres. You can find the address of the centre nearest you at: http://europea.eu/contact

On the phone or by email
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, or
- by electronic mail via: http://europa.eu/contact

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 at: http://europa.eu

EU publications
You can download or order free and priced EU publications from EU Bookshop at: http://bookshop.europa.eu. Multiple copies of free publications may be obtained by contacting Europe Direct or your local information centre (see http://europa.eu/contact).
JRC Mission

As the science and knowledge service of the European Commission, the Joint Research Centre’s mission is to support EU policies with independent evidence throughout the whole policy cycle.

EU Science Hub
ec.europa.eu/jrc

@EU_ScienceHub
EU Science Hub - Joint Research Centre
Joint Research Centre
EU Science Hub