Università degli studi di Padova
Dipartimento di Ingegneria Civile, Edile E Ambientale
Corso di Laurea in Ingegneria Civile

TESI DI LAUREA

SVILUPPO DI UN CODICE AGLI ELEMENTI DISCRETI POLIEDRICI

Relatore:
Chiar.mo Prof. GIANLUCA MAZZUCCO

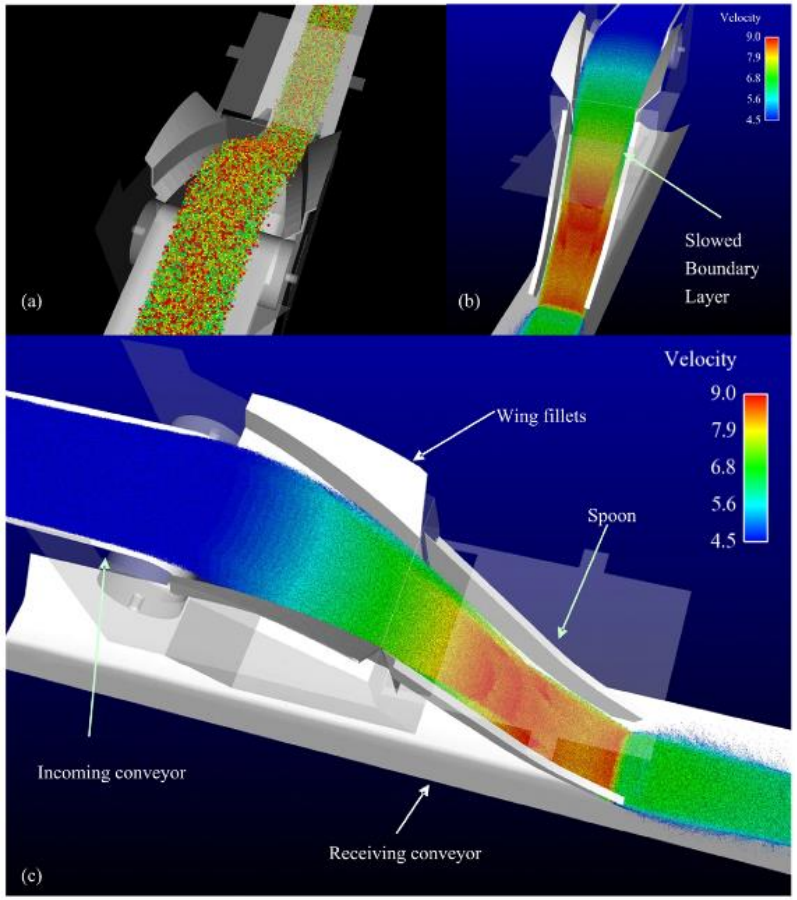
Laureando: RICCARDO LANDO
2052462

Anno Accademico 2023/2024

DEM - INTRODUZIONE

Il Metodo agli Elementi Discreti (Discrete Element Method - DEM) è un metodo specificatamente creato per la simulazione di un elevato numero di corpi indipendenti (particelle) che interagiscono tramite contatti. Nell'ambito dell'ingegneria, è utilizzato per svolgere simulazioni di vario tipo:

- Geotecnica: prove di laboratorio (triassiali), terreni ghiaiosi o frane;
- Civile: prove di taglio su muratura, collasso edifici in muratura;
- Industriale: tramogge, nastri trasportatori, apparati di mixaggio.



Simulazione nastro trasportatore[1]

[1] 2010, Cleary. ≪DEM prediction of industrial and geophysical particle flows≫

DEM - INTRODUZIONE

Si vanno a distinguere 3 punti principali nello sviluppo della tesi:

TEORIA METODO
ELEMENTI DISCRETI

SVILUPPO
INFORMATICO
CODICE

BENCHMARK
CODICE

DEM – IPOTESI SOFT SPHERES

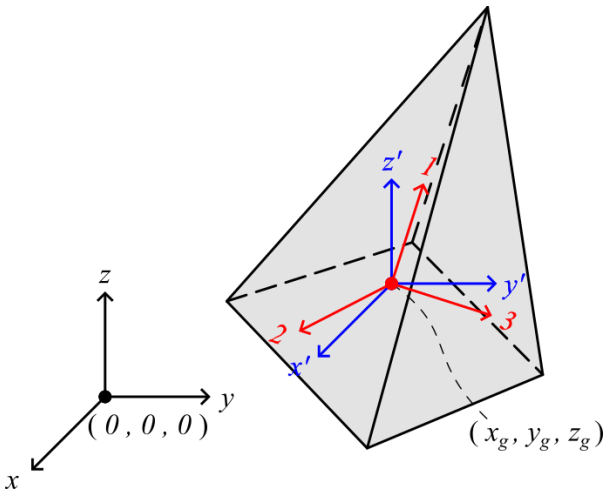
Nella tesi si è sviluppato un codice DEM di tipo Soft Spheres (*Sfere Deformabili*), il quale opera secondo le seguenti ipotesi:

- Particelle indeformabili;
- Possibile compenetrazione tra particelle (per simulare la deformazione);
- Collisione non istantanea;
- Equazioni di equilibrio dinamico per calcolo interazione;
- Possibilità di avere molteplici collisioni contemporane e modelli costitutivi di contatto complesso.

DEM – PARTICELLE POLIEDRICHE

Gli elementi poliedrici sono elementi 3-dimensionali generati da:

- una nuvola di punti con le coordinate della superficie;
- una matrice di incidenza faccia-vertici per definire le facce triangolari



Poliedro con i vari sistemi di riferimento

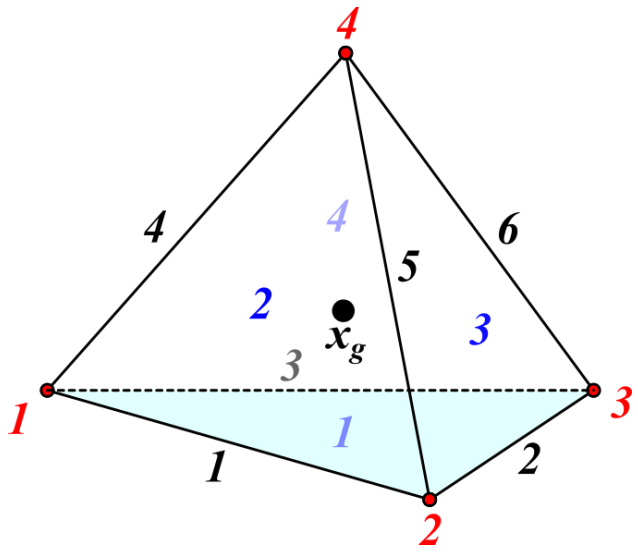
Nel codice, i poliedri sono collassati nel **baricentro**:

$$x_G = \frac{\sum_{v=1}^{n} A_v \, x_v}{\sum_{v=1}^{n} A_v}$$

dove sono riferite le **variabili cinematiche**:

$$u_i \, , \, v_i \, , \, a_i \, , \, \vartheta_i \, , \, \omega_i \, , \, \dot{\omega}_i \qquad \forall i = (x, y, z)$$

La superficie composta da facce e spigoli è necessaria esclusivamente per la ricerca del contatto.



Poliedro con elementi evidenziati:
vertici – spigoli – facce - **baricentro**

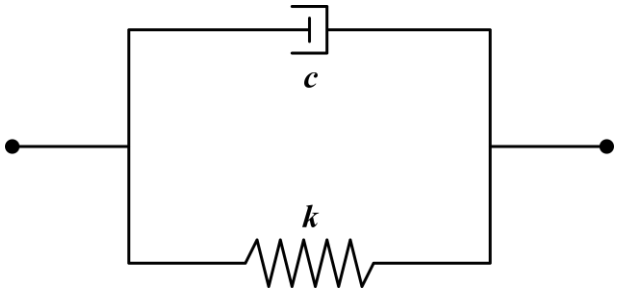
DEM – MODELLO COSTITUTIVO DI CONTATTO

Per modellare le interazioni di contatto si è utilizzato il modello linear spring-dashpot (molla lineare-ammortizzatore), descritto dall'equazione:

$$\boldsymbol{F}_n(\boldsymbol{\delta}_n, \dot{\boldsymbol{\delta}}_n) = k_n \cdot \boldsymbol{\delta}_n + c_n \cdot \dot{\boldsymbol{\delta}}_n$$

molla lineare

ammortizzatore



Modello reologico ls-d

dove:

- $\mathbf{F}_n$ = forza di contatto normale;
- $\boldsymbol{\delta}_n$ = distanza di compenetrazione in direzione normale tra poliedri (corrisponde al modulo del vettore gap **g**);
- $\dot{\boldsymbol{\delta}}_n$ = velocità relativa di contatto in direzione normale poliedri;
- k_n = coefficiente di rigidezza penalty della molla lineare;
- c_n = coefficiente di smorzamento dell'ammortizzatore;

DEM – SOLUTORE ED EQUAZIONI EQUILIBRIO

Si è utilizzato un solutore dinamico esplicito alle differenze centrali, specificatamente la variante nota con il nome *leap-frog* per calcolare le variabili cinematiche. Il solutore esplicito permettere di risolvere l'Equazione di Equilibrio Dinamico singolarmente per ogni particella, rendendo possibile un risparmio di RAM:

$$m_p \ddot{\boldsymbol{d}}_p = \sum \boldsymbol{F}_p^{con} + \boldsymbol{F}_p^g + \boldsymbol{D}_p + \boldsymbol{dF}_p$$

dove:

- m_p : massa particella;
- $\ddot{\boldsymbol{d}}_p$: accelerazione particella;
- $\sum \boldsymbol{F}_p^{con}$: forze di contatto calcolate con linear spring-dashpot;
- $\boldsymbol{F}_p^g$: forza esterna imposta tramite condizioni al contorno;
- $\boldsymbol{D}_p$: forza di smorzamento viscoso;
- $\boldsymbol{dF}_p$: forza smorzamento numerico.

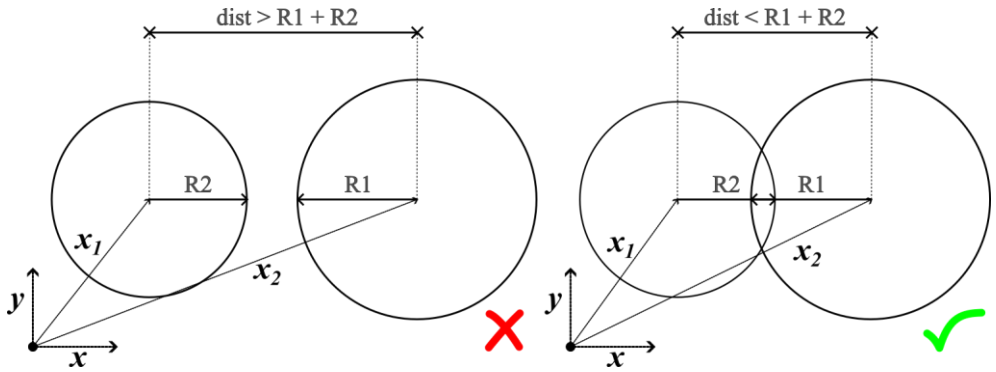
DEM – RICERCA CONTATTO

Nel codice è stata implementata la ricerca del contatto in 2 passaggi:

Neighbour Search tramite **Ricerca a Sfere**:

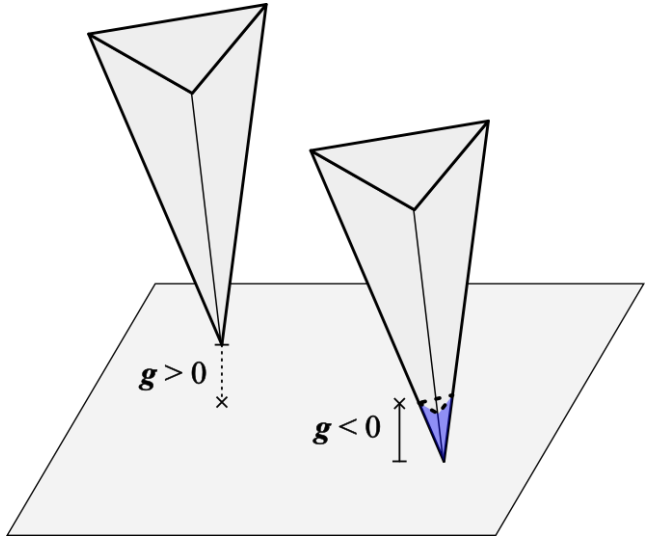
$$\sqrt{\boldsymbol{x}_1 \cdot \boldsymbol{x}_2} \leq R_1 + R_2$$

- $\boldsymbol{x_1}$ e $\boldsymbol{x_2}$ vettori posizione sfere
- $\boldsymbol{R_1}$ e $\boldsymbol{R_2}$ raggi sfere

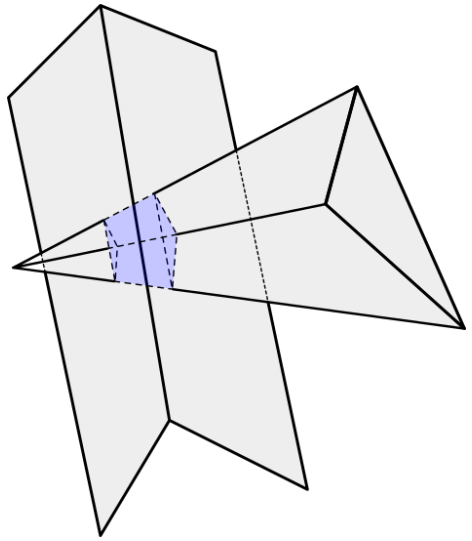


Rappresentazione della Ricerca a Sfere nel piano

Delicate Search tramite **Direct-Search:**



Contatto Vertice-Faccia



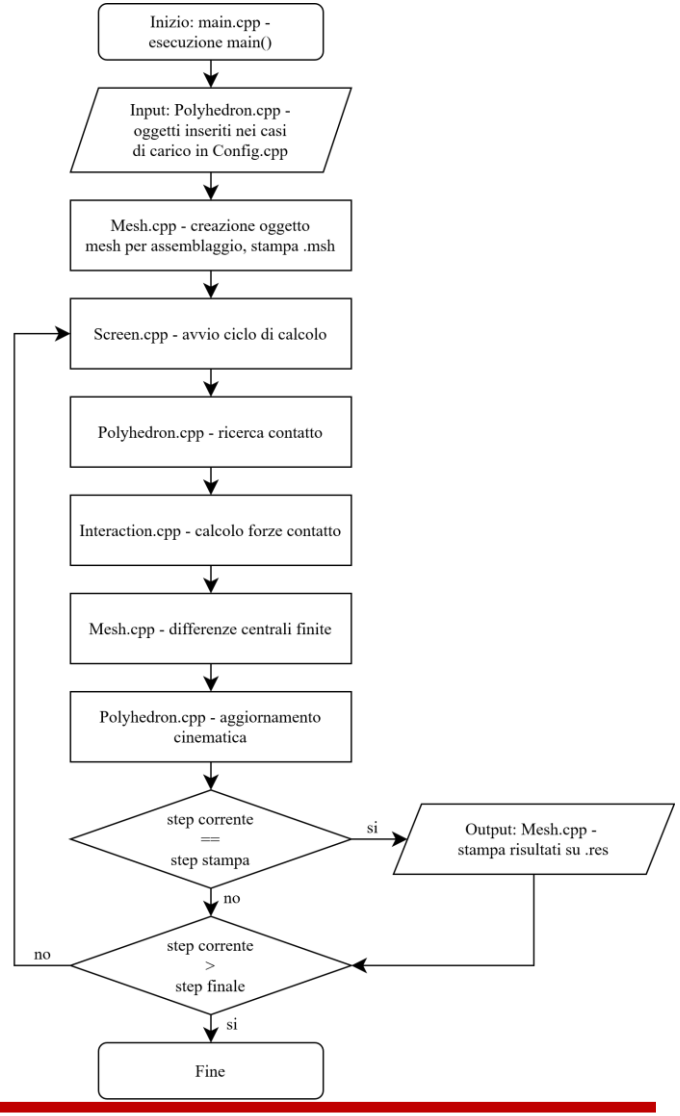
Contatto Spigolo-Spigolo

DEM – GENERALITÀ CODICE E FLOWCHART

Il codice è stato scritto in C++, al fine di ottenere le massime prestazioni possibili ed il totale controllo sulle variabili di calcolo. Il codice è composto da 6 classi principali con le loro principali funzionalità:

- Global (variabili globali, funzioni algebriche)
- Polyhedron (elementi poliedrici e metodi ricerca contatto)
- Mesh (solutore esplicito, variabili cinematiche, stampa su file)
- Contact (informazioni contatto)
- Interaction (calcolo forza di contatto, funzione linear spring-dashpot)
- Screen (gestione ciclo di calcolo)

Si riporta a fianco il diagramma di flusso del codice.



DEM – CONTROLLI SPECIFICI CODICE

Nel codice sono stati implementati delle funzioni di controllo per aiutare la convergenza dell'analisi e per monitorare l'andamento della simulazione, in particolare:

- Controllo timestep massimo con eventuale riduzione timestep:

$$\Delta t \leq [t_{coll}/20,\ t_{coll}/50]$$

- Controllo sugli spostamenti massimi con eventuale riduzione timestep:

$$d_{max} = [\xi_{max}/10,\ \xi_{max}/50]$$

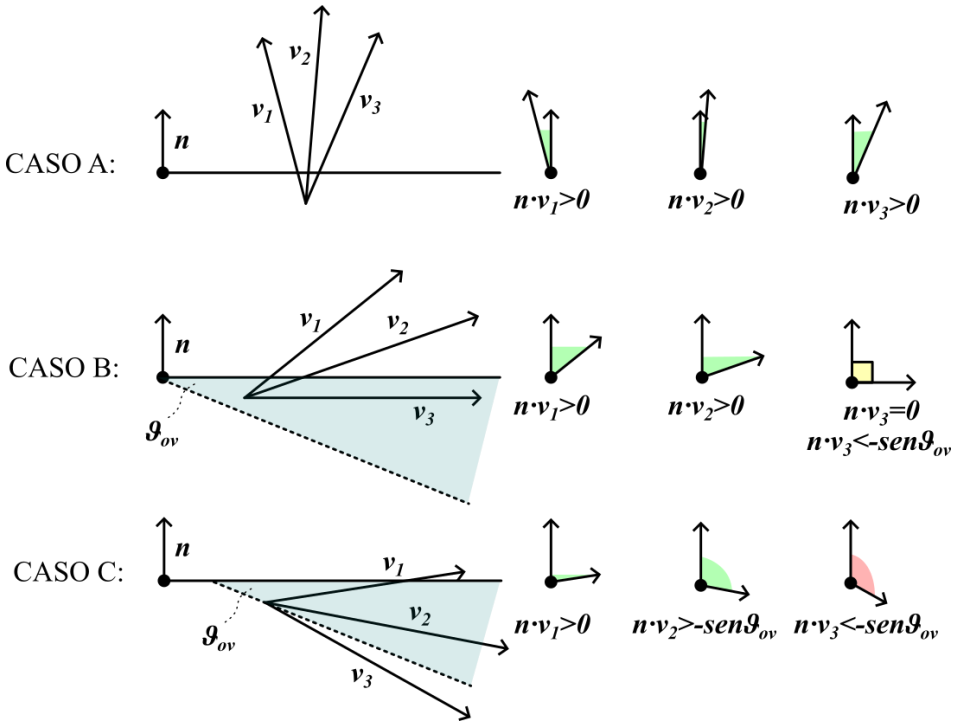
- Controllo di sovrapposizione massima con warning nel log;

- Controllo sul successo dello step di calcolo corrente, con eventuale ricalcolo a timestep diminuito.

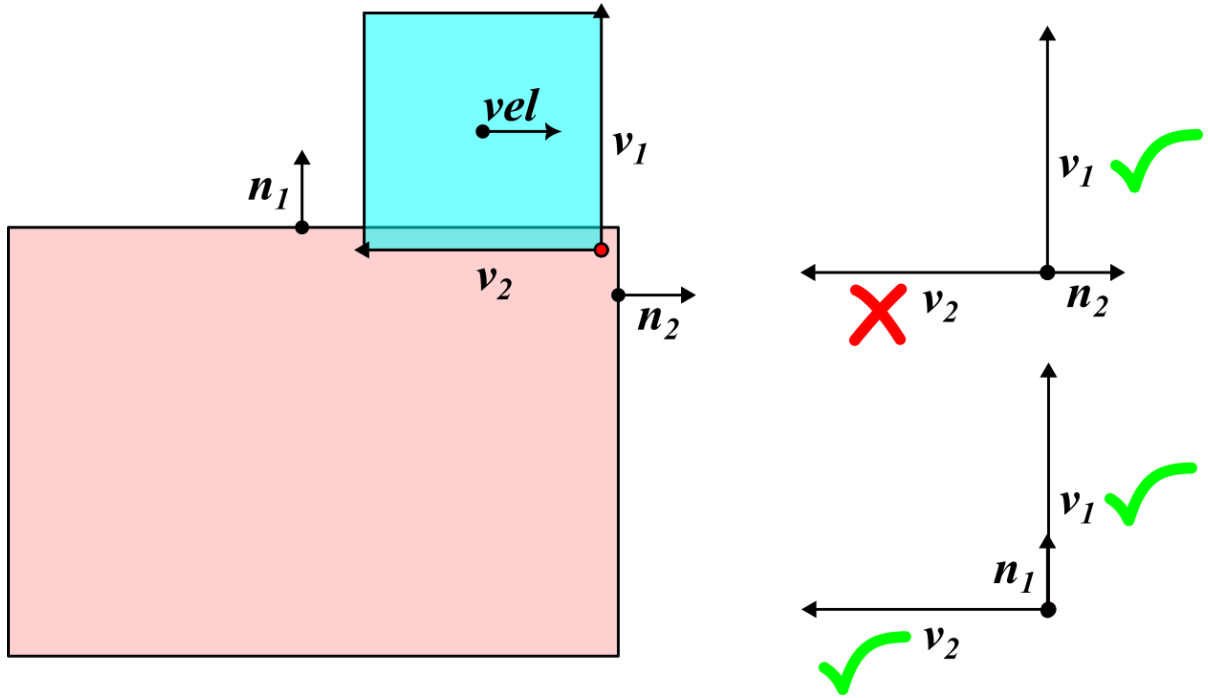
Legenda Simboli:
- d_{max} = spostamento massimo
- ξ_{max} = sovrapposizione massima
- Δt = step temporale
- t_{coll} = *tempo di collisione*

DEM – CONTROLLI SPECIFICI CODICE

- Controllo di non-sovrapposizione contatti VF/EE con esclusione contatti invalidi.



Controlli di non-sovrapposizione contatti VF



Dimostrazione caso misto valido/non-valido

DEM – OBIETTIVI BENCHMARK

Si vogliono investigare 3 tipi di benchmark, al fine di rilevare:

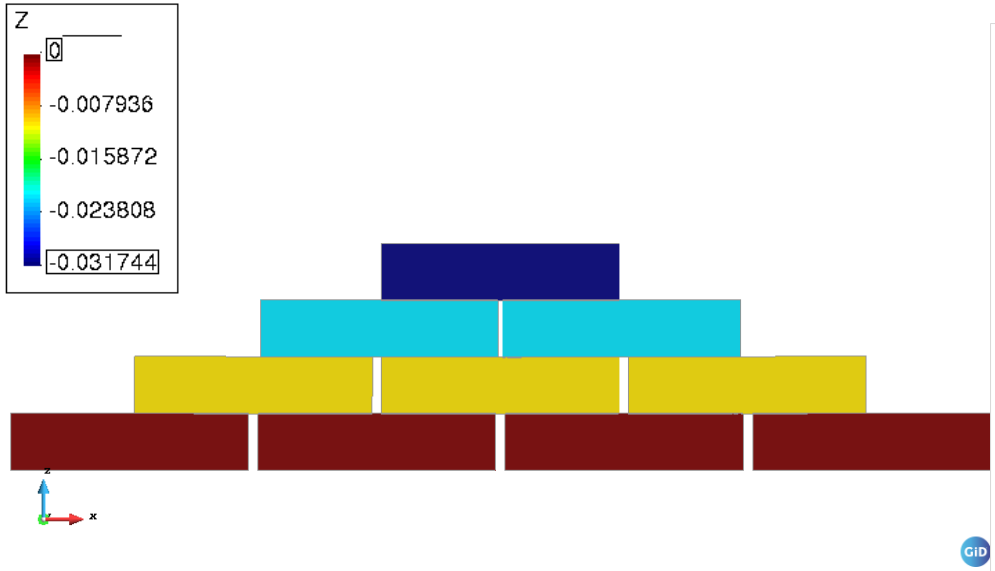
STABILITÀ

SIMMETRIA

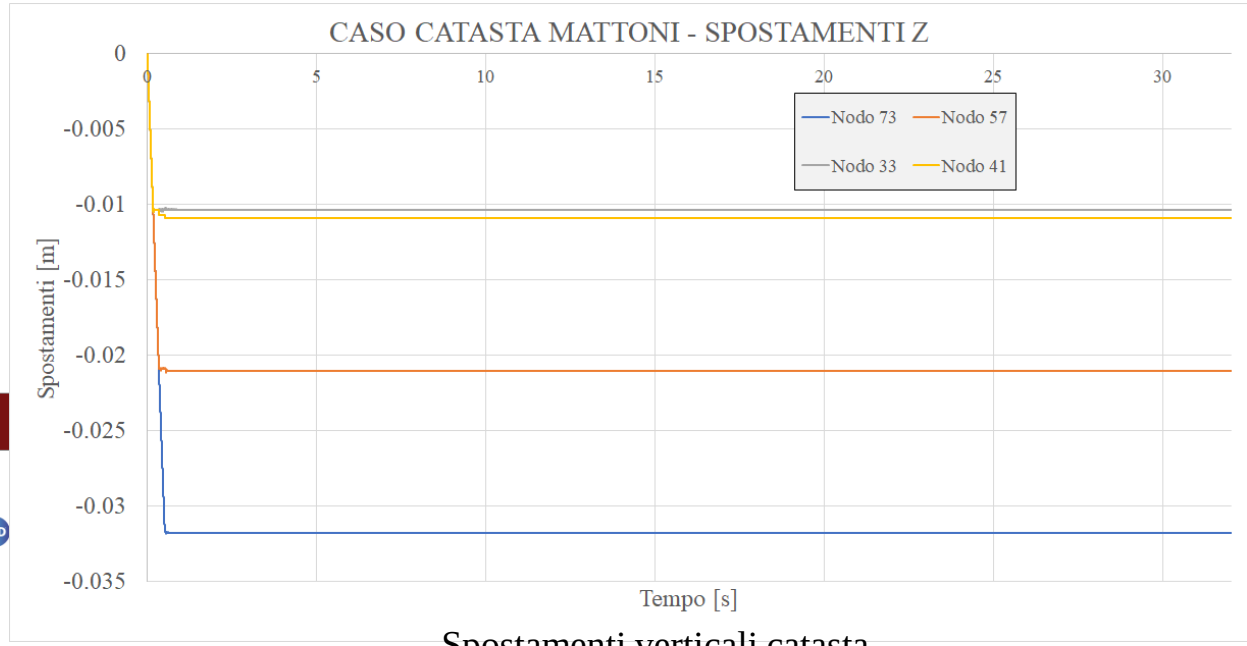
ACCURATEZZA (CALIBRAZIONE)

DEM – BENCHMARK STABILITÀ

Si sono eseguiti svariati test per verificare la stabilità del codice.



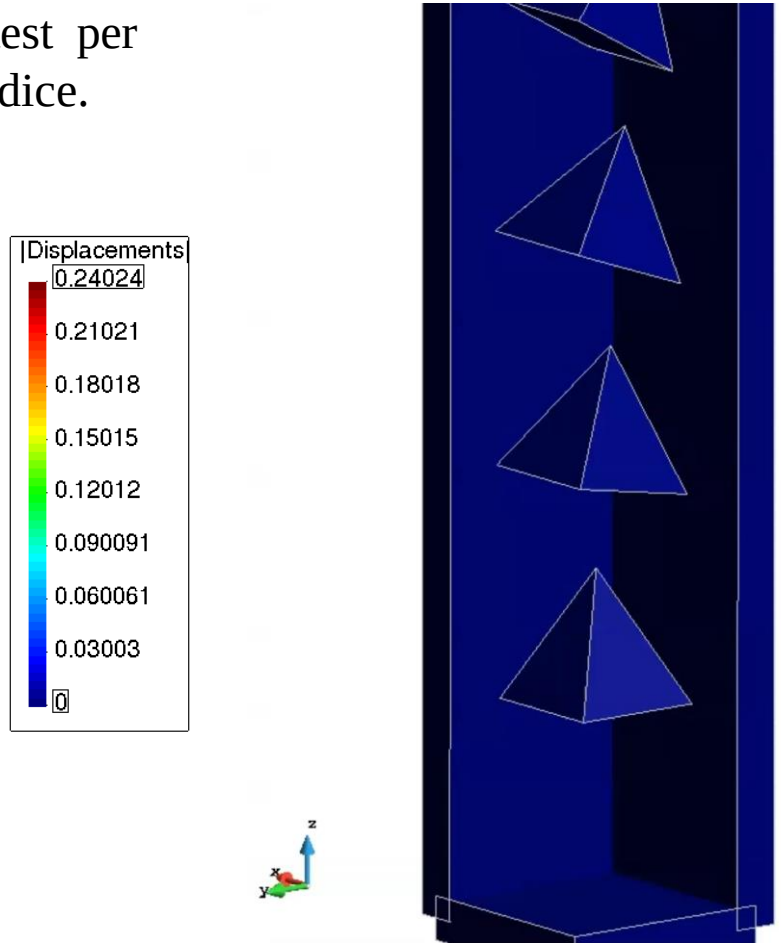
Configurazione finale catasta stabile a 32 s.



Spostamenti verticali catasta.

DEM – BENCHMARK STABILITÀ

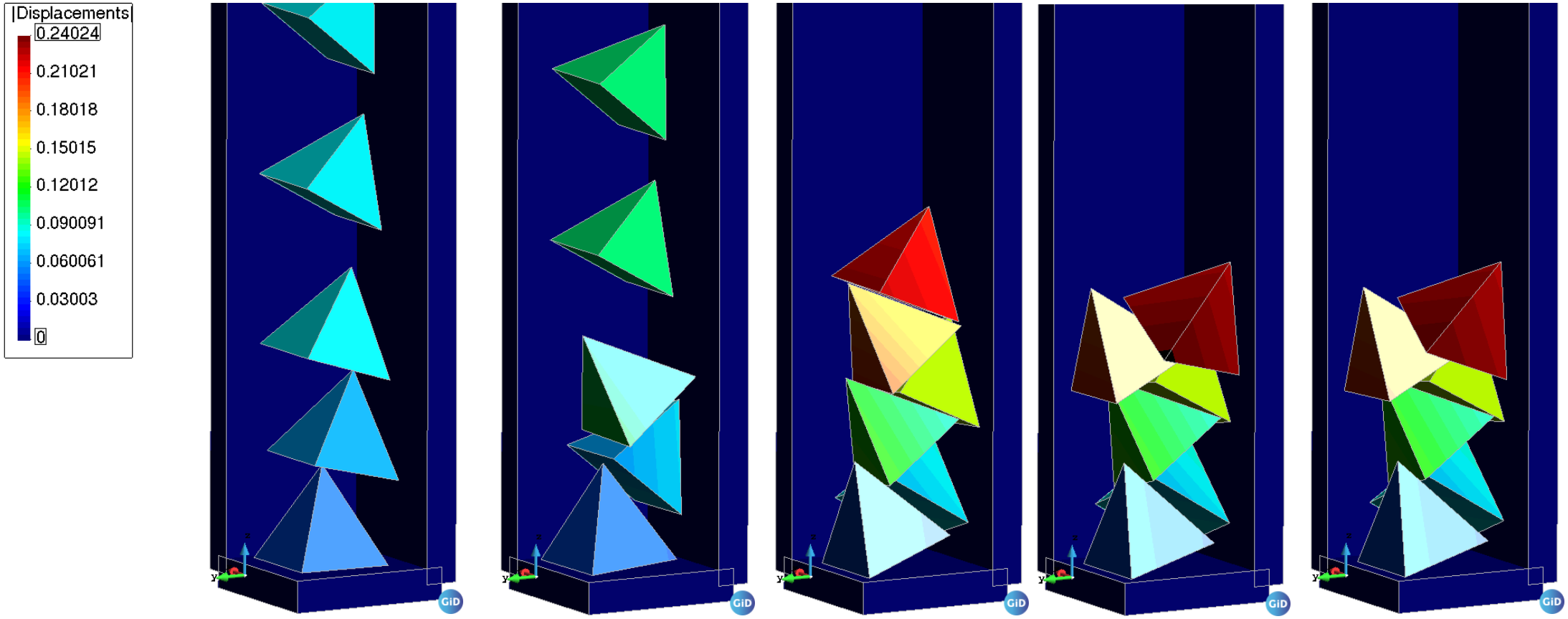
Si sono eseguiti svariati test per verificare la stabilità del codice.



Andamento test di accumulo piramidi in caduta in un contenitore.

DEM – BENCHMARK STABILITÀ

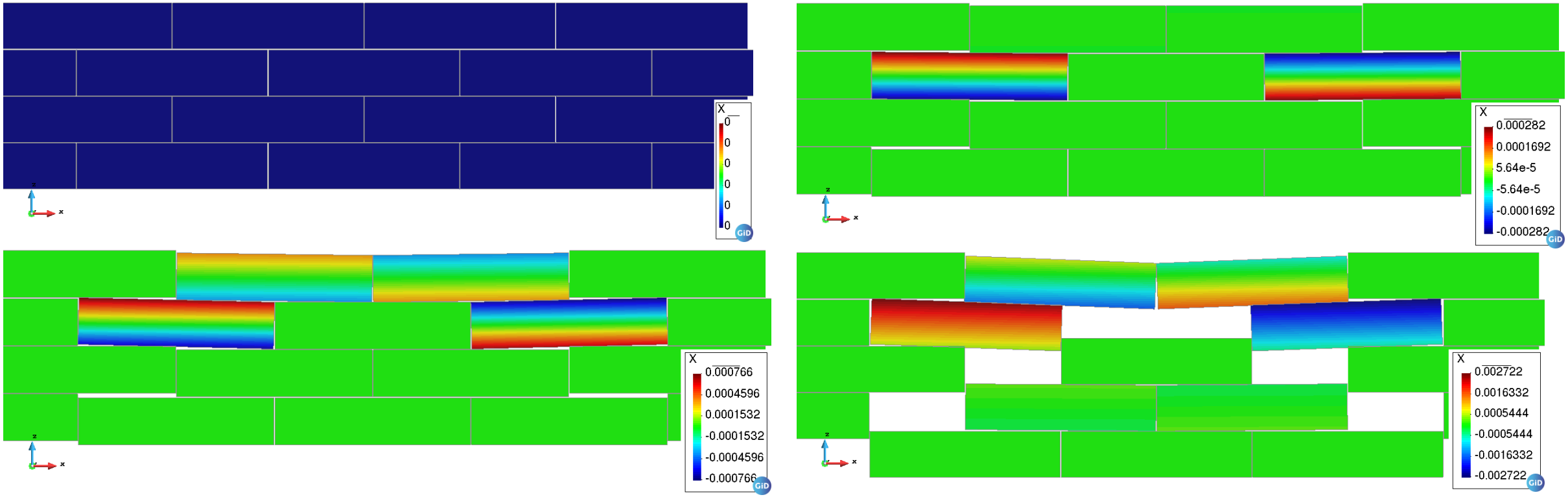
Si sono eseguiti svariati test per verificare la stabilità del codice.



Andamento test di accumulo piramidi in caduta in un contenitore.

DEM – BENCHMARK SIMMETRIA

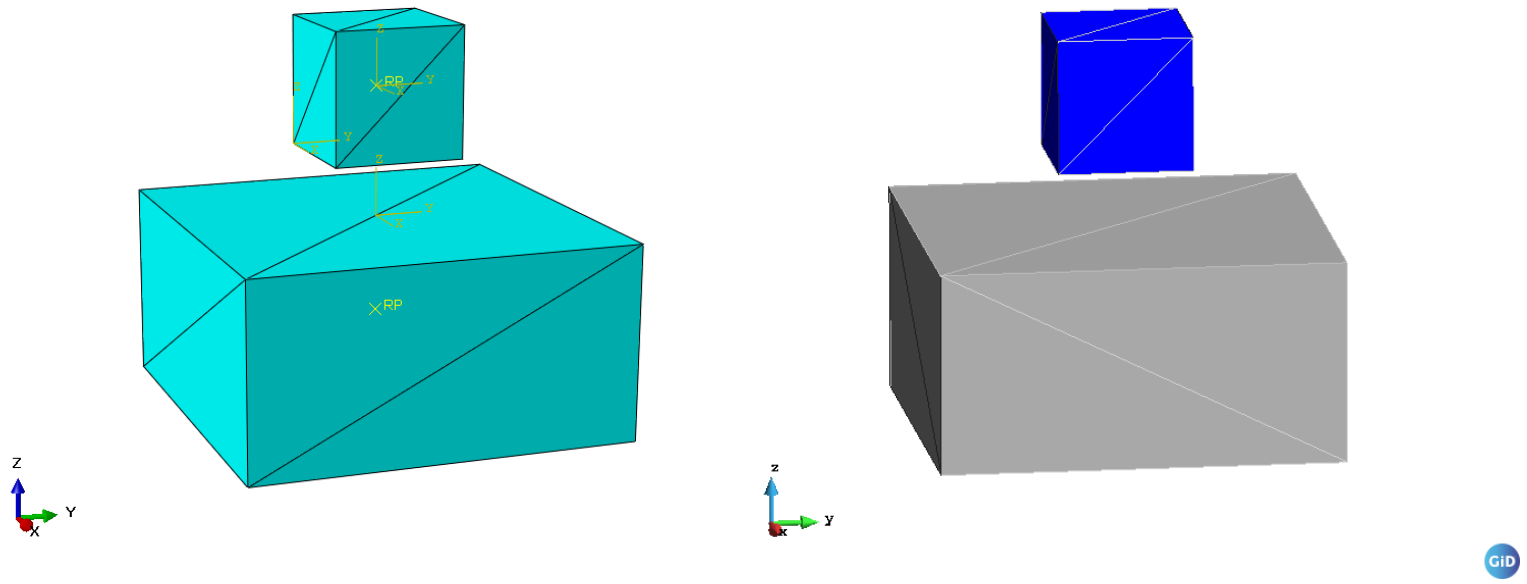
Si sono eseguiti svariati test per verificare la simmetria del codice.



Andamento test perdita di appoggio alla base di un muro di mattoni in condizioni simmetriche, contour agli spostamenti in X.

DEM – CALIBRAZIONE

Si è eseguita una calibrazione su un modello semplificato da confrontare con le soluzioni ottenute da Abaqus (programma commerciale FEM), al fine di capire se è possibile ottenere dei risultati che rappresentano correttamente la fisica del problema. Per il primo caso, si calibra un modello con soltanto traslazioni.



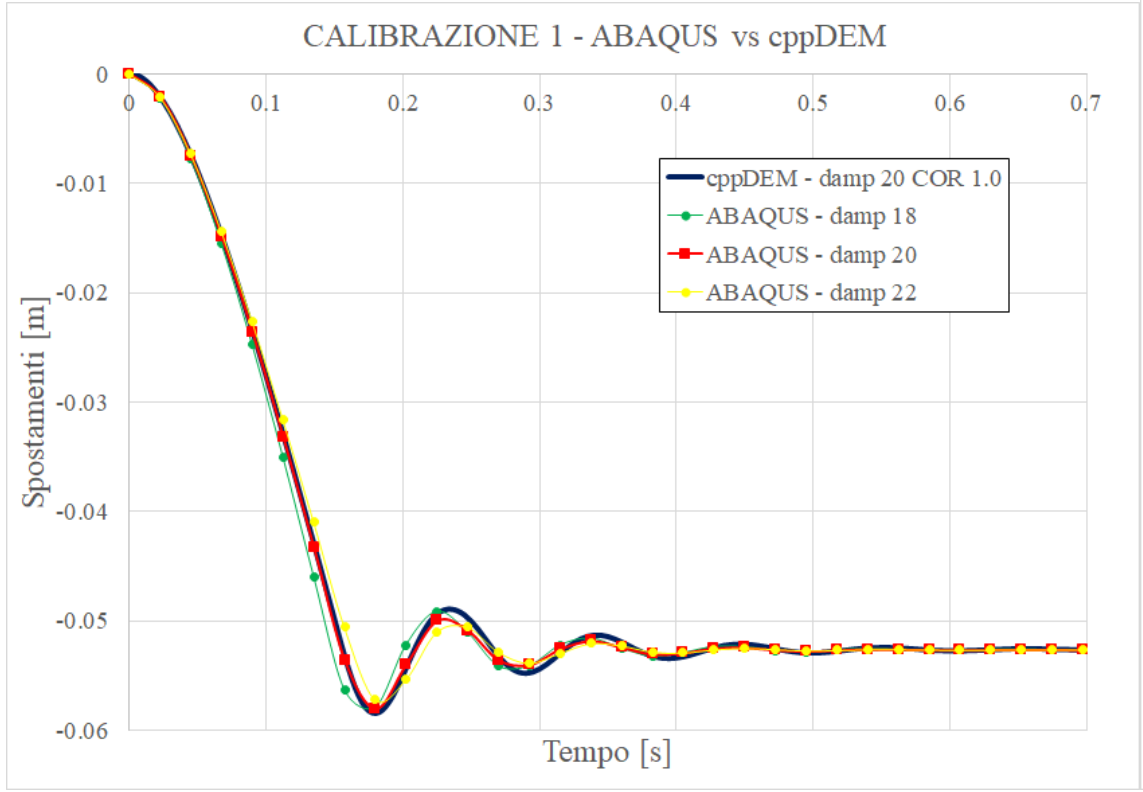
Elementi uguagliati nei 2 codici:

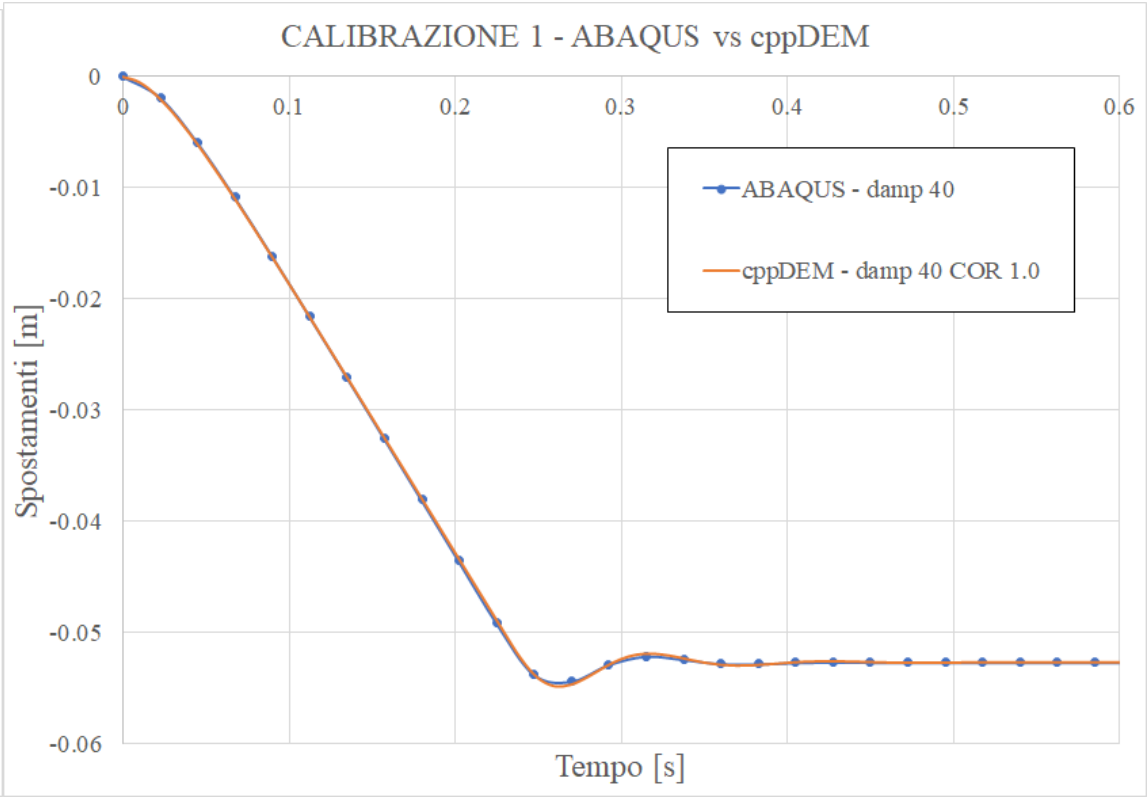
- Dimensioni geometria;
- Discretizzazione Mesh;
- Massa;
- Inerzia;
- Forza di gravità;
- Coefficiente di damping viscoso;
- Contatto di tipo Lineare;
- Coefficiente Penalty molle.

Modelli utilizzati in calibrazione, Abaqus (sx) e cppDEM nell'ambiente di post-processing di GiD (dx).

DEM – CALIBRAZIONE

Si riportano i grafici degli spostamenti in Z per confrontare i 2 risultati ottenuti dai programmi di calcolo:

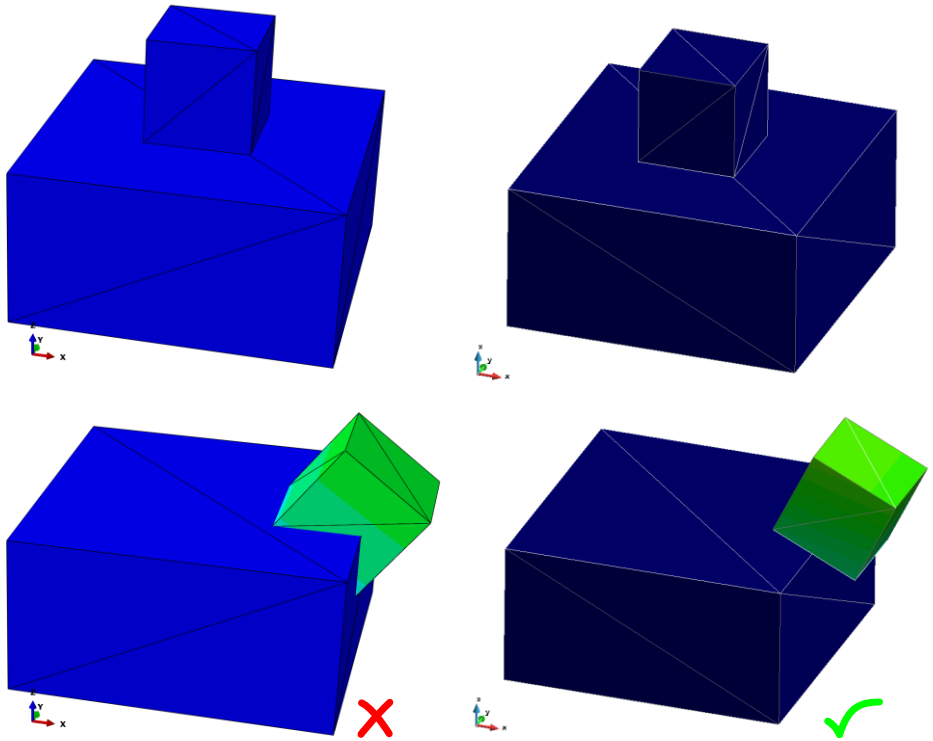




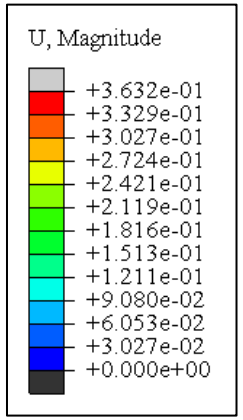
Grafici spostamenti in Z di confronto tra Abaqus e cppDEM. A sx il caso con damping 20, a dx il caso con damping 40.

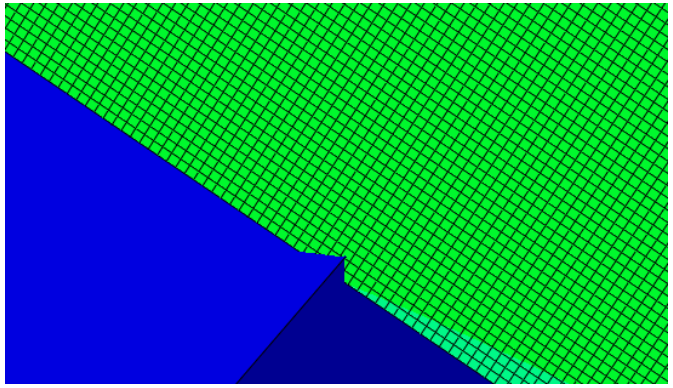
DEM – CALIBRAZIONE 2

Si è eseguita una seconda calibrazione su un modello con rotazioni. Si è riscontrata una difficoltà da parte di Abaqus nel calcolo dei contatti Spigolo-Spigolo, rendendo necessaria una discretizzazione fine della mesh dell'esaedro in movimento.

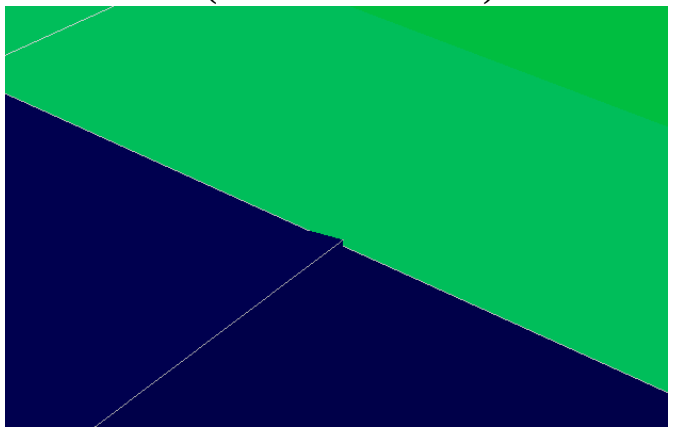


Compenetrazione eccessiva (Abaqus, sx) e compenetrazione corretta (cppDEM, dx) a parità di discretizzazione





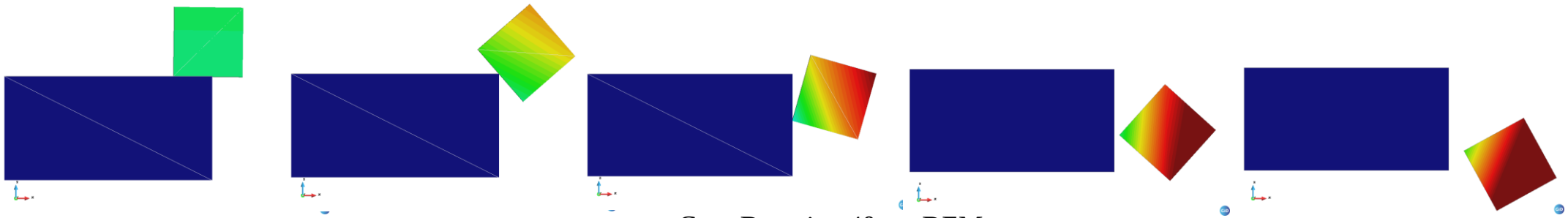
Compenetrazione a bordo esaedro in Abaqus (caso discretizzato)



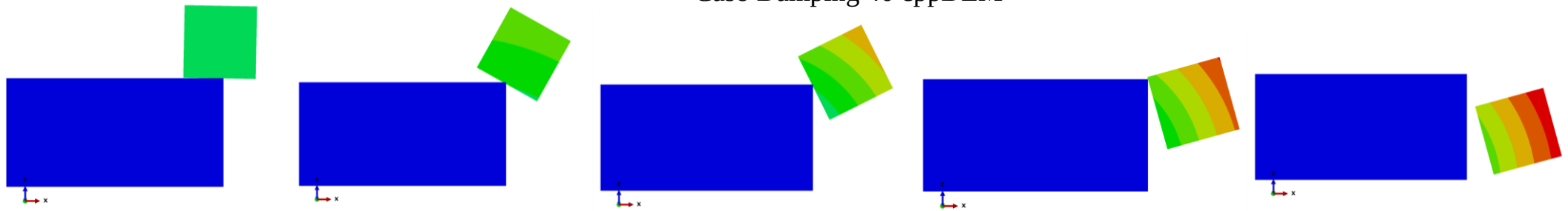
Compenetrazione a bordo esaedro nel cppDEM

DEM – CALIBRAZIONE 2

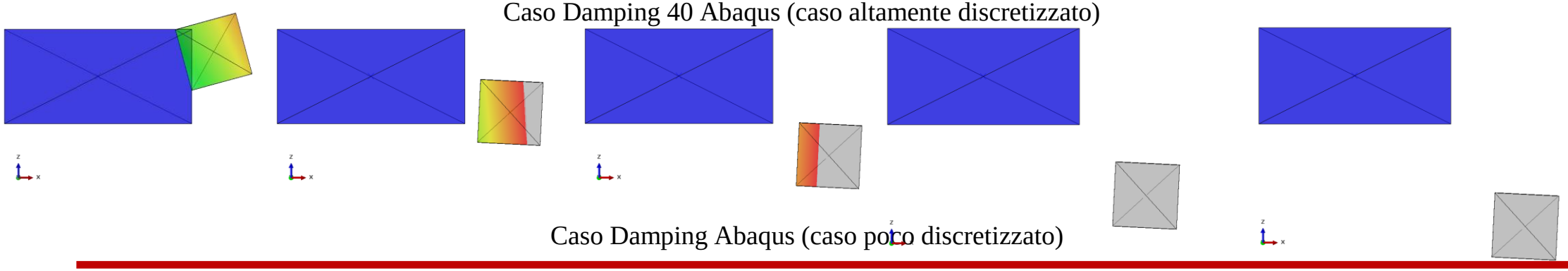
Si riportano gli andamenti delle soluzioni a confronto:



Caso Damping 40 cppDEM



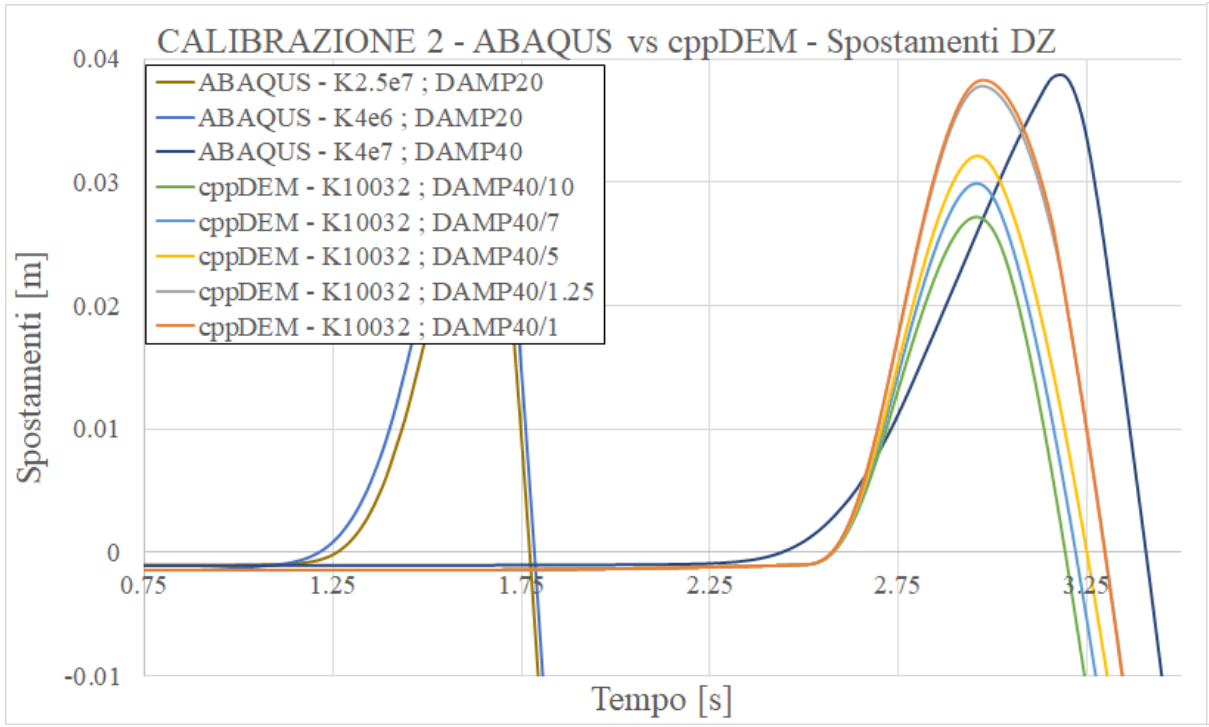
Caso Damping 40 Abaqus (caso altamente discretizzato)



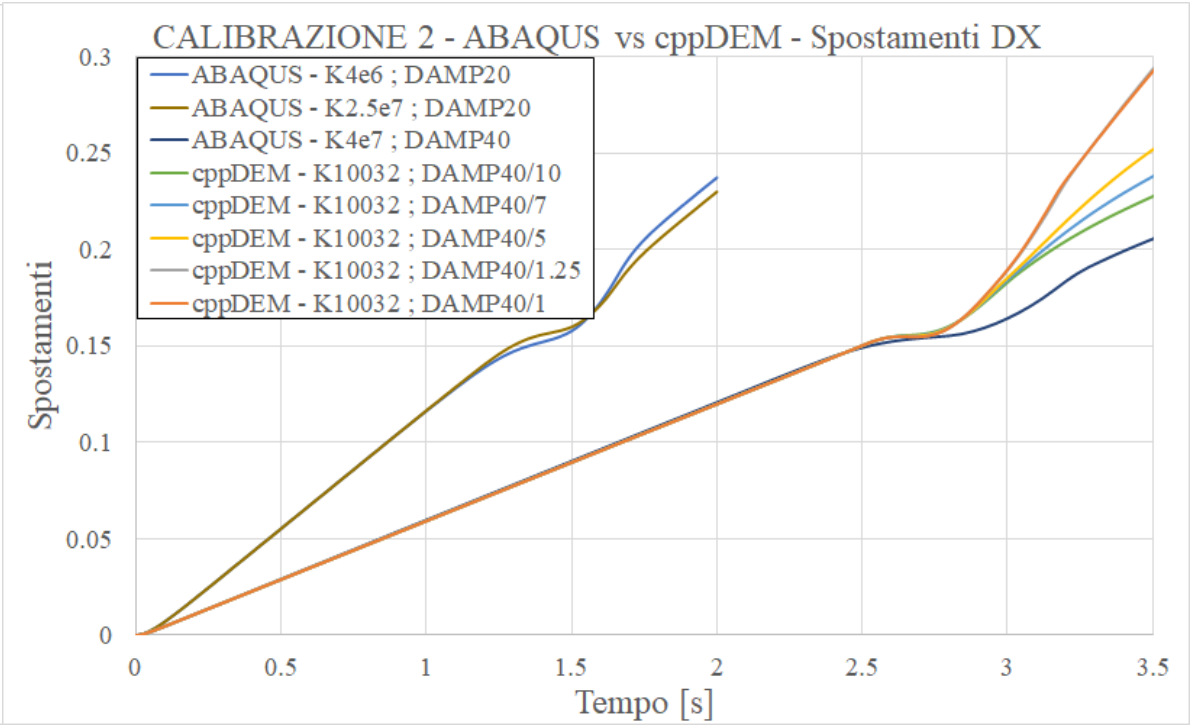
Caso Damping Abaqus (caso poco discretizzato)

DEM – CALIBRAZIONE 2

Dai grafici si può notare come il calcolo della rotazione tra i 2 programmi non fornisce risultati aderenti come nel caso della prima calibrazione. Questa differenza è stata attribuita principalmente al diverso calcolo del contatto di bordo (1 contatto cEE vs più contatti VF)



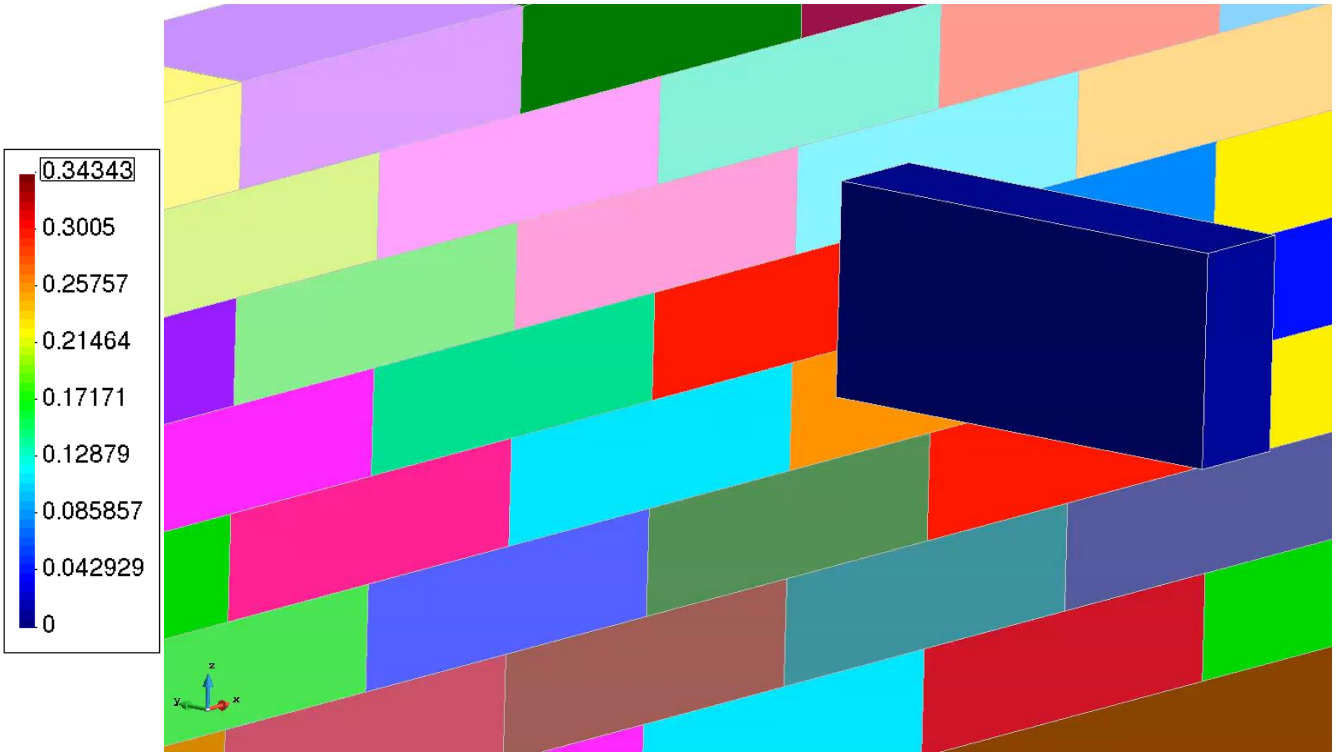
Spostamenti DZ Damping 40 cppDEM



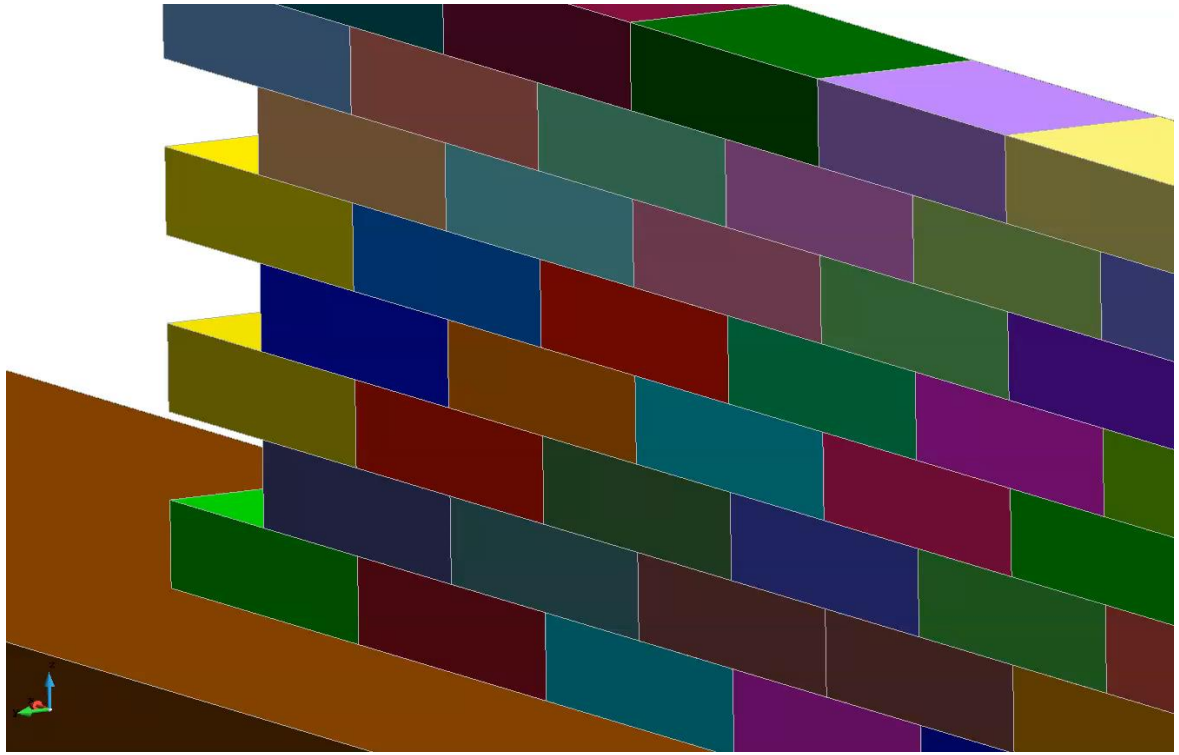
Spostamenti DX Damping 40 cppDEM

DEM – ALTRI CASI DI BENCHMARK

Si riporta la soluzione di altri test di carico più generali (simulazione muro colpito fuori piano), si può notare l'iniziale localizzazione dell'impatto in quanto l'assenza di attrito previene la diffusione delle forze:



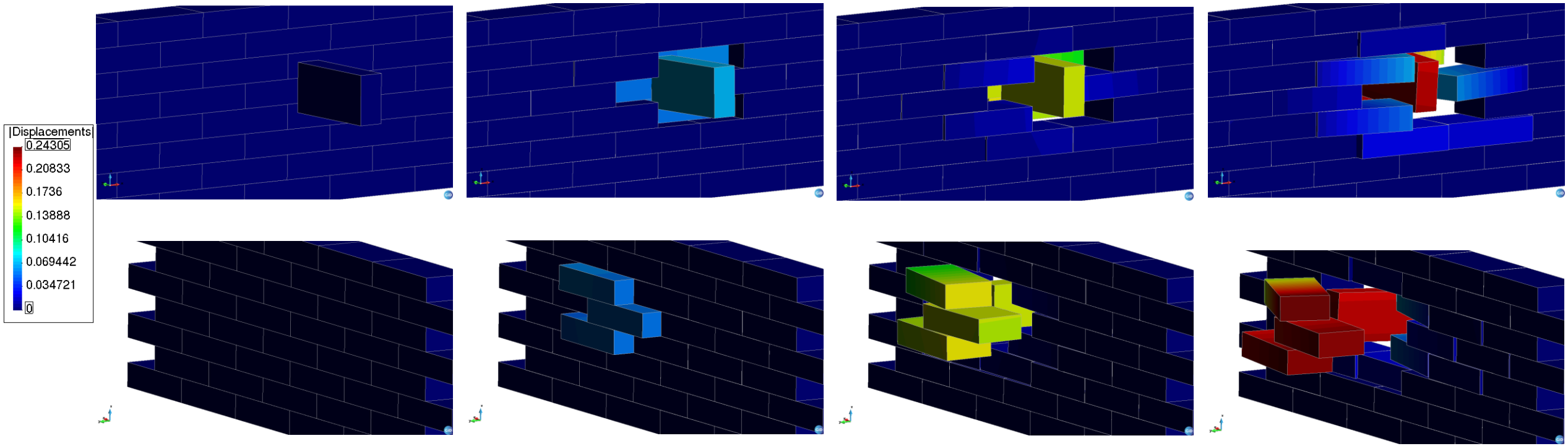
Simulazione muro colpito fuori piano, vista frontale.



Simulazione muro colpito fuori piano, vista posteriore.

DEM – ALTRI CASI DI BENCHMARK

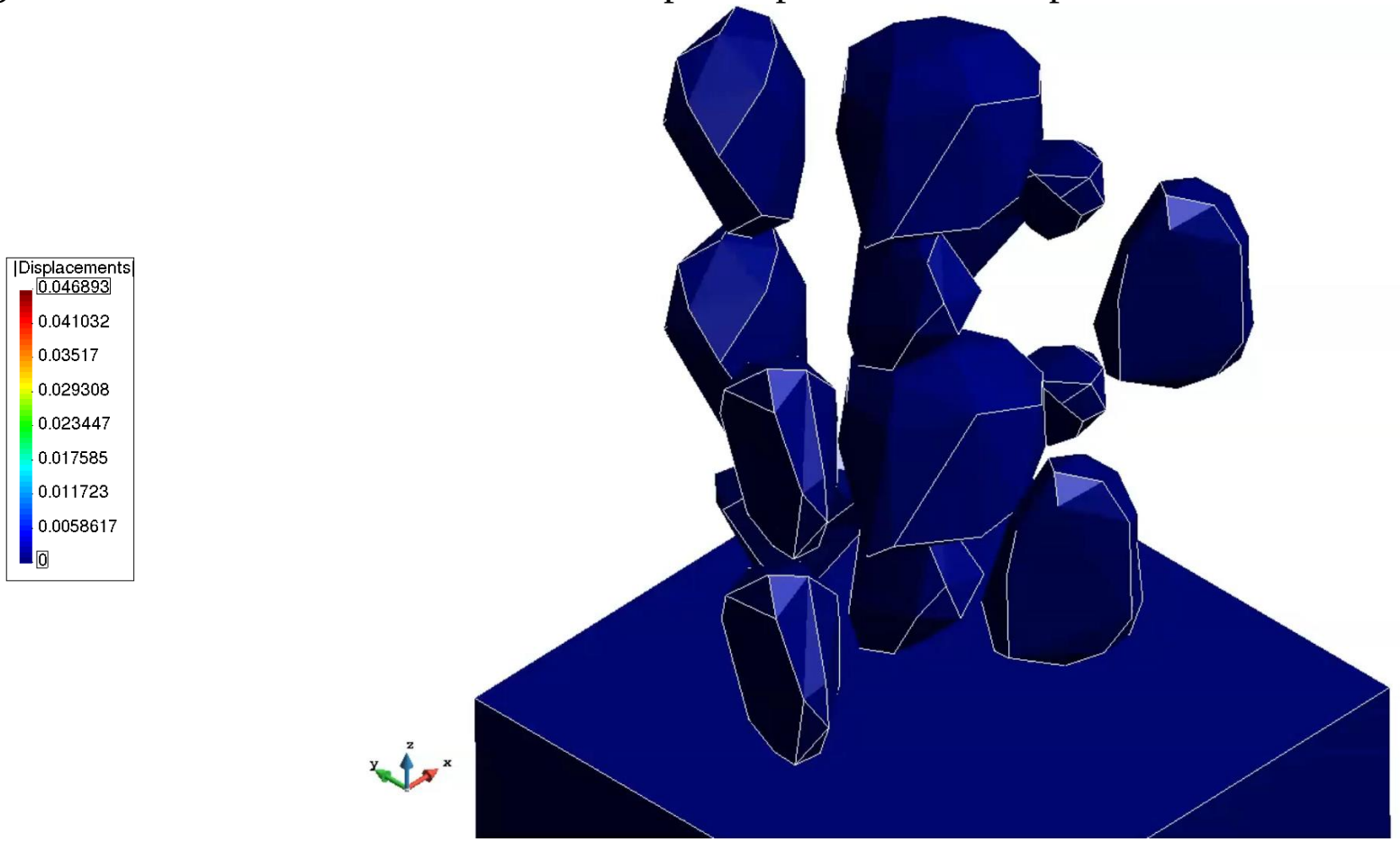
Si riporta la soluzione di altri test di carico più generali (simulazione muro colpito fuori piano), si può notare l'iniziale localizzazione dell'impatto in quanto l'assenza di attrito previene la diffusione delle forze:



Simulazione muro colpito fuori piano.

DEM – ALTRI CASI DI BENCHMARK

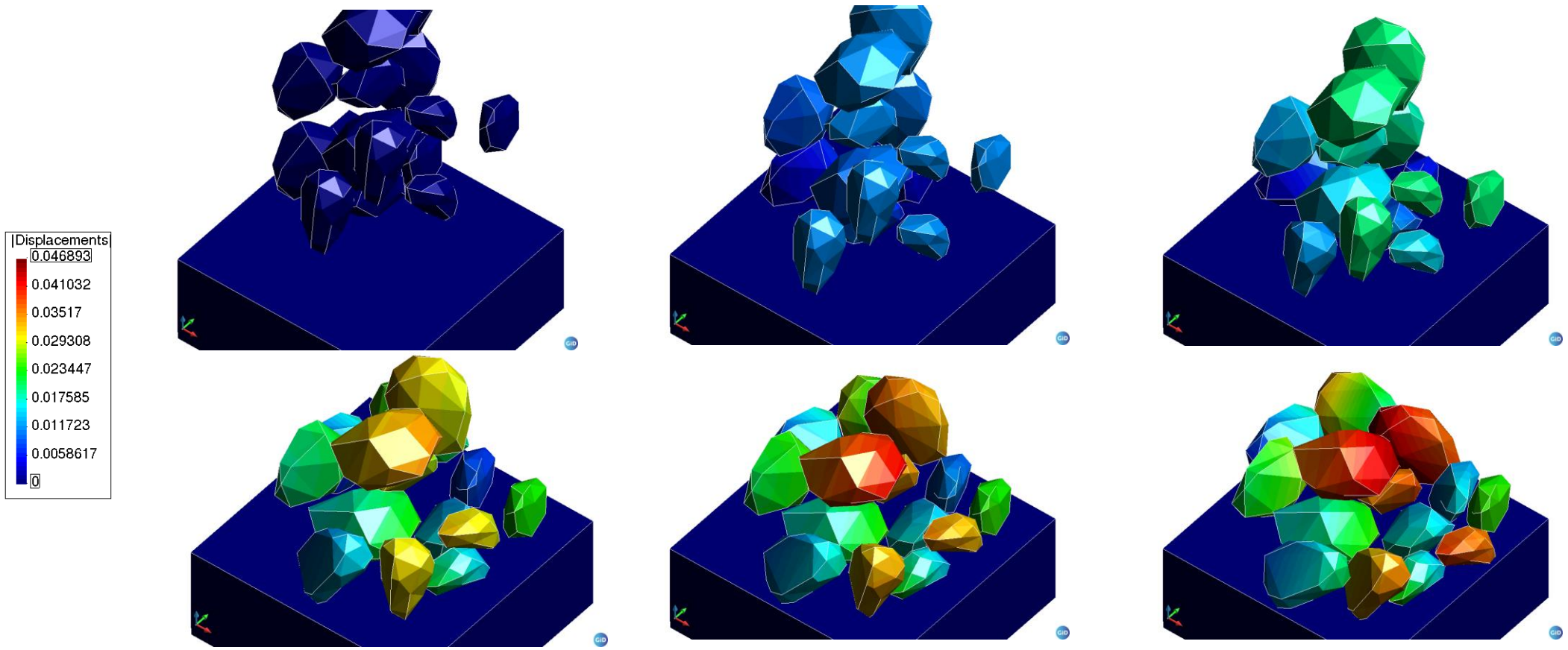
Si riporta la soluzione di altri test di carico più generali (simulazione test caduta inerti) con un numero di elementi maggiore e con una discretizzazione molto più impattante sui tempi di analisi:



Simulazione test caduta inerti.

DEM – ALTRI CASI DI BENCHMARK

Si riporta la soluzione di altri test di carico più generali (simulazione test caduta inerti) con un numero di elementi maggiore e con una discretizzazione molto più impattante sui tempi di analisi:



Simulazione test caduta inerti.

FINE

Si ringrazia per l’attenzione.

EXTRA – ELEMENTI POLIEDRICI PRO/CONTRO

Si sono utilizzati degli elementi Poliedrici, definiti tramite una nuvola di punti ed una matrice di incidenza facce-vertici. Si sono scelti questi elementi considerando eventuali vantaggi e svantaggi:

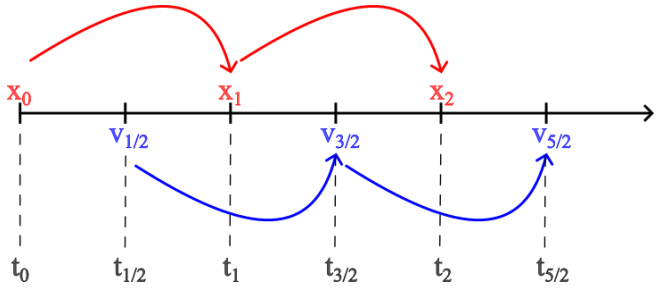
PRO:

- Massimo grado di definizione della particella;
- Facilità nel ricreare geometrie reali tramite file .stl esterni.
- Moto rotazionale automaticamente implementato tramite equazioni di equilibrio;

CONTRO:

- Geometria più complessa da gestire;
- Ricerca contatti più onerosa tra tutte le forme;
- Tempi di analisi più lunghi;

EXTRA – ALGORITMO LEAP-FROG



Andamento calcolo spostamenti e velocità.

Algoritmo modificato equazioni di Verlet, variante *Leap-Frog*:

- Calcolo accelerazioni step successivo tramite Eq. Equilibrio Dinamico:

$$\ddot{\boldsymbol{d}}_p = \frac{\sum \boldsymbol{F}_p^{contatto} + \boldsymbol{F}_p^g + \boldsymbol{D}_p}{m_p}$$

- Calcolo velocità a metà step successivo:

$$\boldsymbol{d}(t + \Delta t) = \boldsymbol{d}(t) + \Delta t\, \dot{\boldsymbol{d}}(t + \tfrac{\Delta t}{2})$$

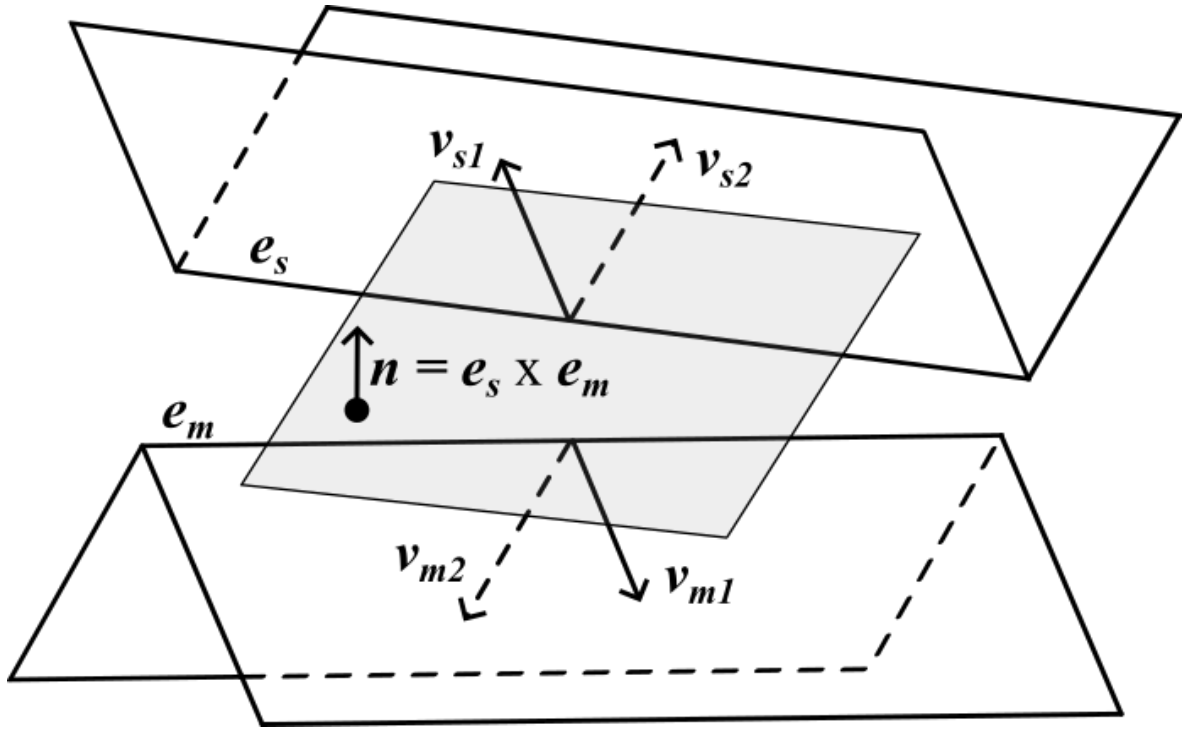
- Calcolo spostamenti step successivo:

$$\dot{\boldsymbol{d}}(t + \tfrac{\Delta t}{2}) = \dot{\boldsymbol{d}}(t - \tfrac{\Delta t}{2}) + \Delta t\, \ddot{\boldsymbol{d}}(t)$$

- (opzionale) Calcolo velocità al timestep corrente come media:

$$\dot{\boldsymbol{d}}(t) = \frac{\dot{\boldsymbol{d}}(t + \frac{\Delta t}{2}) + \dot{\boldsymbol{d}}(t - \frac{\Delta t}{2})}{2}$$

EXTRA – CONTROLLO CONTATTI EE



Vettori coinvolti nel controllo di validità contatti spigolo-spigolo