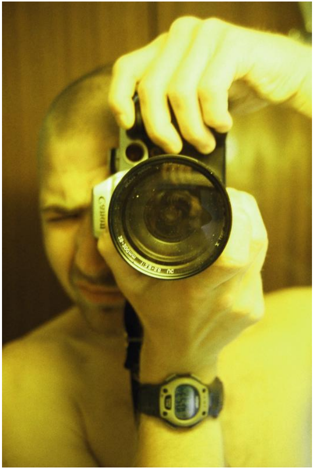
Discrete and hybrid models: Applications to concrete damage

Václav Šmilauer

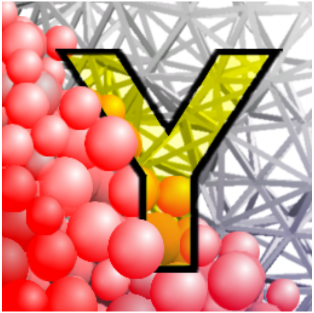
2 July 2007

About me



- PhD student, financed by French ministry of research this year.
- Enrolled both in Prague (Milan Jirásek) and Grenoble (Laurent Daudeville) — "doctorat en co-tutelle".
- Work focusing on Yade:
  - open-source platform for numerical calculations, since 2003;
  - continuing development, funded mostly by laboratory 3S-R in Grenoble.

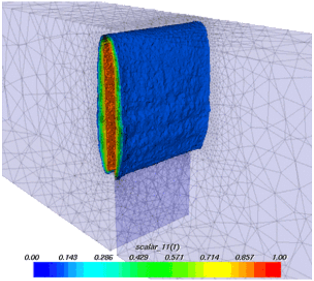
About me



http://yade.wikia.com

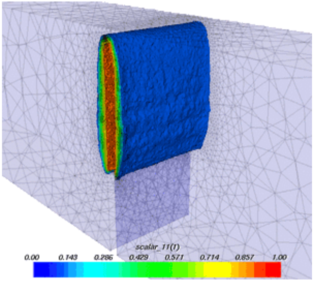
- PhD student, financed by French ministry of research this year.
- Enrolled both in Prague (Milan Jirásek) and Grenoble (Laurent Daudeville) — “doctorat en co-tutelle”.
- Work focusing on Yade:
  - open-source platform for numerical calculations, since 2003;
  - continuing development, funded mostly by laboratory 3S-R in Grenoble.

(Primarily) continuous models



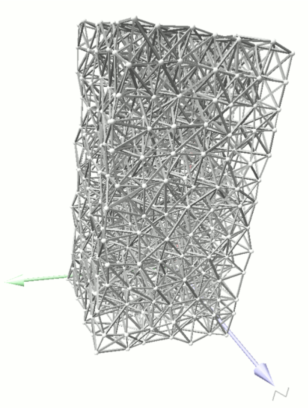
- Problem formulated in terms of differential equations – continuum mechanics.
- Displacement function u, found by numerical solution of boundary value problem.
- Discontinuities in u are an extension of the method.
- Strain undefined at discontinuity, “awkward” (sophisticated) methods.

(Primarily) continuous models



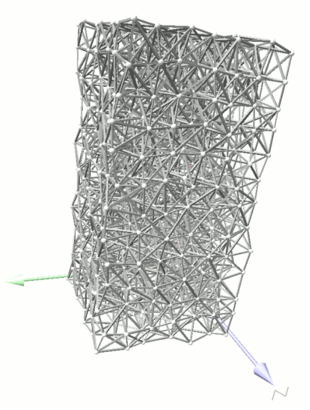
- Problem formulated in terms of differential equations – continuum mechanics.
- Displacement function u, found by numerical solution of boundary value problem.
- Discontinuities in u are an extension of the method.
- Strain undefined at discontinuity, “awkward” (sophisticated) methods.

(Primarily) discrete models



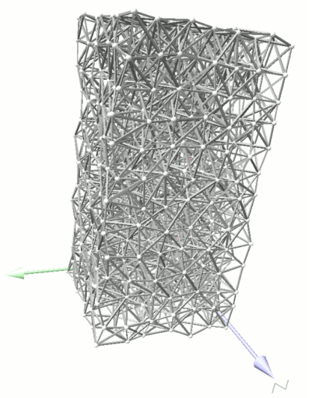
- Local equations determine global behavior numerically — element interactions.
- No integration necessary, more computationally intensive.
- Discontinuity description trivial.
- Continuity (cohesion) by linking elements.
- Discrete element method (DEM).
- Lattice models.

(Primarily) discrete models



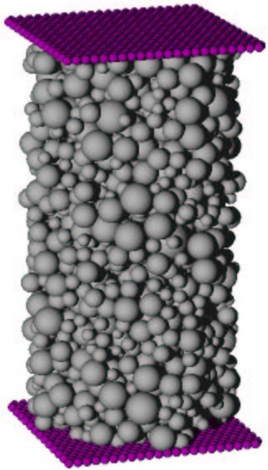
- Local equations determine global behavior numerically — element interactions.
- No integration necessary, more computationally intensive.
- Discontinuity description trivial.
- Continuity (cohesion) by linking elements.
- Discrete element method (DEM).
- Lattice models.

(Primarily) discrete models



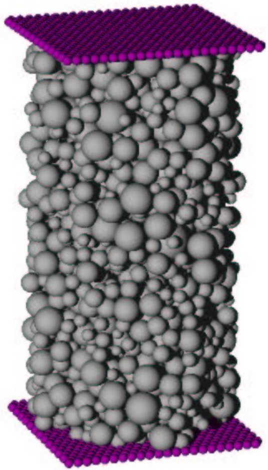
- Local equations determine global behavior numerically — element interactions.
- No integration necessary, more computationally intensive.
- Discontinuity description trivial.
- Continuity (cohesion) by linking elements.
- Discrete element method (DEM).
- Lattice models.

DEM background



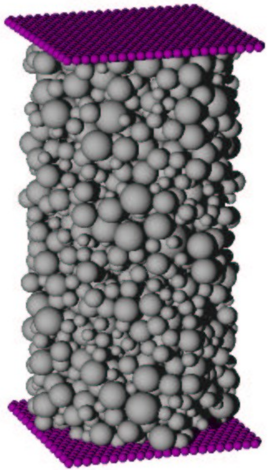
- elements are rigid bodies, motion governed by Newton's laws
- explicit integration in time
- "smooth" (pinball) vs. "non-smooth" (overlaps) DEM
- mechanics of granular media — Cundall, 1971 ("distinct element method" in 2D, spherical elements)
- molecular dynamics of gas (1980s)

DEM background



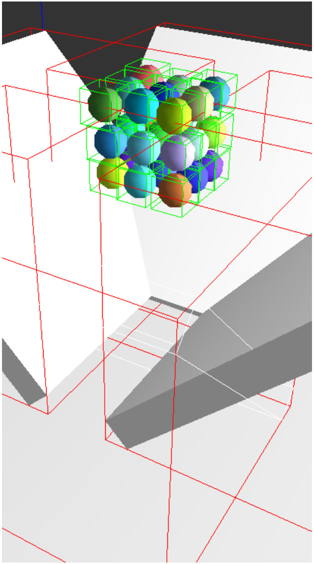
- elements are rigid bodies, motion governed by Newton's laws
- explicit integration in time
- "smooth" (pinball) vs. "non-smooth" (overlaps) DEM
- mechanics of granular media — Cundall, 1971 ("distinct element method" in 2D, spherical elements)
- molecular dynamics of gas (1980s)

DEM background



- elements are rigid bodies, motion governed by Newton's laws
- explicit integration in time
- "smooth" (pinball) vs. "non-smooth" (overlaps) DEM
- mechanics of granular media — Cundall, 1971 ("distinct element method" in 2D, spherical elements)
- molecular dynamics of gas (1980s)

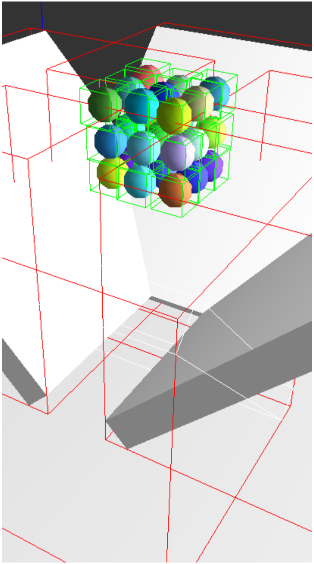
DEM simple example



How to calculate spheres falling through a funnel?

- Known element constants (m, $\mathbf{I}$, ...)
- and variables at $t = t_i$ ($\mathbf{x}$, $\mathbf{o}$, $\mathbf{v}$, ω, state parameters).
- Solve for variables at $t = t_{i+1} = t_i + \Delta t$.

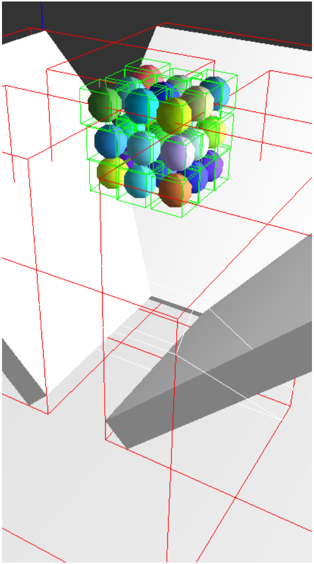
DEM simple example



How to calculate spheres falling through a funnel?

- Known element constants (m, $\mathbf{I}$, …)
- and variables at $t = t_i$ ($\mathbf{x}$, $\mathbf{o}$, $\mathbf{v}$, ω, state parameters).
- Solve for variables at $t = t_{i+1} = t_i + \Delta t$.

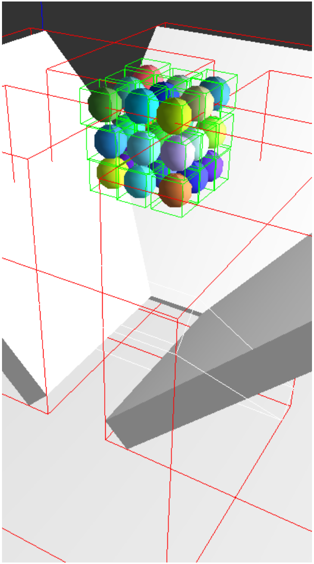
DEM simple example



How to calculate spheres falling through a funnel?

- Known element constants (m, $\mathbf{I}$, …)
- and variables at $t = t_i$ ($\mathbf{x}$, $\mathbf{o}$, $\mathbf{v}$, ω, state parameters).
- Solve for variables at $t = t_{i+1} = t_i + \Delta t$.

DEM simple example



How to calculate spheres falling through a funnel?

- Known element constants (m, $\mathbf{I}$, …)
- and variables at $t = t_i$ ($\mathbf{x}$, $\mathbf{o}$, $\mathbf{v}$, ω, state parameters).
- Solve for variables at $t = t_{i+1} = t_i + \Delta t$.

DEM iteration


```
void SphericalDEMSimulator::doOneIteration()
{
   // compute dt
   if (useTimeStepper)
     dt=computeDt(spheres,contacts);
   // detect potential collision
   sap.action(spheres,contacts);
   // detect real collision
   findRealCollision(spheres,contacts);
   // compute response
   computeResponse(spheres,contacts);
   // add damping
   addDamping(spheres);
   // apply response
   applyResponse(spheres);
   // time integration
   timeIntegration(spheres);
}
```

1. Calculate forces:
   - independent fields,
   - inter-element links,
   - element collisions.
2. Calculate acceleration from forces.
3. Integrate over Δt, $t = t_i + \Delta t$.
4. (Adjust Δt.)

DEM iteration

```
void SphericalDEMSimulator::doOneIteration()
{
   // compute dt
   if (useTimeStepper)
     dt=computeDt(spheres,contacts);
   // detect potential collision
   sap.action(spheres,contacts);
   // detect real collision
   findRealCollision(spheres,contacts);
   // compute response
   computeResponse(spheres,contacts);
   // add damping
   addDamping(spheres);
   // apply response
   applyResponse(spheres);
   // time integration
   timeIntegration(spheres);
}
```

1. Calculate forces:
   - independent fields,
   - inter-element links,
   - element collisions.
2. Calculate acceleration from forces.
3. Integrate over Δt, $t = t_i + \Delta t$.
4. (Adjust Δt.)

DEM iteration

```
void SphericalDEMSimulator::doOneIteration()
{
   // compute dt
   if (useTimeStepper)
     dt=computeDt(spheres,contacts);
   // detect potential collision
   sap.action(spheres,contacts);
   // detect real collision
   findRealCollision(spheres,contacts);
   // compute response
   computeResponse(spheres,contacts);
   // add damping
   addDamping(spheres);
   // apply response
   applyResponse(spheres);
   // time integration
   timeIntegration(spheres);
}
```

1. Calculate forces:
   - independent fields,
   - inter-element links,
   - element collisions.
2. Calculate acceleration from forces.
3. Integrate over Δt, $t = t_i + \Delta t$.
4. (Adjust Δt.)

DEM iteration


```
void SphericalDEMSimulator::doOneIteration()
{
   // compute dt
   if (useTimeStepper)
     dt=computeDt(spheres,contacts);
   // detect potential collision
   sap.action(spheres,contacts);
   // detect real collision
   findRealCollision(spheres,contacts);
   // compute response
   computeResponse(spheres,contacts);
   // add damping
   addDamping(spheres);
   // apply response
   applyResponse(spheres);
   // time integration
   timeIntegration(spheres);
}
```

1. Calculate forces:
   - independent fields,
   - inter-element links,
   - element collisions.
2. Calculate acceleration from forces.
3. Integrate over Δt, $t = t_i + \Delta t$.
4. (Adjust Δt.)

DEM iteration

```
void SphericalDEMSimulator::doOneIteration()
{
   // compute dt
   if (useTimeStepper)
     dt=computeDt(spheres,contacts);
   // detect potential collision
   sap.action(spheres,contacts);
   // detect real collision
   findRealCollision(spheres,contacts);
   // compute response
   computeResponse(spheres,contacts);
   // add damping
   addDamping(spheres);
   // apply response
   applyResponse(spheres);
   // time integration
   timeIntegration(spheres);
}
```

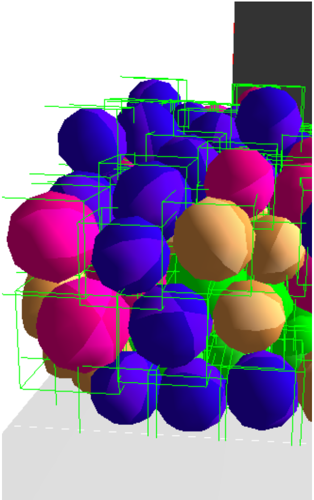
1. Calculate forces:
   - independent fields,
   - inter-element links,
   - element collisions.
2. Calculate acceleration from forces.
3. Integrate over Δt, $t = t_i + \Delta t$.
4. (Adjust Δt.)

DEM iteration


```
void SphericalDEMSimulator::doOneIteration()
{
   // compute dt
   if (useTimeStepper)
     dt=computeDt(spheres,contacts);
   // detect potential collision
   sap.action(spheres,contacts);
   // detect real collision
   findRealCollision(spheres,contacts);
   // compute response
   computeResponse(spheres,contacts);
   // add damping
   addDamping(spheres);
   // apply response
   applyResponse(spheres);
   // time integration
   timeIntegration(spheres);
}
```

1. Calculate forces:
   - independent fields,
   - inter-element links,
   - element collisions.
2. Calculate acceleration from forces.
3. Integrate over Δt, $t = t_i + \Delta t$.
4. (Adjust Δt.)

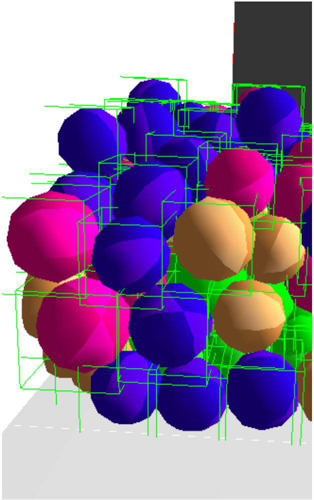
Collision detection



Big research issue in applied mathematics, trivial approach is $O(n^2)$. Better approach:

1. Replace each element by its AABB (Axis-Aligned Bounding Box).
2. Sort x, y, z min-max arrays independently.
3. Overlaps on all coordinates are *collision candidates*.
4. Candidates tested geometrically for collision.

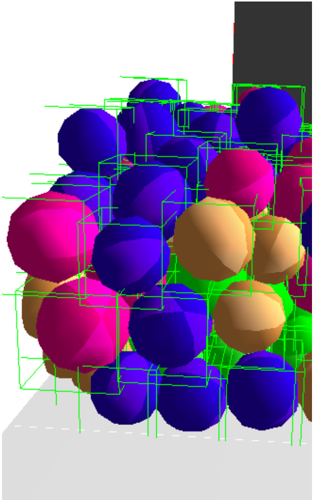
Collision detection



Big research issue in applied mathematics, trivial approach is $O(n^2)$. Better approach:

1. Replace each element by its AABB (Axis-Aligned Bounding Box).
2. Sort x, y, z min-max arrays independently.
3. Overlaps on all coordinates are *collision candidates*.
4. Candidates tested geometrically for collision.

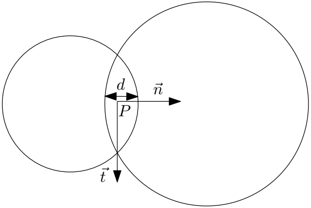
Collision detection

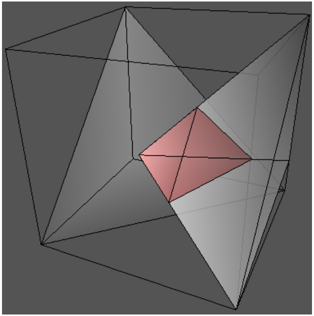


Big research issue in applied mathematics, trivial approach is $O(n^2)$. Better approach:

1. Replace each element by its AABB (Axis-Aligned Bounding Box).
2. Sort x, y, z min-max arrays independently.
3. Overlaps on all coordinates are *collision candidates*.
4. Candidates tested geometrically for collision.

Collision detection



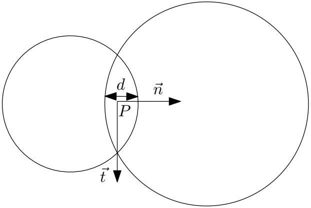


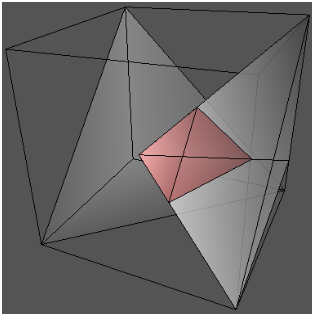
Geometry:

- P, n, t, d for spheres (trivial).
- Complicated for other shapes (e.g. tetrahedra: C, V, $\mathbf{I}$).
- Combinations: sphere with tetrahedron, parallelepiped, ...

Forces are yet to be found.

Collision detection



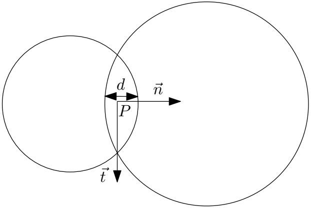


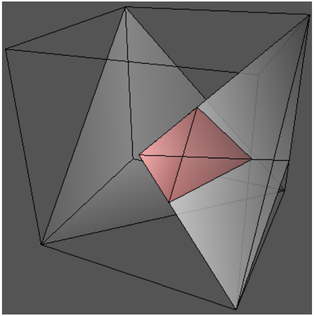
Geometry:

- P, n, t, d for spheres (trivial).
- Complicated for other shapes (e.g. tetrahedra: C, V, **I**).
- Combinations: sphere with tetrahedron, parallelepiped, ...

Forces are yet to be found.

Collision detection



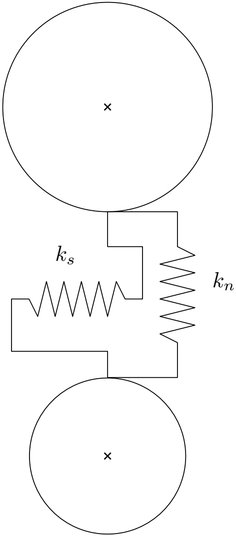


Geometry:

- P, n, t, d for spheres (trivial).
- Complicated for other shapes (e.g. tetrahedra: C, V, $\mathbf{I}$).
- Combinations: sphere with tetrahedron, parallelepiped, ...

Forces are yet to be found.

Physical laws



The most simple model:

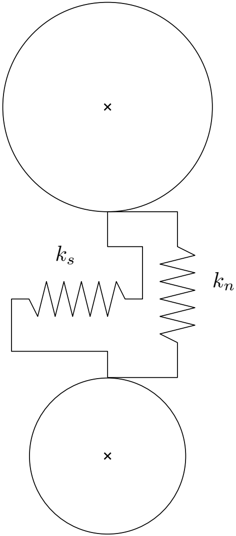
- $F_n = k_n d$, $\Delta F_s = k_s \Delta u_t$ (incremental).
- Fracture with Coulomb criterion $\max F_s = F_n \tan\phi$.

How to determine k_n, k_s from macroscopic characteristics?

- Simplistically for sphere $F = Ku$, $F = \sigma S = E(1 - d/2r)\pi r^2$ $(d \ll r)$
- Really used formulas: coefficients without physical meaning.

$\longrightarrow$ Model calibration.

Physical laws



The most simple model:

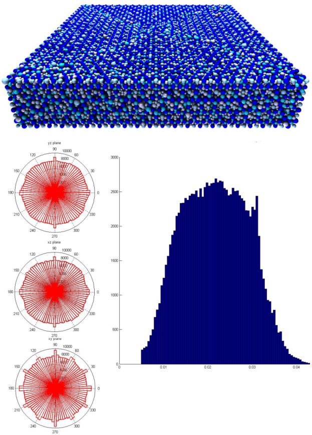
- $F_n = k_n d$, $\Delta F_s = k_s \Delta u_t$ (incremental).
- Fracture with Coulomb criterion $\max F_s = F_n \tan \phi$.

How to determine k_n, k_s from macroscopic characteristics?

- Simplistically for sphere $F = Ku$, $F = \sigma S = E(1 - d/2r)\pi r^2$ $(d \ll r)$
- Really used formulas: coefficients without physical meaning.

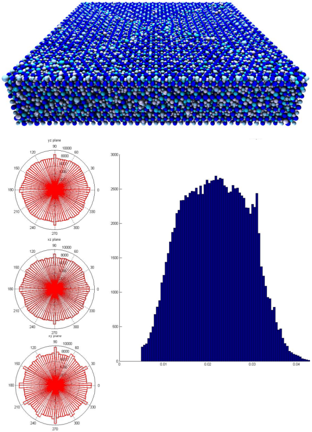
$\longrightarrow$ Model calibration.

Mesh generation



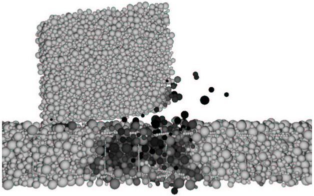
- Requirements: isotropy, high coordination number and compacity, size distribution.
- Regular packing leads to anisotropic behavior.
- Dynamic methods: gravity, growing spheres. Slow.
- Jean-François Durier: Geometric method based on tetrahedral mesh:
  - Leverages existing FEM meshers — arbitrary shapes.
  - Very fast with excellent parameters.

Mesh generation



- Requirements: isotropy, high coordination number and compacity, size distribution.
- Regular packing leads to anisotropic behavior.
- Dynamic methods: gravity, growing spheres. Slow.
- Jean-François Durier: Geometric method based on tetrahedral mesh:
  - Leverages existing FEM meshers — arbitrary shapes.
  - Very fast with excellent parameters.

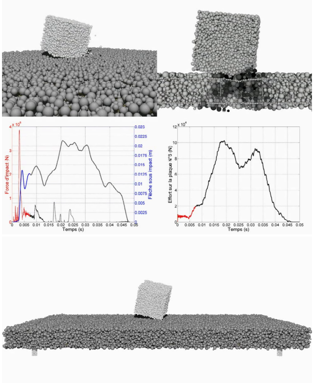
DEM concrete fracture (Hentz)



Modélisation d'une Structure en Béton Armé Soumise à un Choc par la Méthode des Éléments Discrets (PhD thesis of Sebastian Hentz, 2003)

- Dropping reinforced concrete cube on reinforced concrete slab.
- Concrete elements not "physical" (matrix, inclusions).
- Reinforcement modelled by special elements (including plastification).
- Parameters calibrated on basic setups, not *ex post* — predictive value.

DEM concrete fracture, results

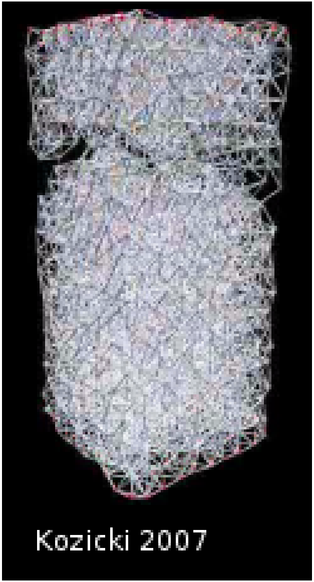


- 80+110 thousand elements (reinforcement + concrete)
- Comparison with instrumented experiment at different limit states.

+ Good results (e.g. displacements ±10%).

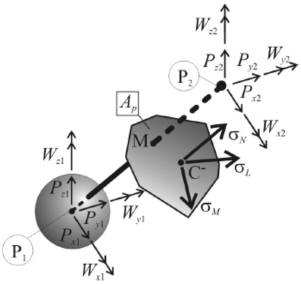
– Long calculation

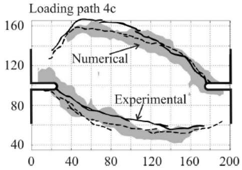
Lattice overview



- Replaces continuum by arrangement of 1D elements (trusses, rods, beams).
- Nodes may carry inertia mass (dynamic) or not.
- Irregular meshes, less sensitive to degenerate geometry.
- Voronoï tesselation / Delaunay triangulation.

Lattice beam model (Cusatis & co.)





(Nooru-Mohamed)

- Meso-scale lattice *beam* model (matrix, inclusions).
- Constituive law with damage, fracture, plasticity.
- Elaborate beam properties based on geometry of the respective Voronoï cell.
- Good match in tensile as well as compressive (usual weak point of lattices) loading.

Method considerations

FEM

+ Efficient and easy for undamaged continuum.
– Difficult discontinuity description.
→ Undamaged zone.

Lattice

+ No collision detection necessary.
– No volumetric information.
→ Fragmenting zone interior.

DEM

+ Compressive links created during simulation.
– Collisions: computationally expensive.
→ Highly fragmented, collapsing zones.
→ Colliding boundaries or detached zones.

Method considerations

FEM

+ Efficient and easy for undamaged continuum.
− Difficult discontinuity description.
→ Undamaged zone.

Lattice

+ No collision detection necessary.
− No volumetric information.
→ Fragmenting zone interior.

DEM

+ Compressive links created during simulation.
− Collisions: computationally expensive.
→ Highly fragmented, collapsing zones.
→ Colliding boundaries or detached zones.

Method considerations

FEM

- \+ Efficient and easy for undamaged continuum.
- – Difficult discontinuity description.
- → Undamaged zone.

Lattice

- \+ No collision detection necessary.
- – No volumetric information.
- → Fragmenting zone interior.

DEM

- \+ Compressive links created during simulation.
- – Collisions: computationally expensive.
- → Highly fragmented, collapsing zones.
- → Colliding boundaries or detached zones.

Method considerations

FEM

- + Efficient and easy for undamaged continuum.
- − Difficult discontinuity description.
- → Undamaged zone.

Lattice

- + No collision detection necessary.
- − No volumetric information.
- → Fragmenting zone interior.

DEM

- + Compressive links created during simulation.
- − Collisions: computationally expensive.
- → Highly fragmented, collapsing zones.
- → Colliding boundaries or detached zones.

Method considerations

FEM

+ Efficient and easy for undamaged continuum.

– Difficult discontinuity description.

→ Undamaged zone.

Lattice

+ No collision detection necessary.

– No volumetric information.

→ Fragmenting zone interior.

DEM

+ Compressive links created during simulation.

– Collisions: computationally expensive.

→ Highly fragmented, collapsing zones.

→ Colliding boundaries or detached zones.

Method considerations

FEM

+ Efficient and easy for undamaged continuum.
– Difficult discontinuity description.
→ Undamaged zone.

Lattice

+ No collision detection necessary.
– No volumetric information.
→ Fragmenting zone interior.

DEM

+ Compressive links created during simulation.
– Collisions: computationally expensive.
→ Highly fragmented, collapsing zones.
→ Colliding boundaries or detached zones.

Method considerations

FEM

+ Efficient and easy for undamaged continuum.

– Difficult discontinuity description.

→ Undamaged zone.

Lattice

+ No collision detection necessary.

– No volumetric information.

→ Fragmenting zone interior.

DEM

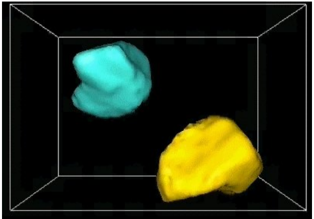
+ Compressive links created during simulation.

– Collisions: computationally expensive.

→ Highly fragmented, collapsing zones.

→ Colliding boundaries or detached zones.

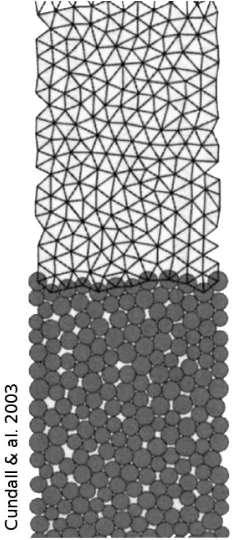
FEM inside DEM



Munjiza, 2004

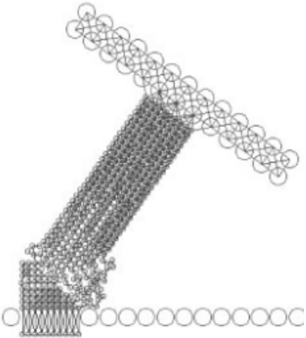
- Particles in DEM are themselves FEM domains.
- Interest: reduces computational expenses wrt pure DEM for unfractures parts.
- Allows for dynamical states — in pure FEM, leads to statically under-determined states.
- More difficult collision detection (mesh — mesh).
- For non-predetermined fracture, FEM→DEM transition (via crack modeling and tracing) must be provided.

FEM next to DEM



- Part of domain expected to break is DEM, the rest is FEM.
- Reduces computation wrt pure DEM.
- Must know fracturing (DEM) domain beforehand.
- Parameters must be tuned to have similar elastic behavior in both domains.
- The domain interface reflects waves (remedy: overlap zone — E. Frangin).

DEM with lattice



Sun &al, 2003

- Nodes are also DEM elements.
- Or: boundary nodes are DEM (collision), insert equivalent DEM element as needed.
- Preserves volume when fractured; pure lattice collapses.

References

- Sebastian Hentz, *Modélisation d'une Structure en Béton Armé Soumise à un Choc par la Méthode des Éléments Discrets*, PhD thesis, Grenoble 2003.
- Janek Kozicki, *Application of Discrete Models to Describe the Fracture Process in Brittle Materials*, PhD thesis, Gdańsk 2007.
- Gianluca Cusatis &al., *Confinement-Shear Lattice CSL Model for Fracture Propagation in Concrete*, Confinement-Shear Lattice CSL Model for Fracture Propagation in Concrete 2006, 7154—7171.
- Ante Munjiza, *Combined Finite-Discrete Element Method*, Wiley, 2004.
- Limin Sun &al., *Application of extended distinct element method with latticemodel to collapse analysis of RC bridges*, Earthquake Engng Struct. Dyn. 2003, 1217–1236.