



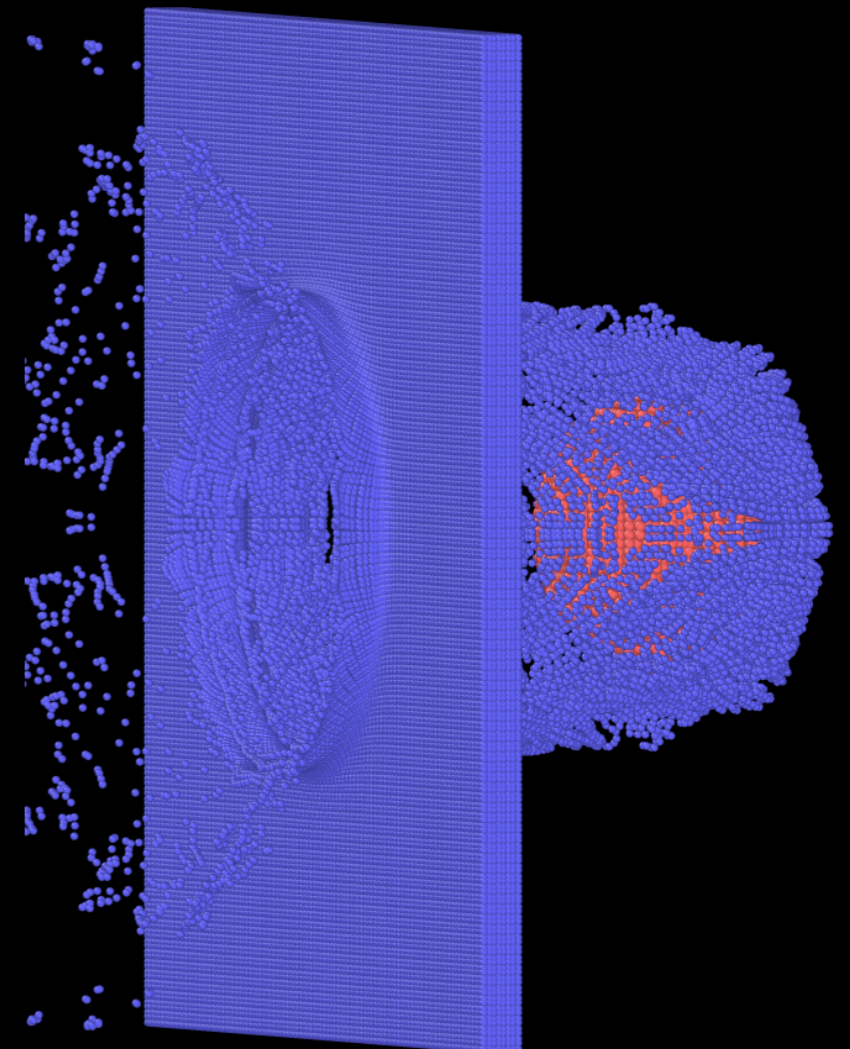
# Material Point Method Theory and Applications

Phu Nguyen

Department of Civil Engineering  
Monash University

[phu.nguyen@monash.edu](mailto:phu.nguyen@monash.edu)

<http://nvinhphu.wixsite.com/mysite>



# Collaborators and funding agencies

---



**Alban de Vaucorbeil**  
**Deakin University**



**Sina Sinaie**  
**Melbourne University**



**Tushar Mandal**  
**PhD candidate**



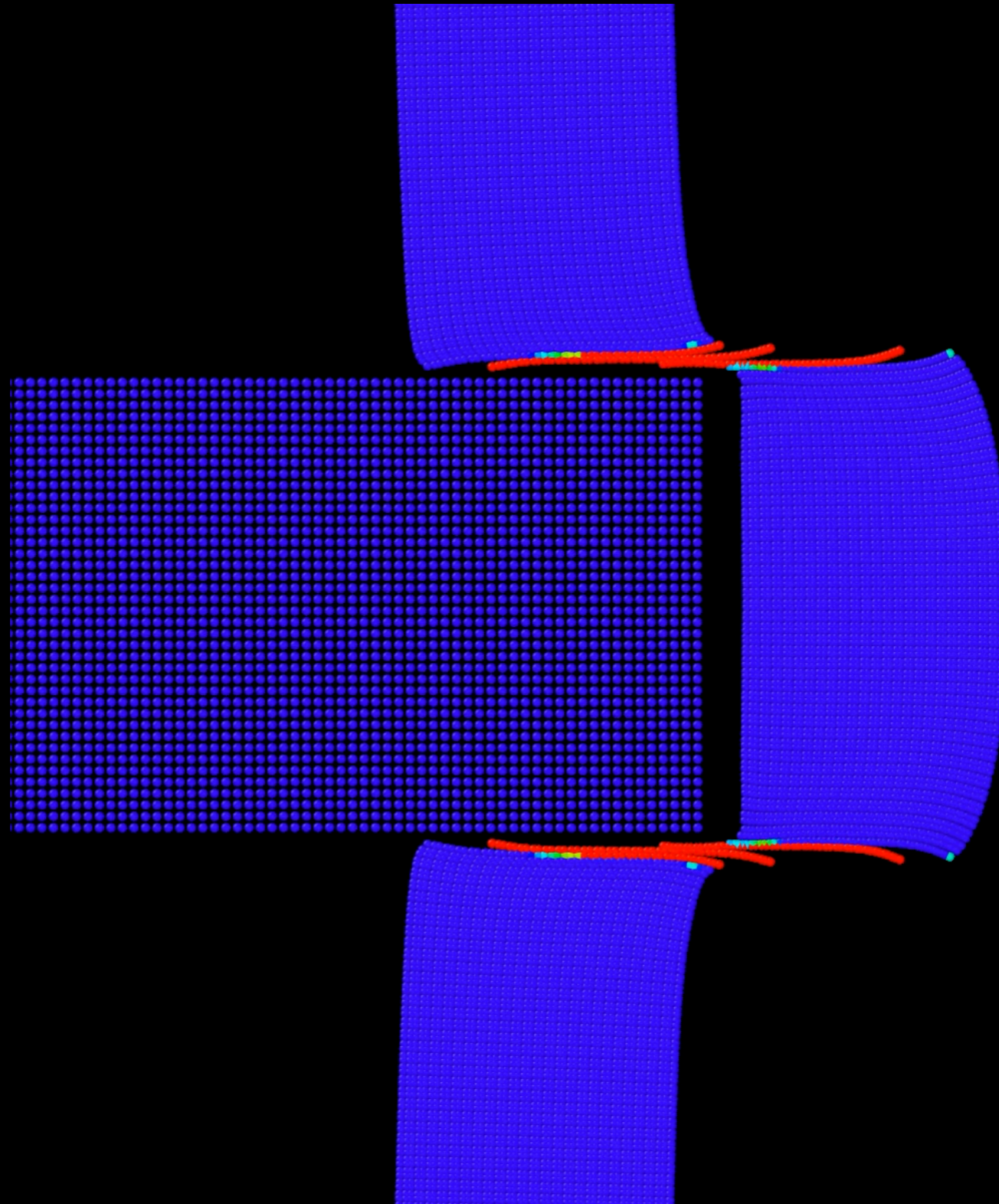
**Australian Government**

---

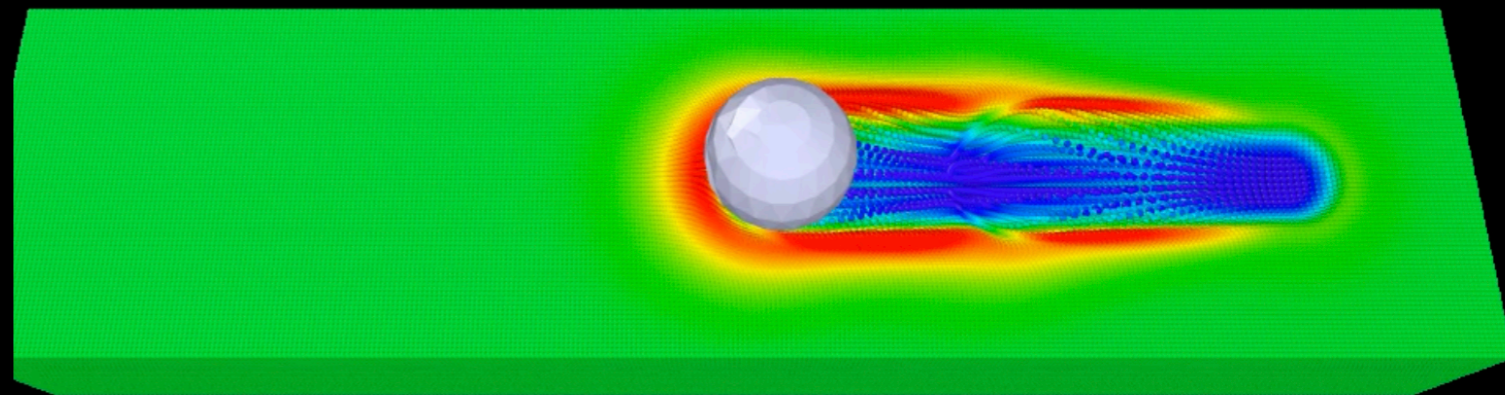
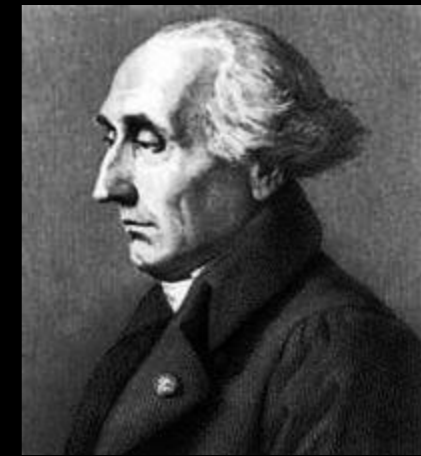
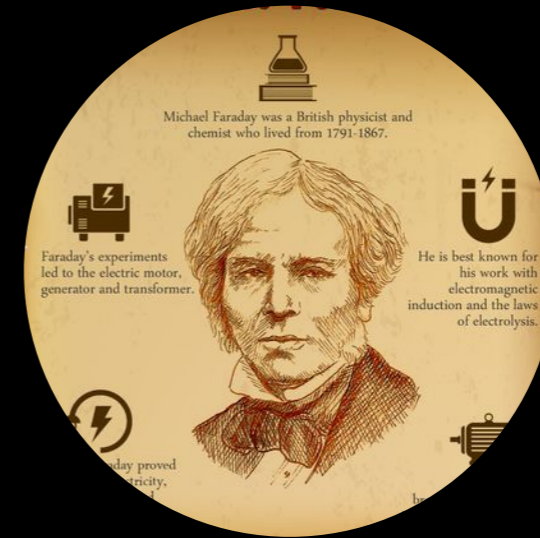
**Australian Research Council**

# The problem: deformation/fracture of solids

---



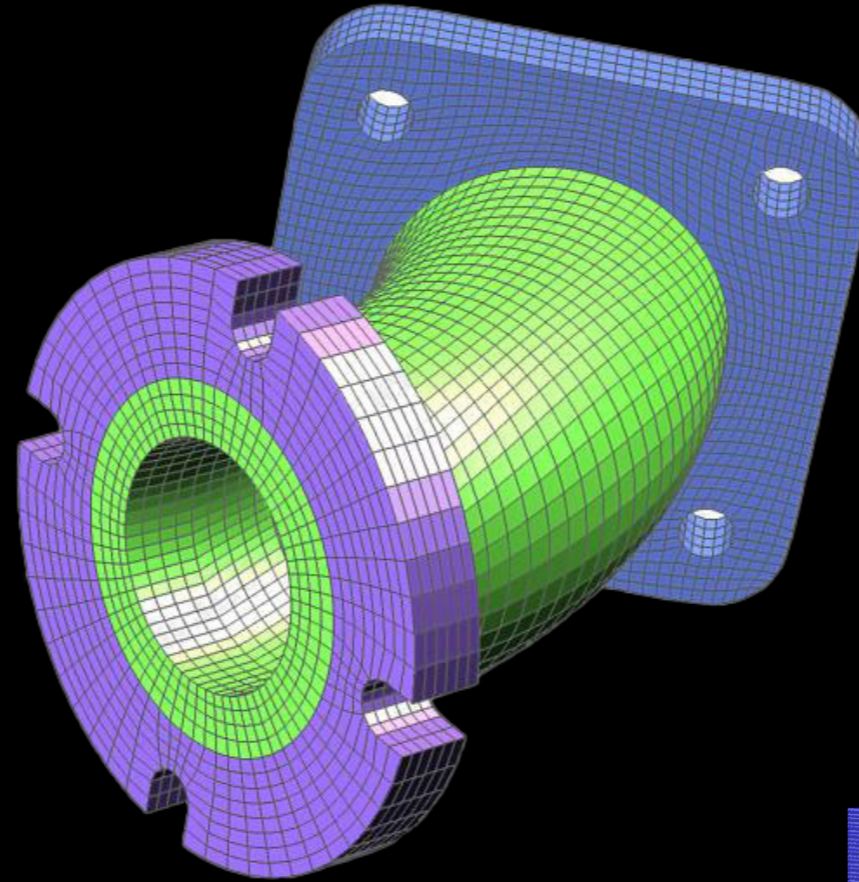
# Deformation/fracture of solids: approaches



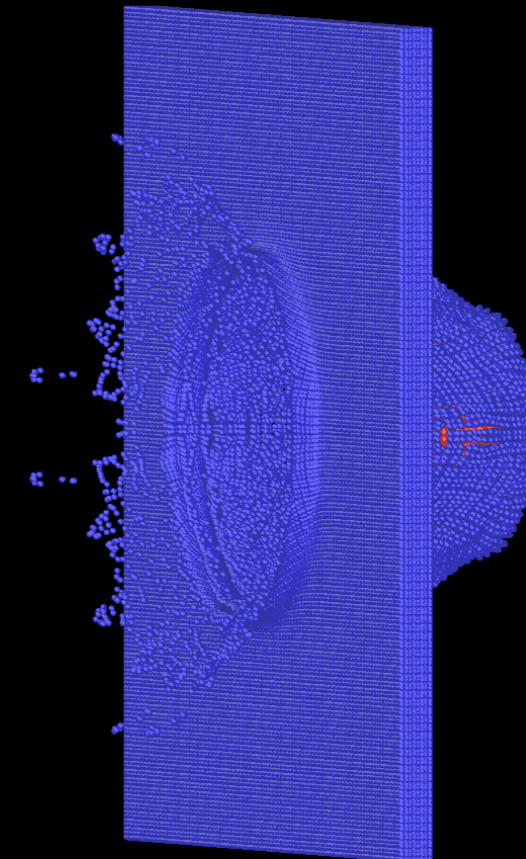
# What computational engineer/scientist do

Balance equations  
Constitutive models

$$\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} = f$$

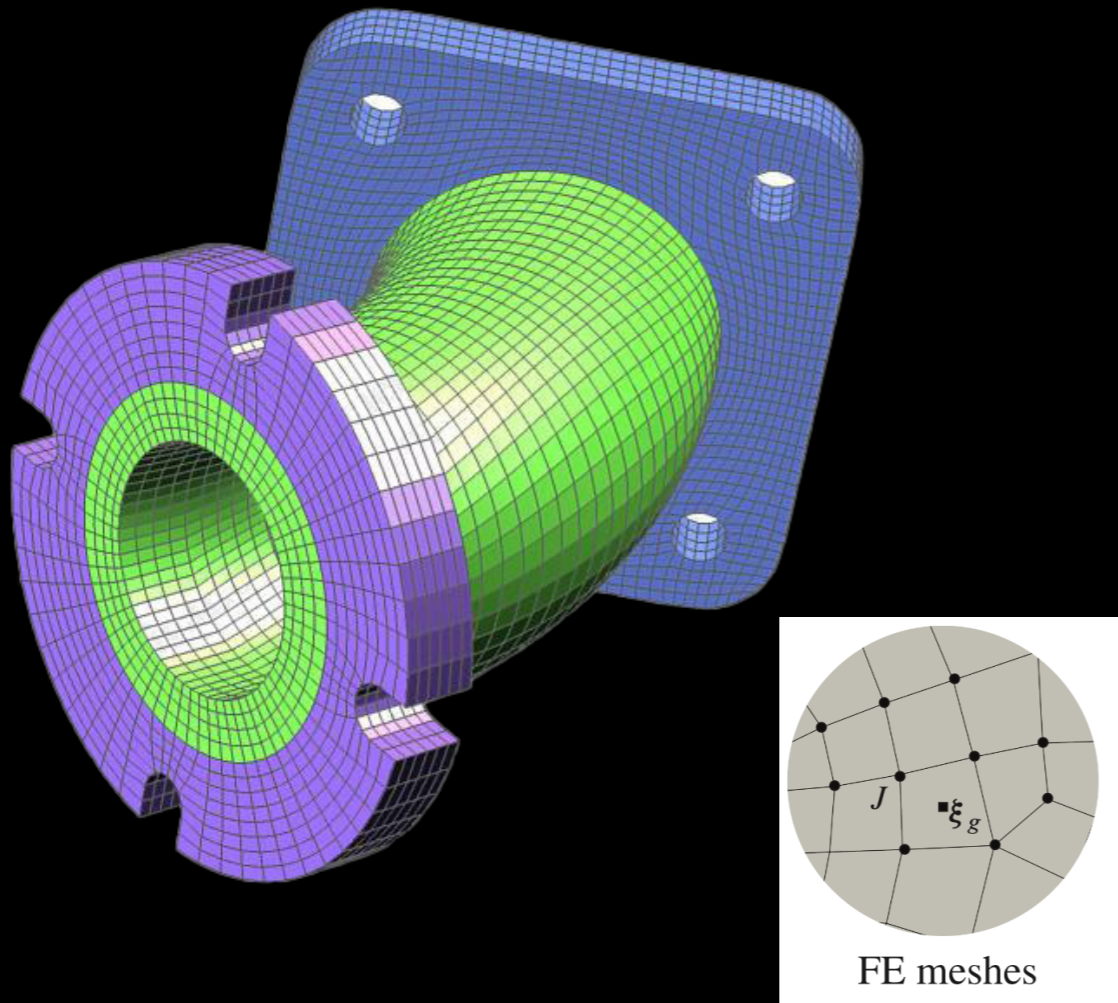


$$\begin{aligned}3x + 2y - z &= 1 \\1x - 2y + 4z &= 2 \\4x + 3y - 2z &= 5\end{aligned}$$

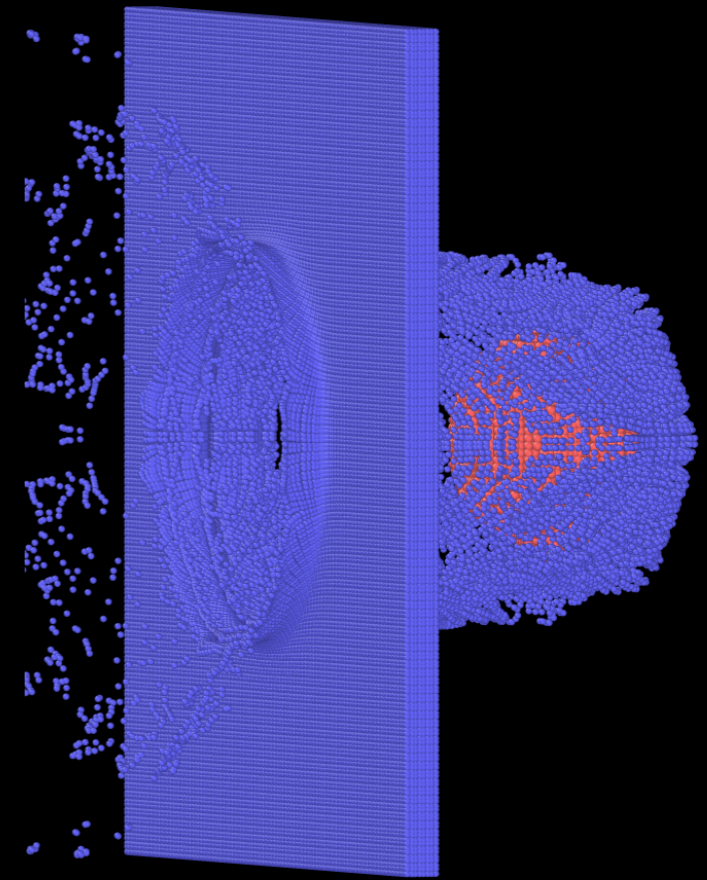


```
Welcome JS differenceWith.js x
1 import baseDifference from './.internal/baseDifference.js'
2 import baseFlatten from './.internal/baseFlatten.js'
3 import isArrayLikeObject from './isArrayLikeObject.js'
4 import last from './last.js'
5
6 /**
7  * This method is like 'difference' except that it accepts 'comparator'
8  * which is invoked to compare elements of 'array' to 'values'. The order and
9  * references of result values are determined by the first array. The comparator
10  * is invoked with two arguments: (arrVal, othVal).
11  *
12  * **Note:** Unlike 'pullAllWith', this method returns a new array.
13  *
14  * @since 4.0.0
15  * @category Array
16  * @param {Array} array The array to inspect.
17  * @param {...Array} [values] The values to exclude.
18  * @param {Function} [comparator] The comparator invoked per element.
19  * @returns {Array} Returns the new array of filtered values.
20  * @example
21  *
22  * const objects = [{ 'x': 1, 'y': 2 }, { 'x': 2, 'y': 1 }]
23  *
24  * differenceWith(objects, [{ 'x': 1, 'y': 2 }], isEqual)
25  * // => [{ 'x': 2, 'y': 1 }]
26  */
27 function differenceWith(array, ...values) {
28   let comparator = last(values)
29   let isObject = typeof comparator === 'object'
30   let isFunction = typeof comparator === 'function'
31   let isObjectLike = isObjectLikeObject(comparator)
32   let isObjectLikeAndFunction = isObjectLike && isFunction
33   let comparator = isObjectLikeAndFunction ? comparator : isFunction ? comparator : undefined
34   let result = []
35   let length = array.length
36   let valuesLength = values.length
37   let value = array[length - 1]
38   let valueIndex = length - 1
39   let seen = isObjectLikeAndFunction ? new Set(values) : undefined
40   while (valueIndex >= 0) {
41     value = array[valueIndex]
42     valueIndex--
43     if (valueIndex < 0 && !seen) {
44       continue
45     }
46     if (isFunction ? comparator(value, values[0]) : !seen.has(value)) {
47       result.push(value)
48     }
49     if (seen) {
50       seen.add(value)
51     }
52   }
53   return result
54 }
```

# Mesh based and mesh-free methods

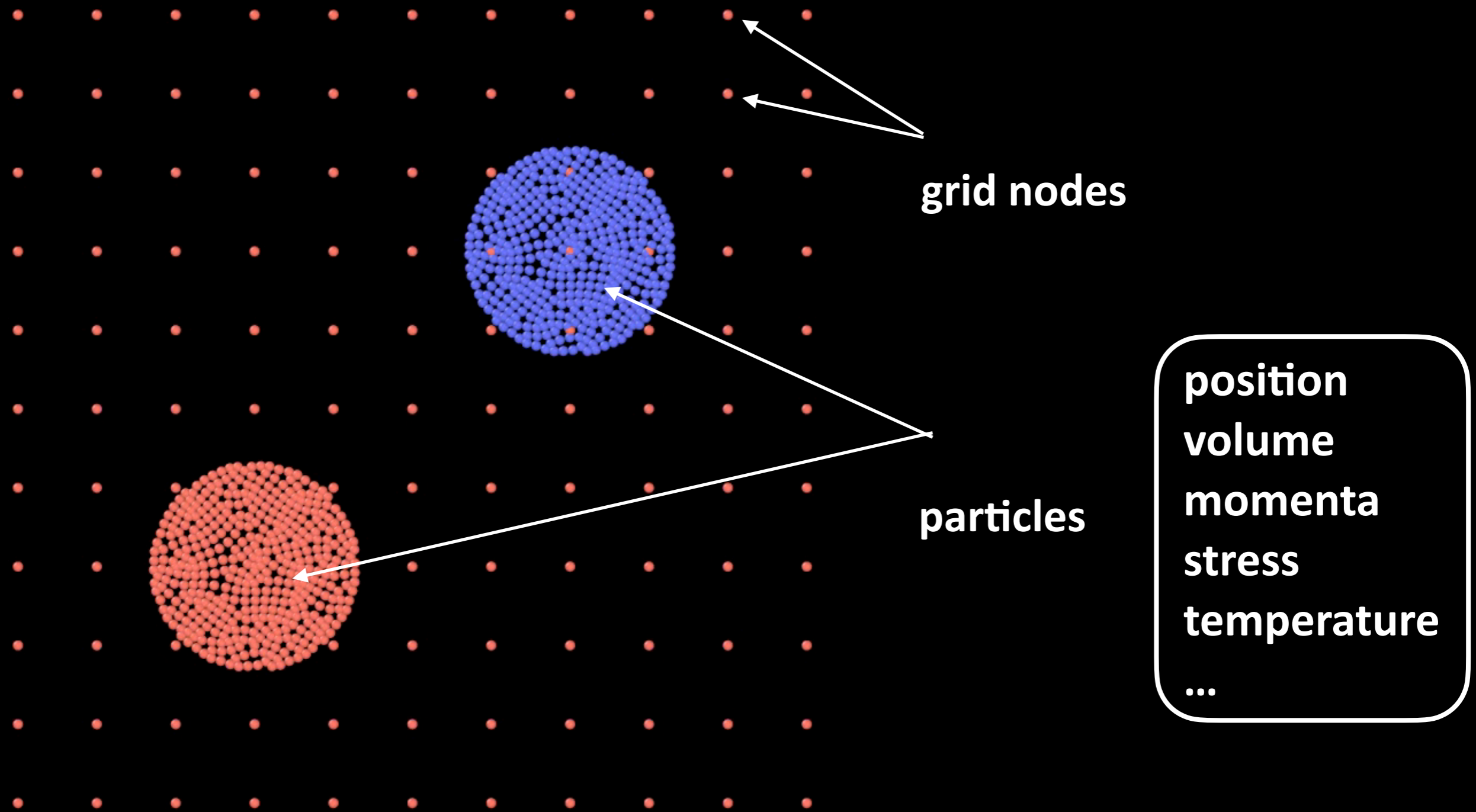


Basic concept: a mesh/grid  
FDM, FVM, FEM  
Efficient, accurate  
FEM: **prone to mesh distortion**



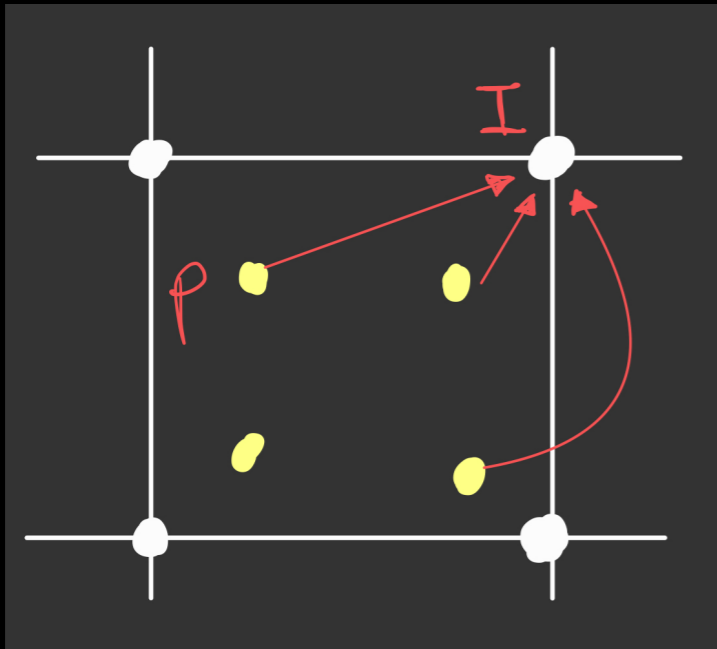
Basic concept: points/particles  
EFG, SPH, PFEM, MPM, ...  
Mesh distortion free  
**Inaccurate geometry**

# Material Point Method (MPM): particles over a fixed grid

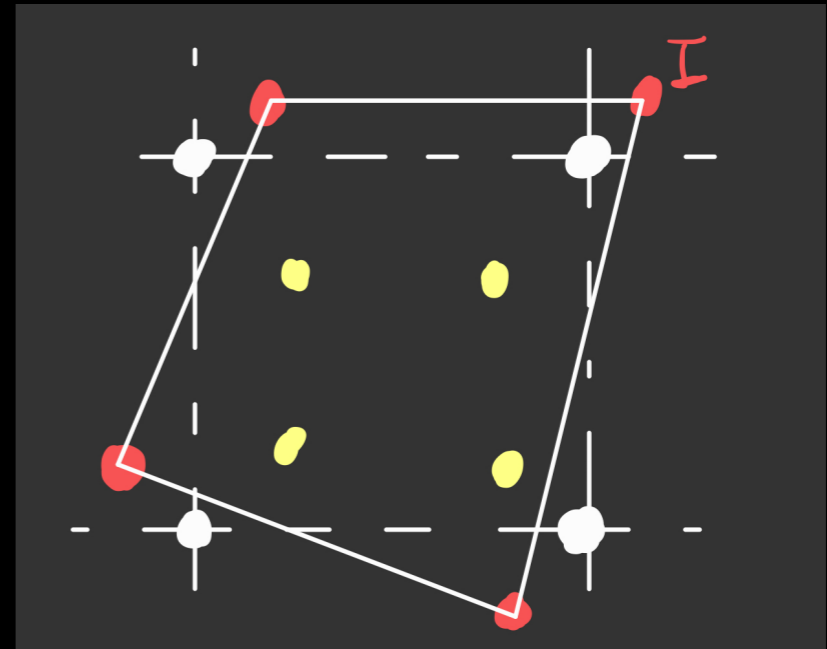


# Material Point Method: algorithm (explicit, ULMPM)

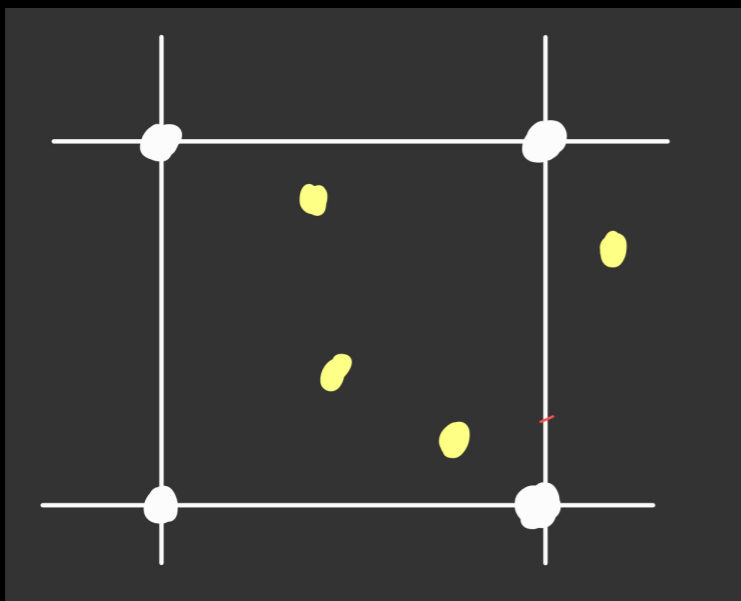
① P2G



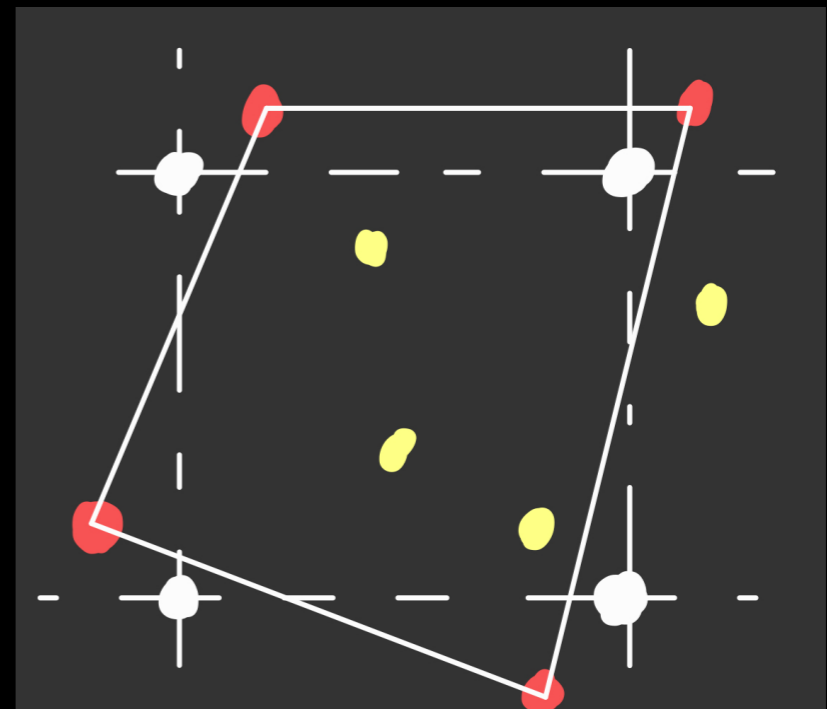
② Grid update



④ Grid reset

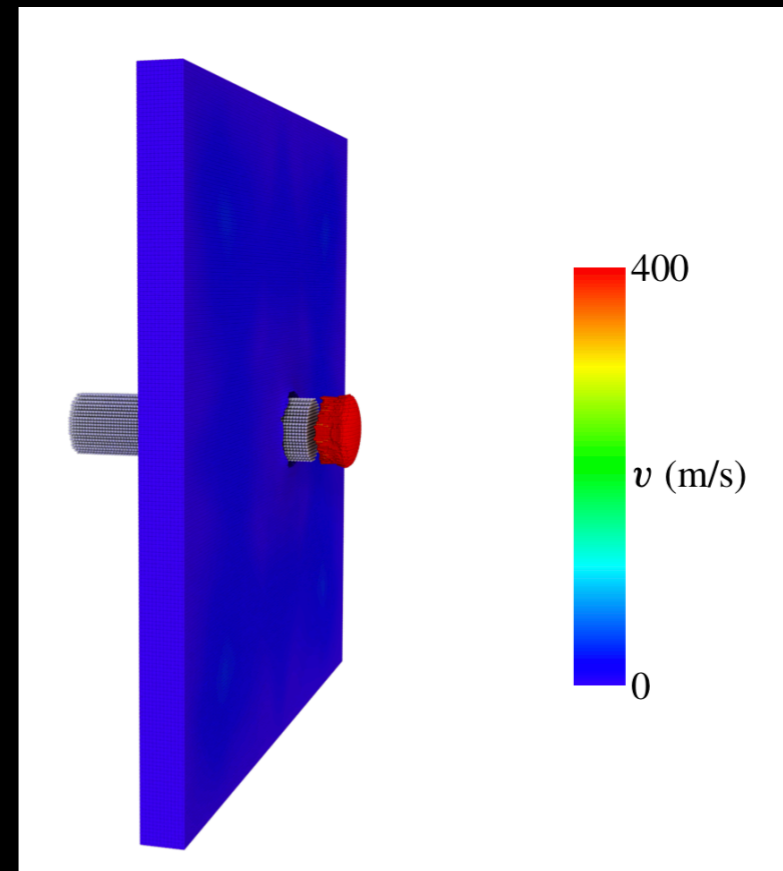
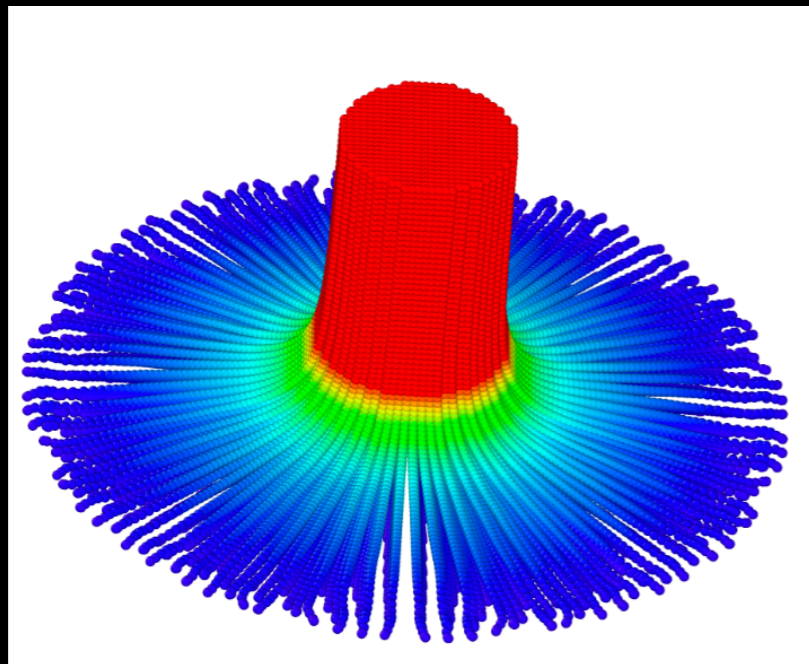
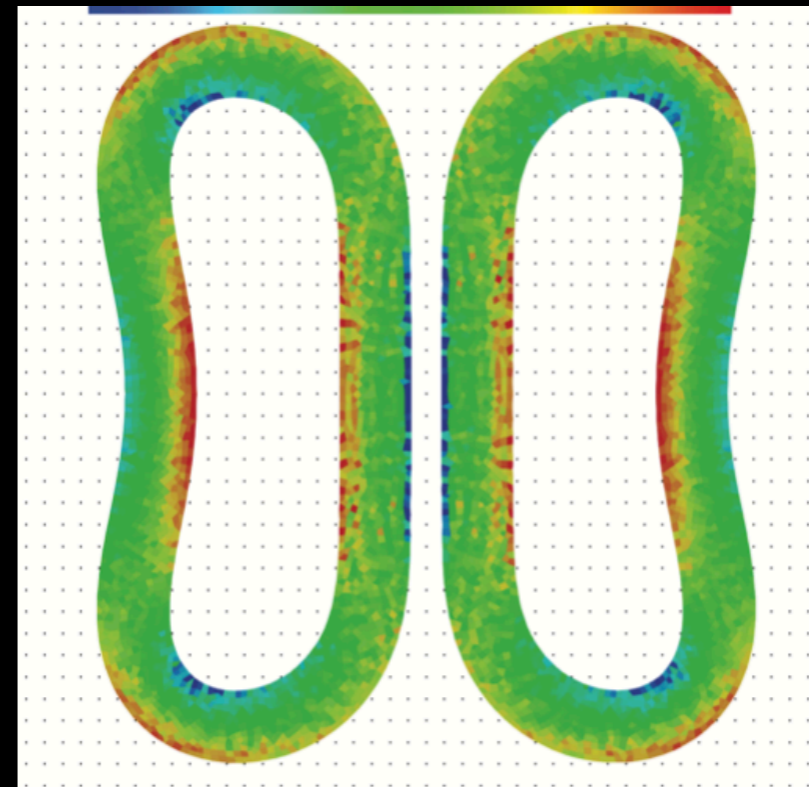
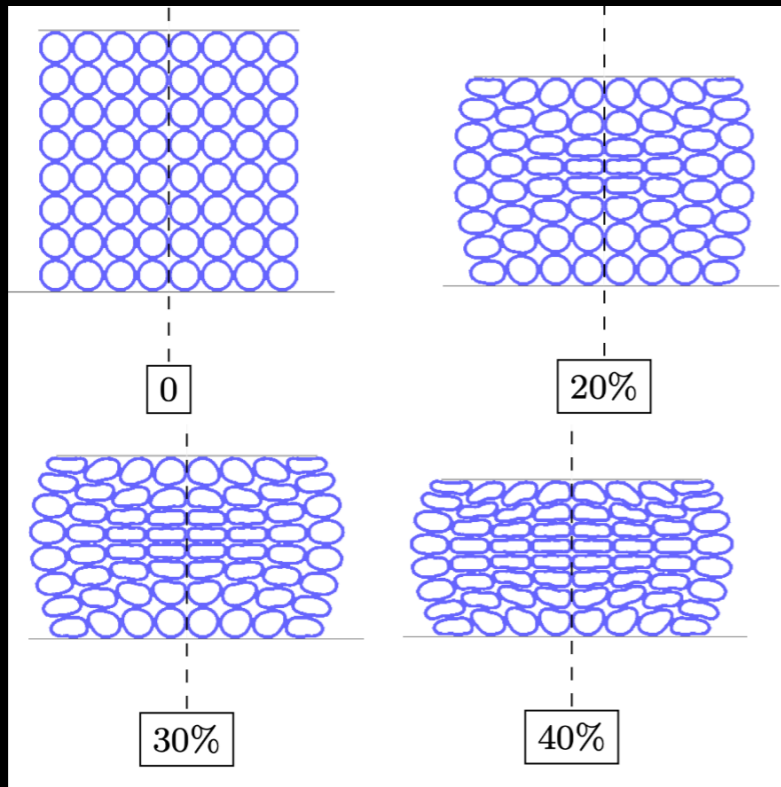


③ G2P





**MPM really can simulate complex problems**



# MPM: Example 1

---

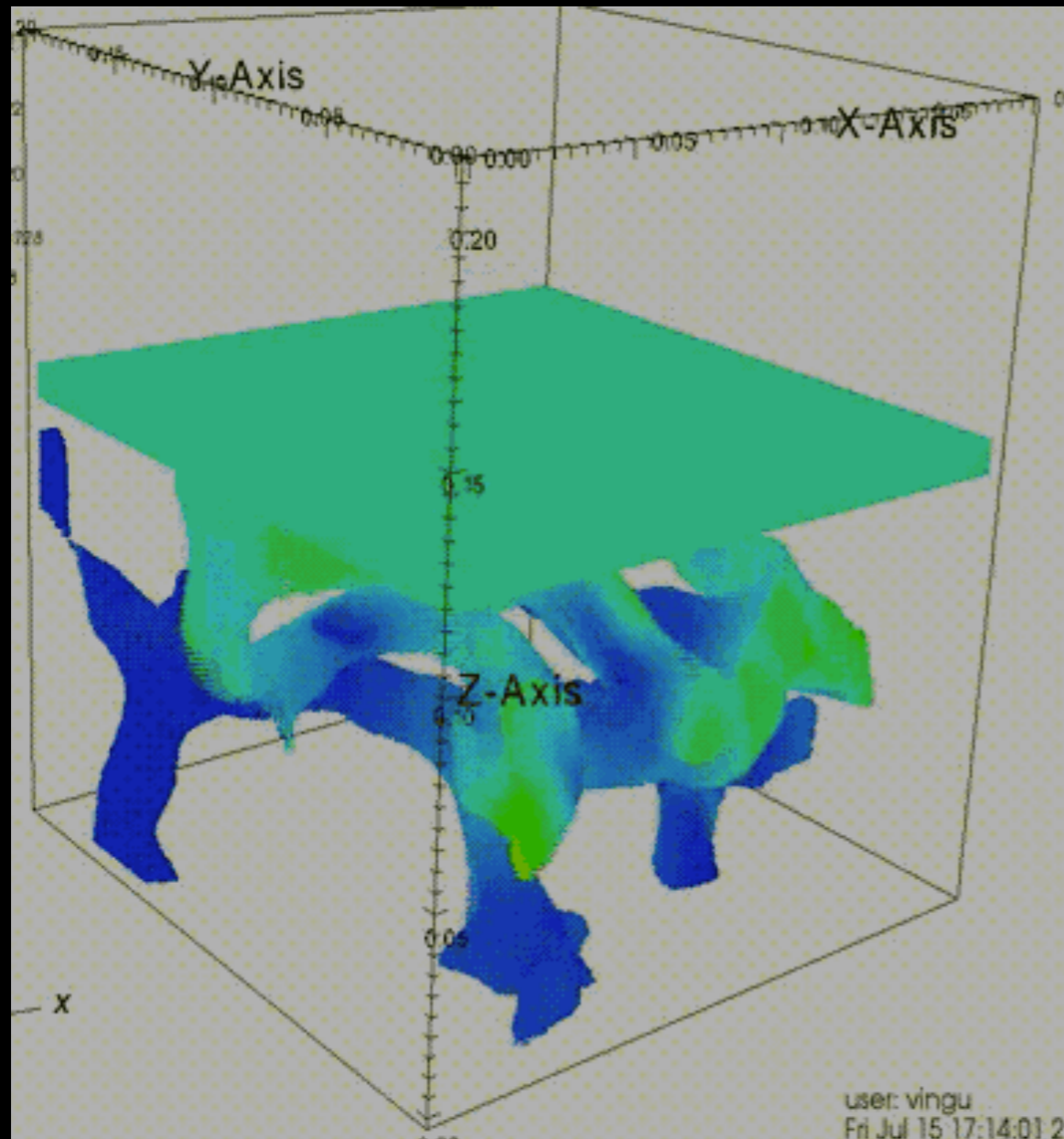
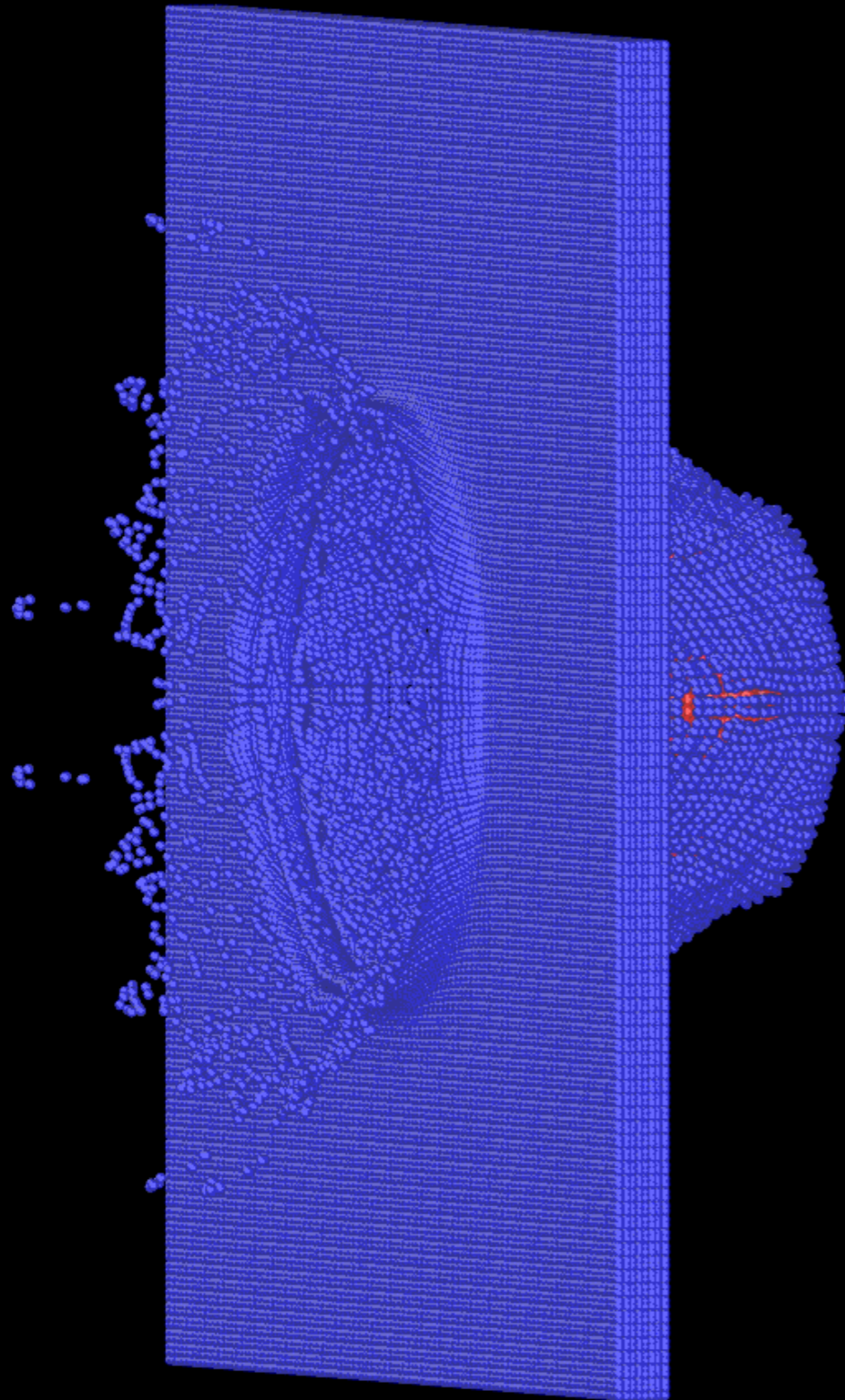


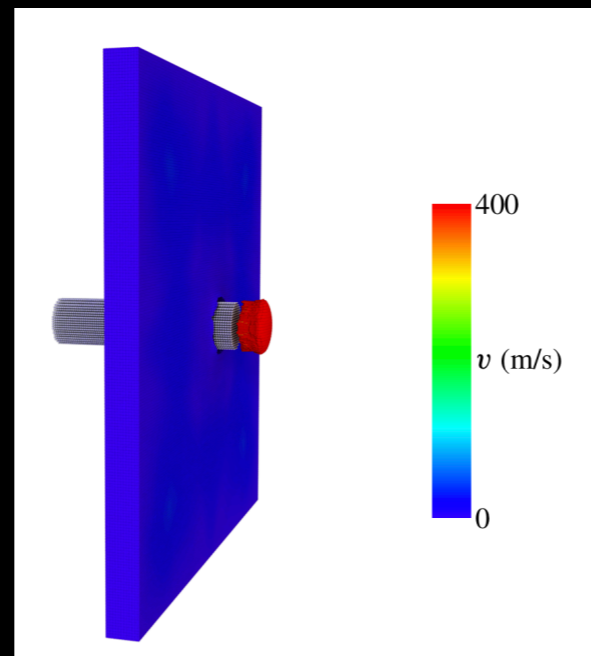
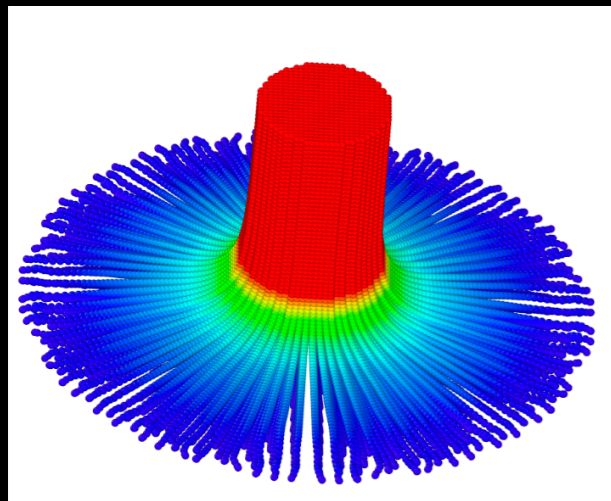
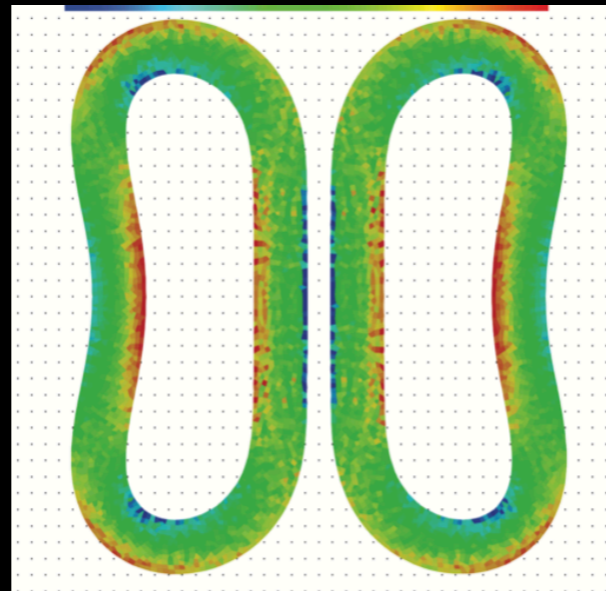
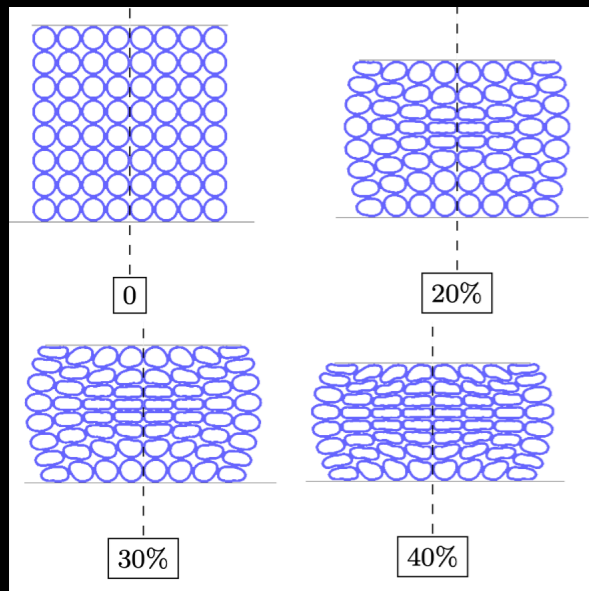
Image based analysis  
Foamed materials: network of struts  
Very large deformation  
Lots of **self contacts**

# MPM: example 2

---



Hyper-velocity impact  
Steel plate: Johnson-Cook model  
Massive deformation  
Karamelo (in-house C++ MPI)



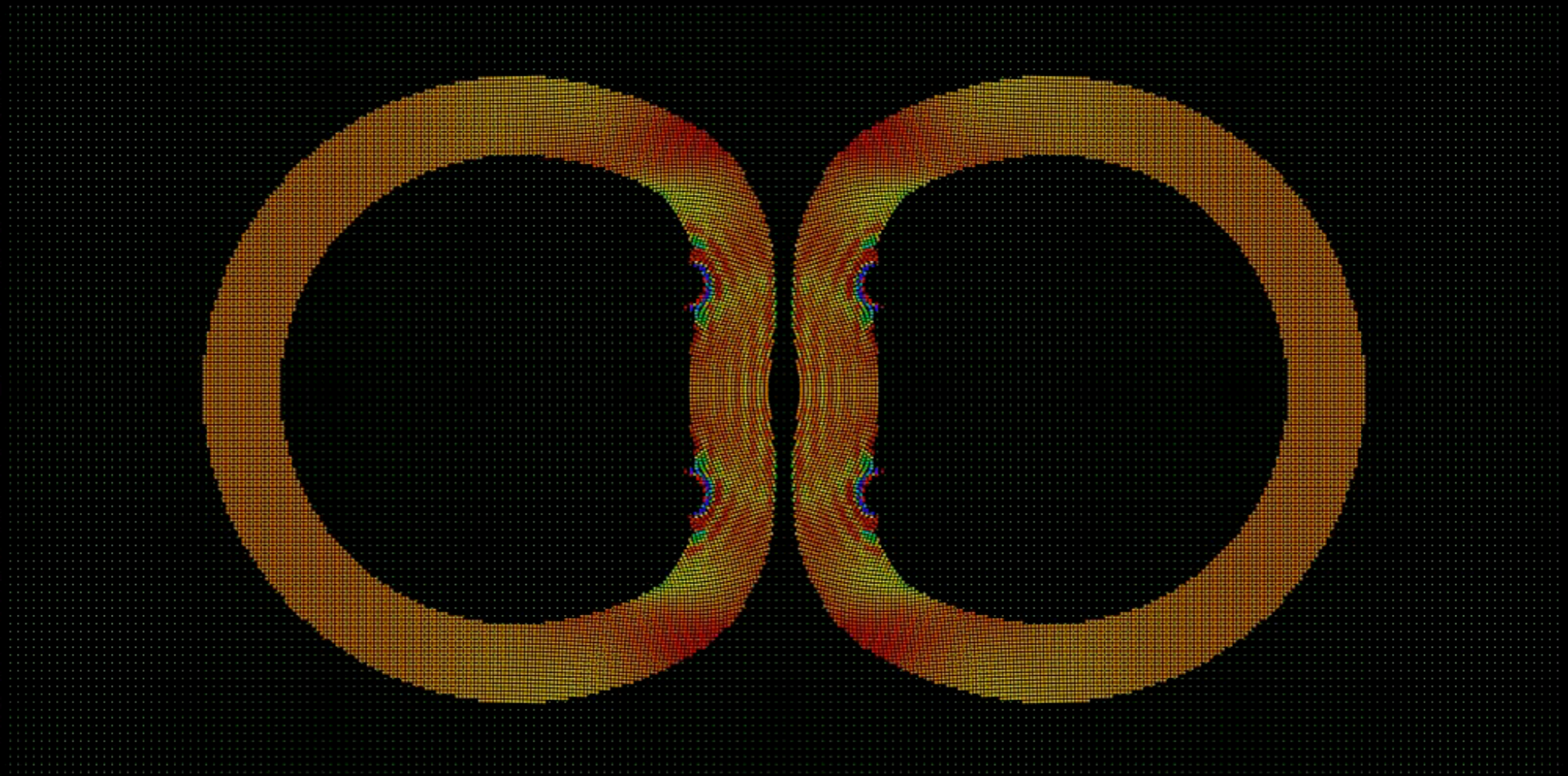
**no mesh generation**  
**simple basis functions**  
**large deformation**  
**automatic no-slip contact**  
**multiple contacts**  
**simple BC treatment**  
**simple implementation**

**MPM really can simulate complex problems**

**BUT**

# (UL)MPM: issues

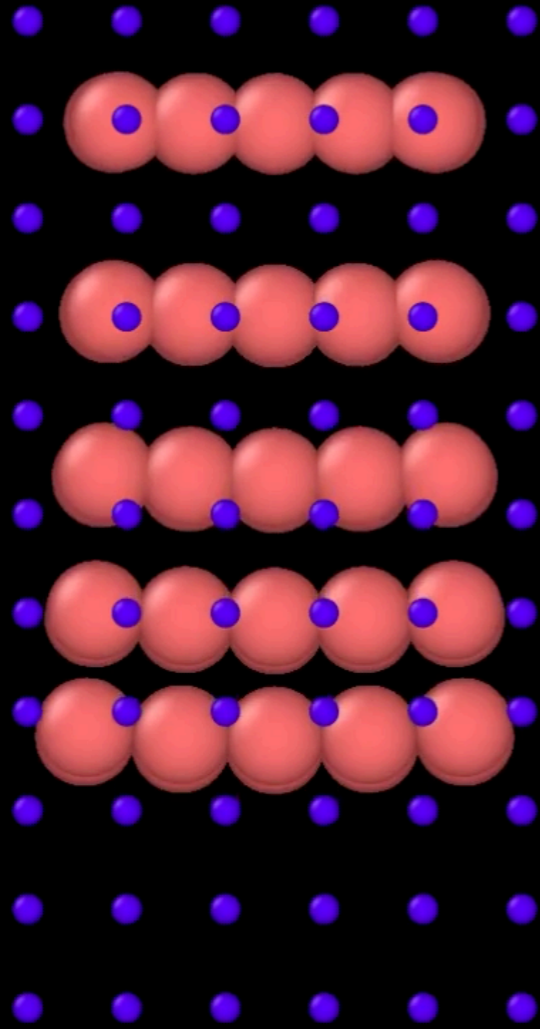
---



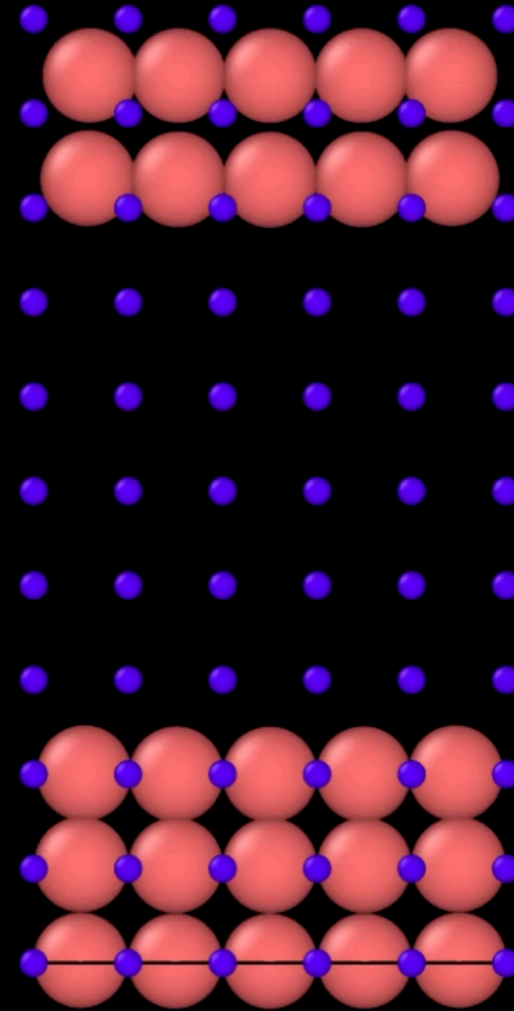
**cell-crossing instability**

# (UL)MPM: issues

---



**Total Lagrangian**

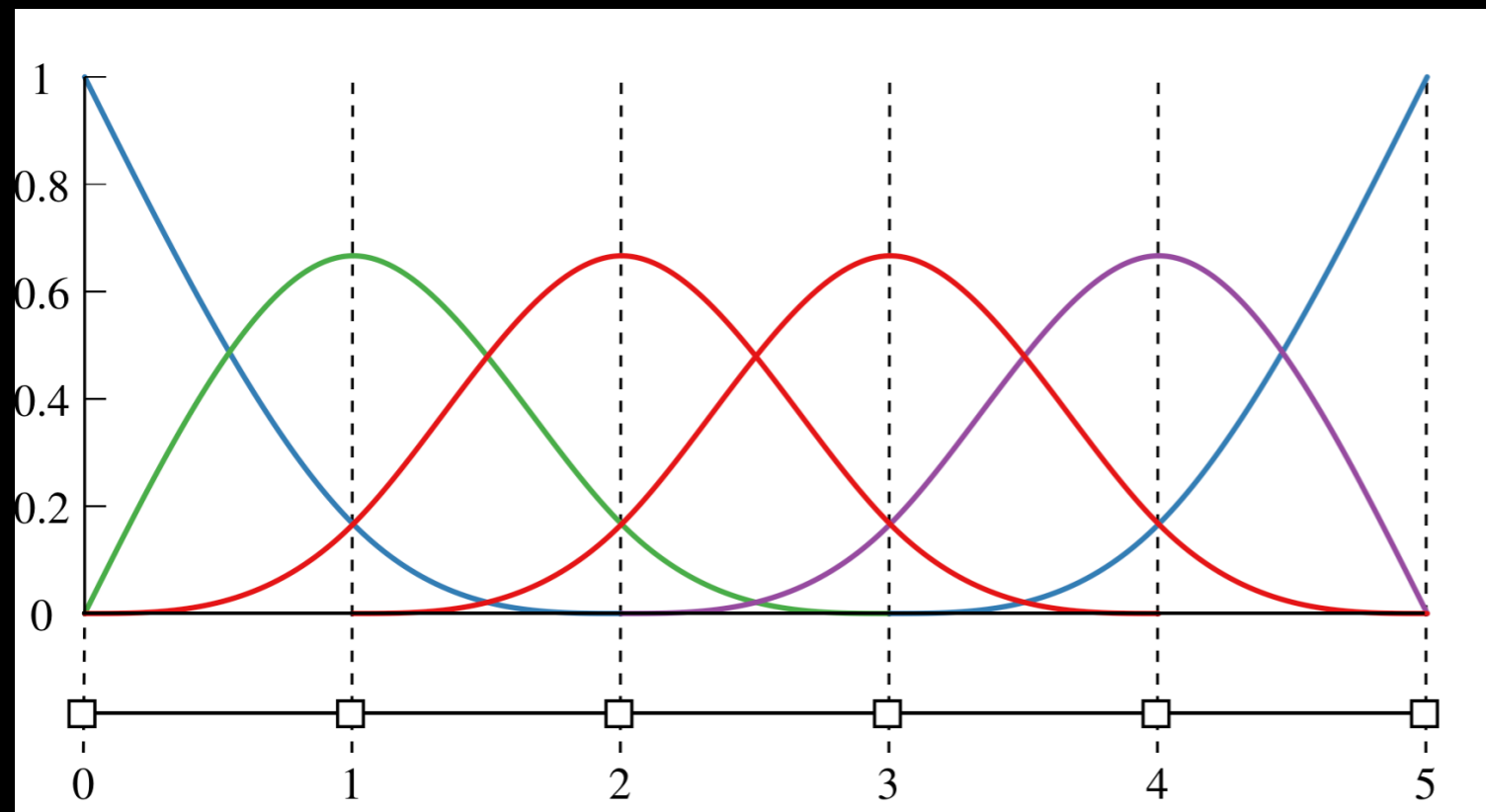
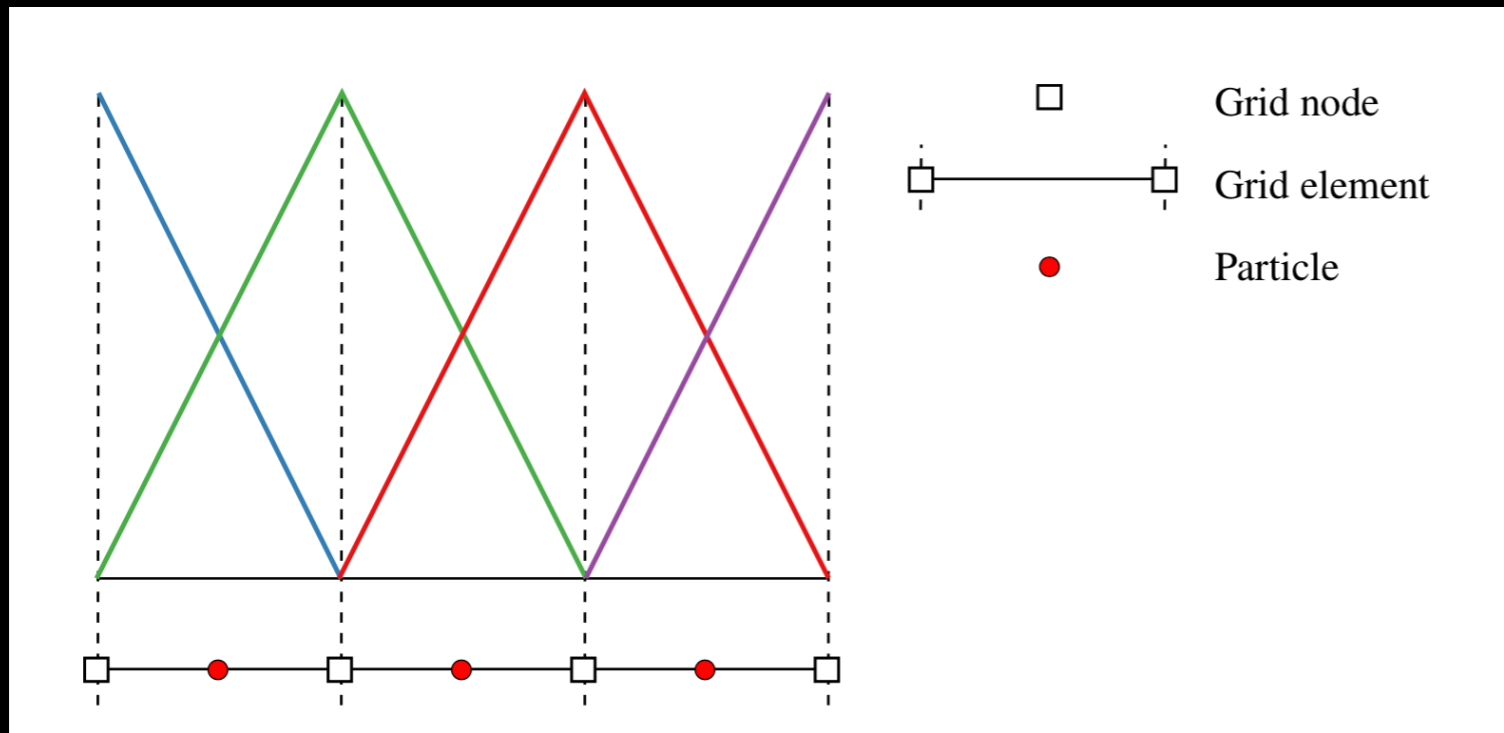


**Standard**

numerical fracture

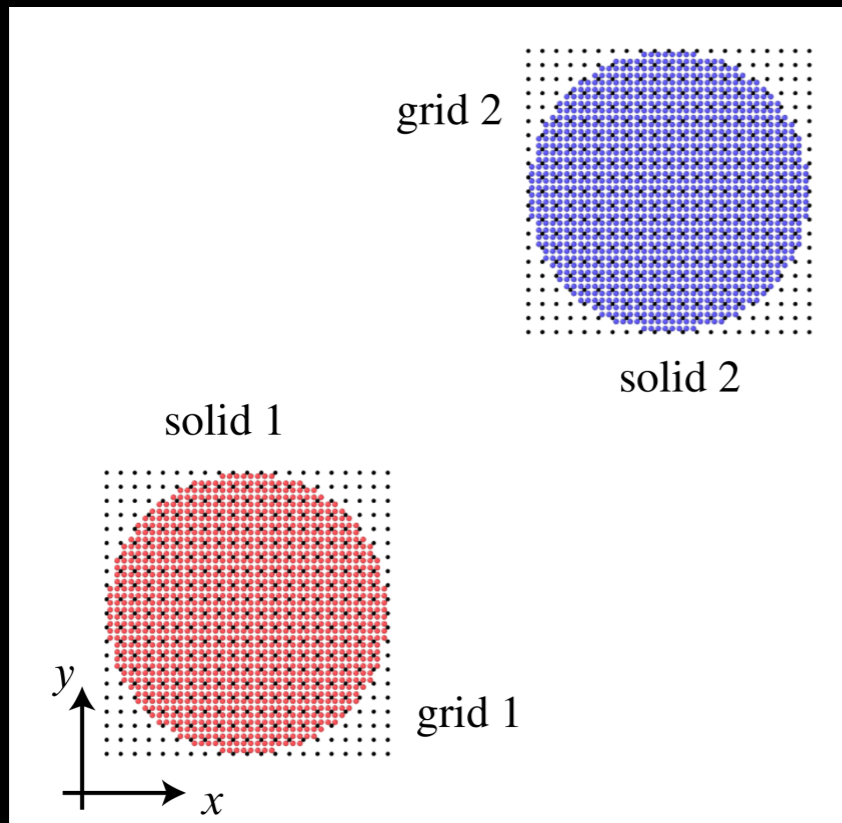


# ULMPM: issues and *partial* solutions

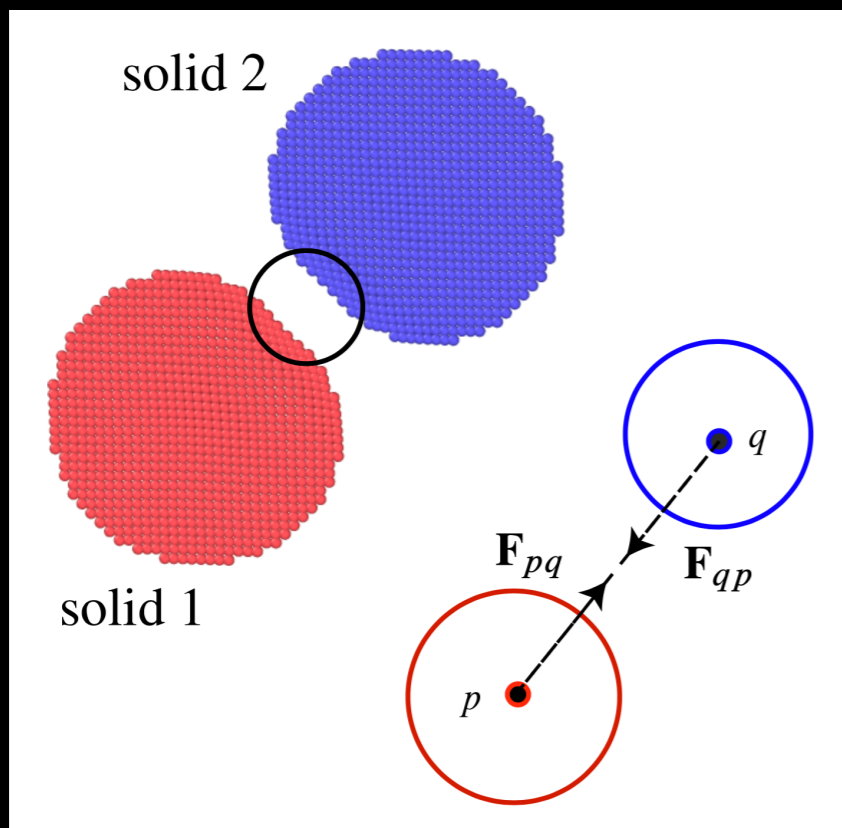


**We present two solutions to solve issues of ULMPM**

# Total Lagrangian MPM (TLMPM): initial configuration



use the initial undeformed configuration  
 no cell-crossing instability  
 no numerical fracture  
 most accurate quadrature  
**no inherent no-slip contact**  
**small time step for compression**



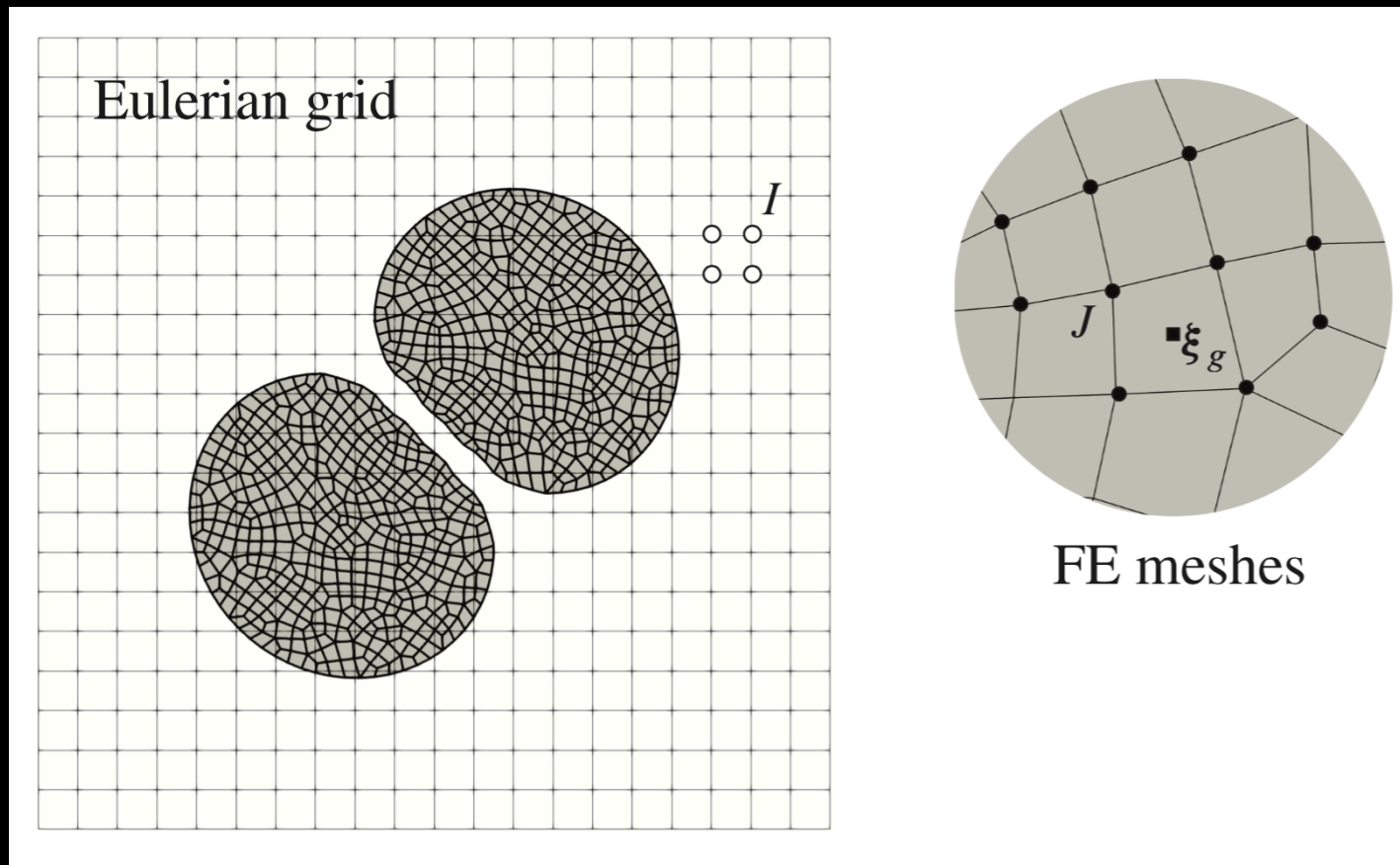
$$\delta^t = R_p + R_q - \|\mathbf{x}_q^t - \mathbf{x}_p^t\|$$

$$\mathbf{F}_{pq} = \frac{1}{\Delta t^2} \frac{m_p m_q}{m_p + m_q} \left( 1 - \frac{R_p + R_q}{\|\mathbf{x}_{pq}^t\|} \right) \mathbf{x}_{pq}^t$$

A. de Vaucorbeil, V. P. Nguyen, and C. R. Hutchinson. A total-lagrangian material point method for solid mechanics problems involving large deformations. CMAME, 2020.

A. de Vaucorbeil and V.P. Nguyen. Modeling contacts with a total lagrangian material point method. CMAME 2021.

# Generalised Particle in Cell (GPIC): mesh again



solids are **meshed**  
initial configuration: forces  
current configuration: contact

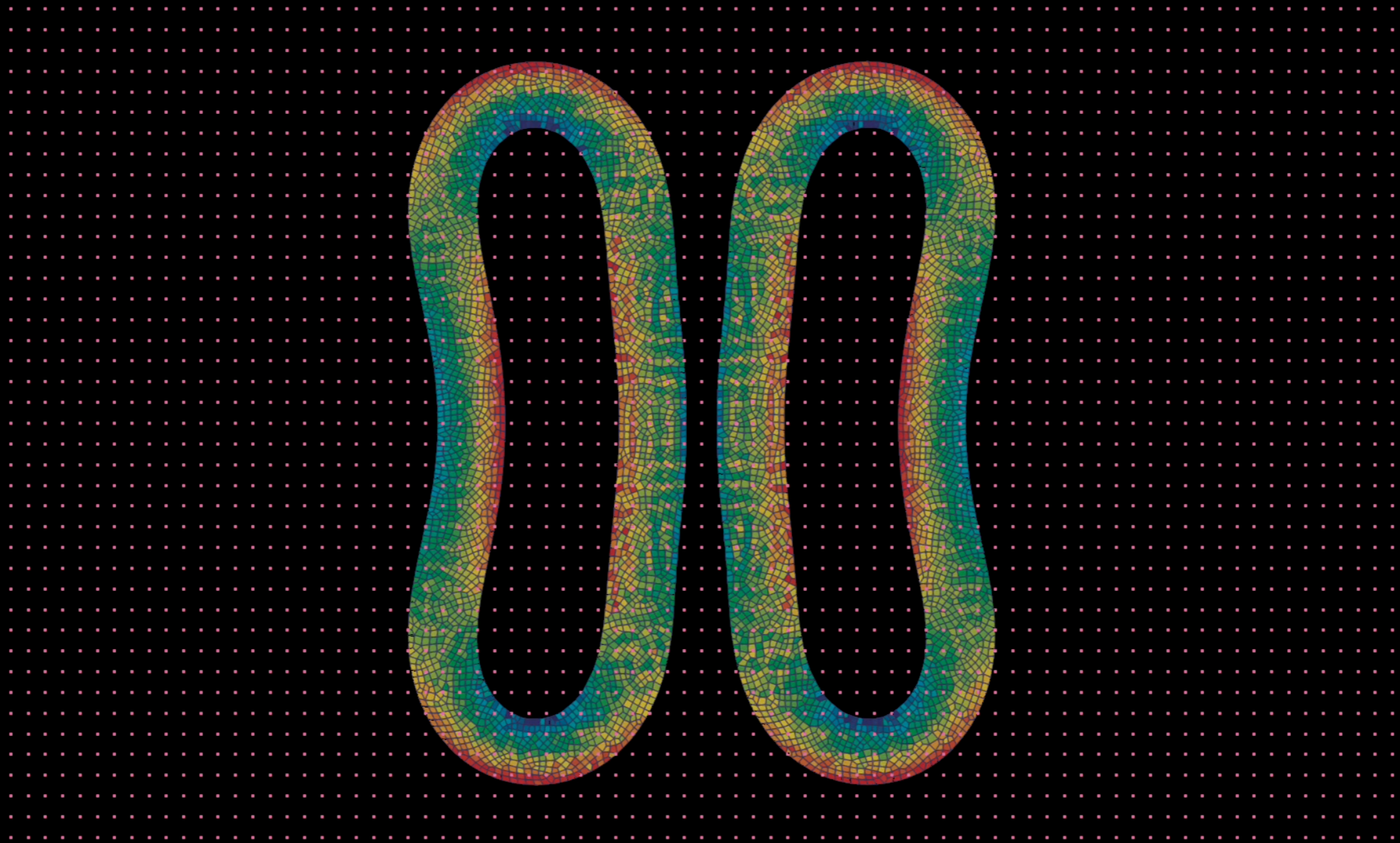
no cell crossing issue  
no numerical fracture  
accurate quadrature  
**accurate boundary**  
**easy boundary conditions**  
**easy material interfaces**

**fragmentation?**  
**extreme deformation?**

1. A. Sadeghirad, R. M. Brannon, and J. Burghardt. A convected particle domain interpolation technique to extend applicability of the material point method for problems involving massive deformations. *International Journal for Numerical Methods in Engineering*, 86(12):1435–1456, 2011.
2. V. P. Nguyen, C. T. Nguyen, T. Rabczuk, and S. Natarajan. On a family of convected particle domain interpolations in the material point method. *Finite Elements in Analysis and Design*, 126:50–64, 2017.
3. V.P. Nguyen, A. de Vaucorbeil, C. Nguyen-Thanh, and T. Mandal. A generalized particle in cell method for explicit solid dynamics. *Computer Methods in Applied Mechanics and Engineering*, 360:112783, 2020.

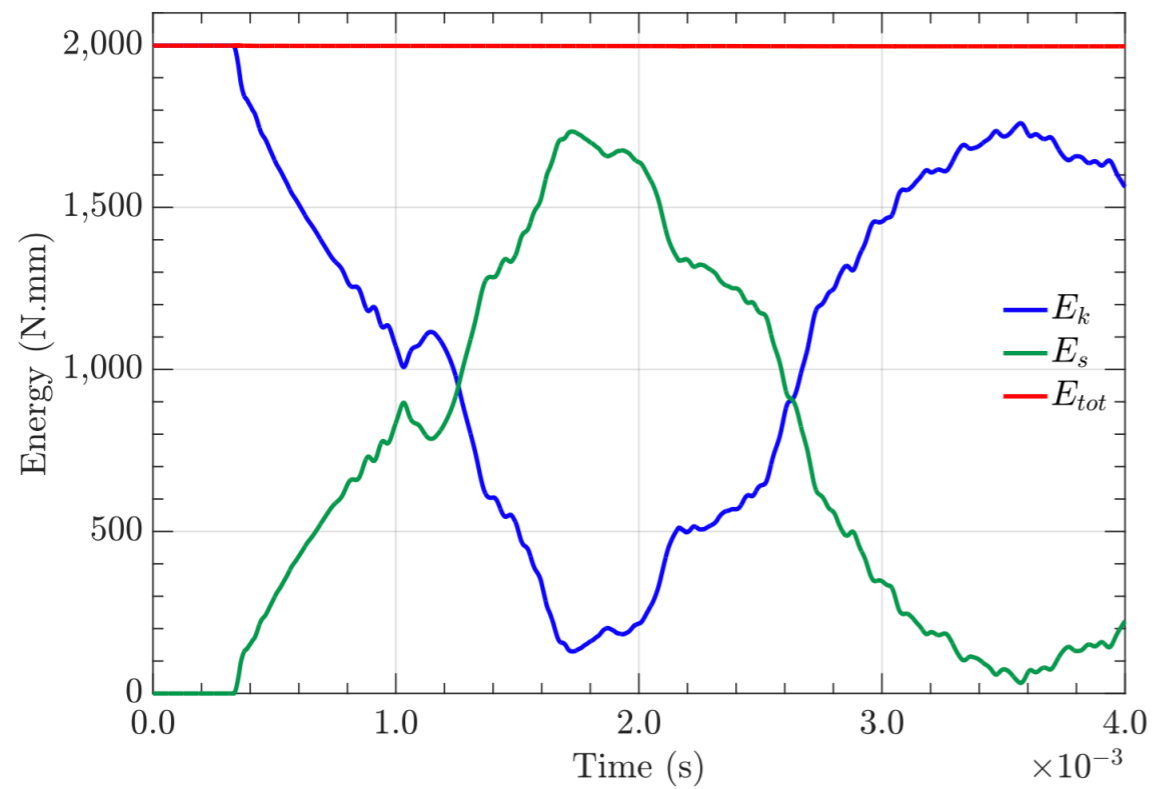
# Generalised Particle in Cell (GPIC): example 1

---

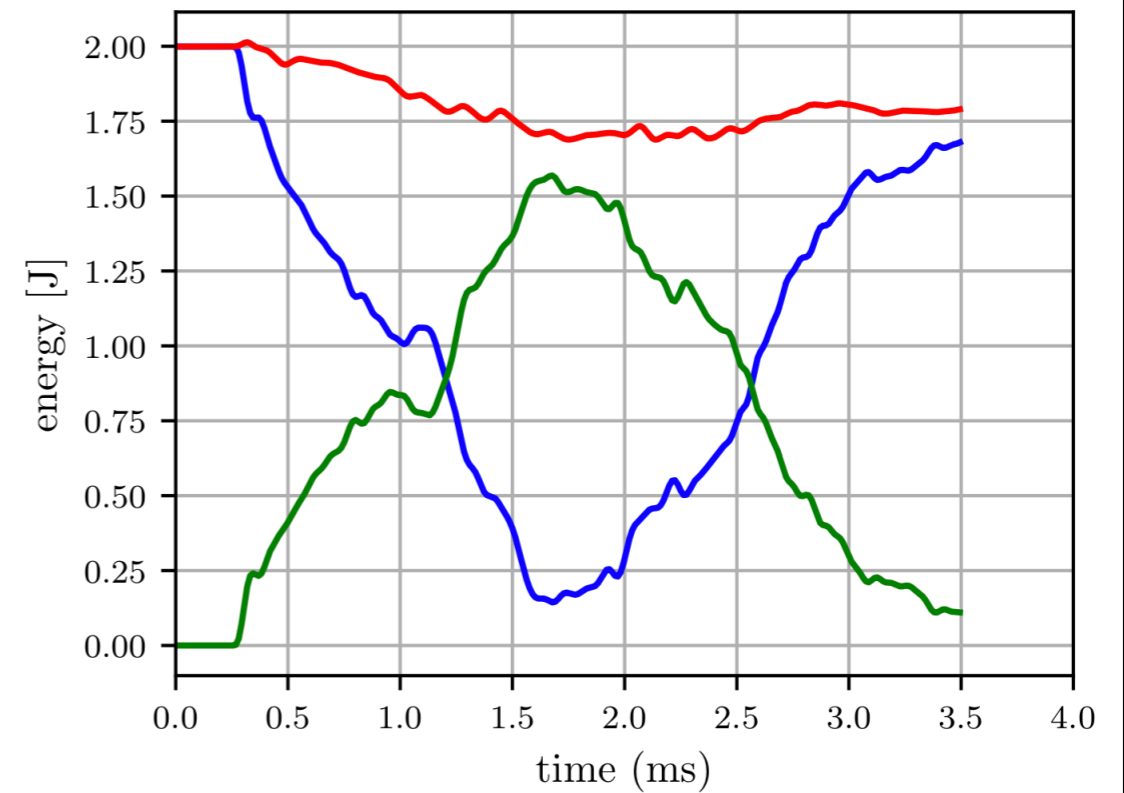


# Generalised Particle in Cell (GPIC): example 1

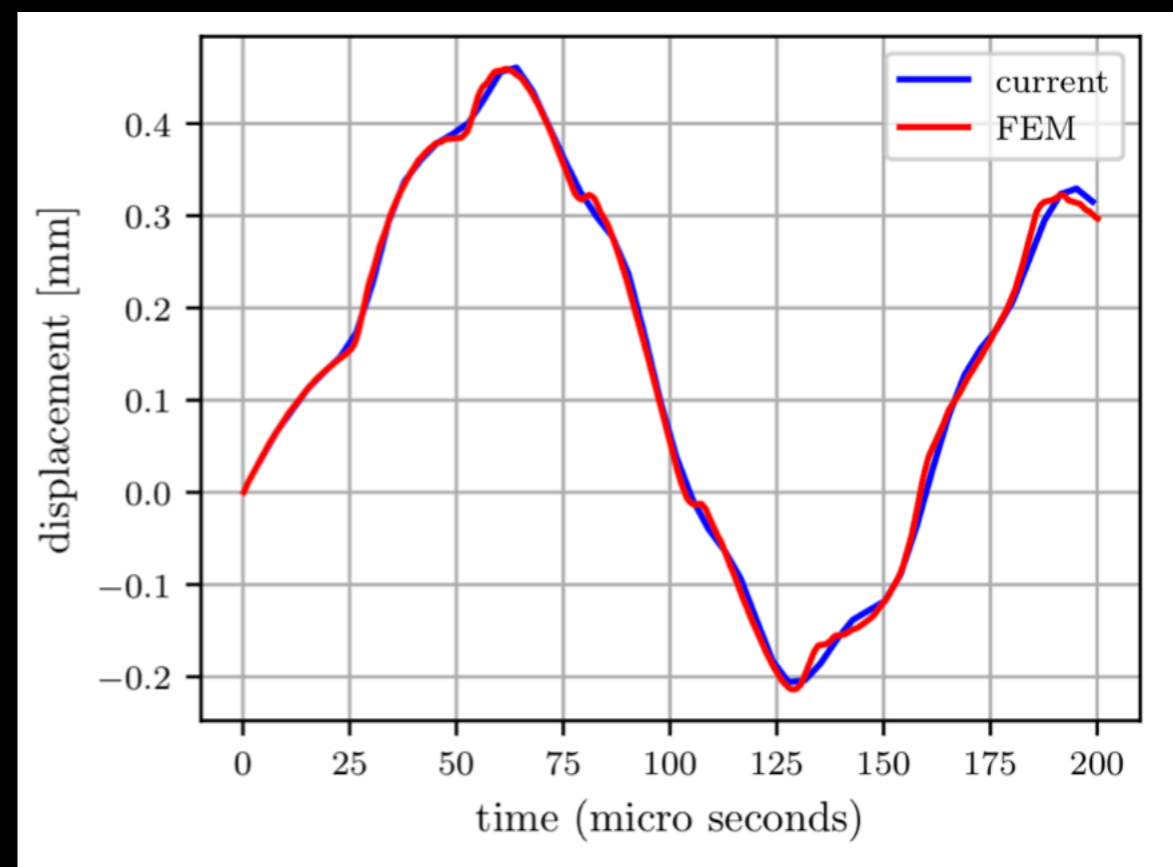
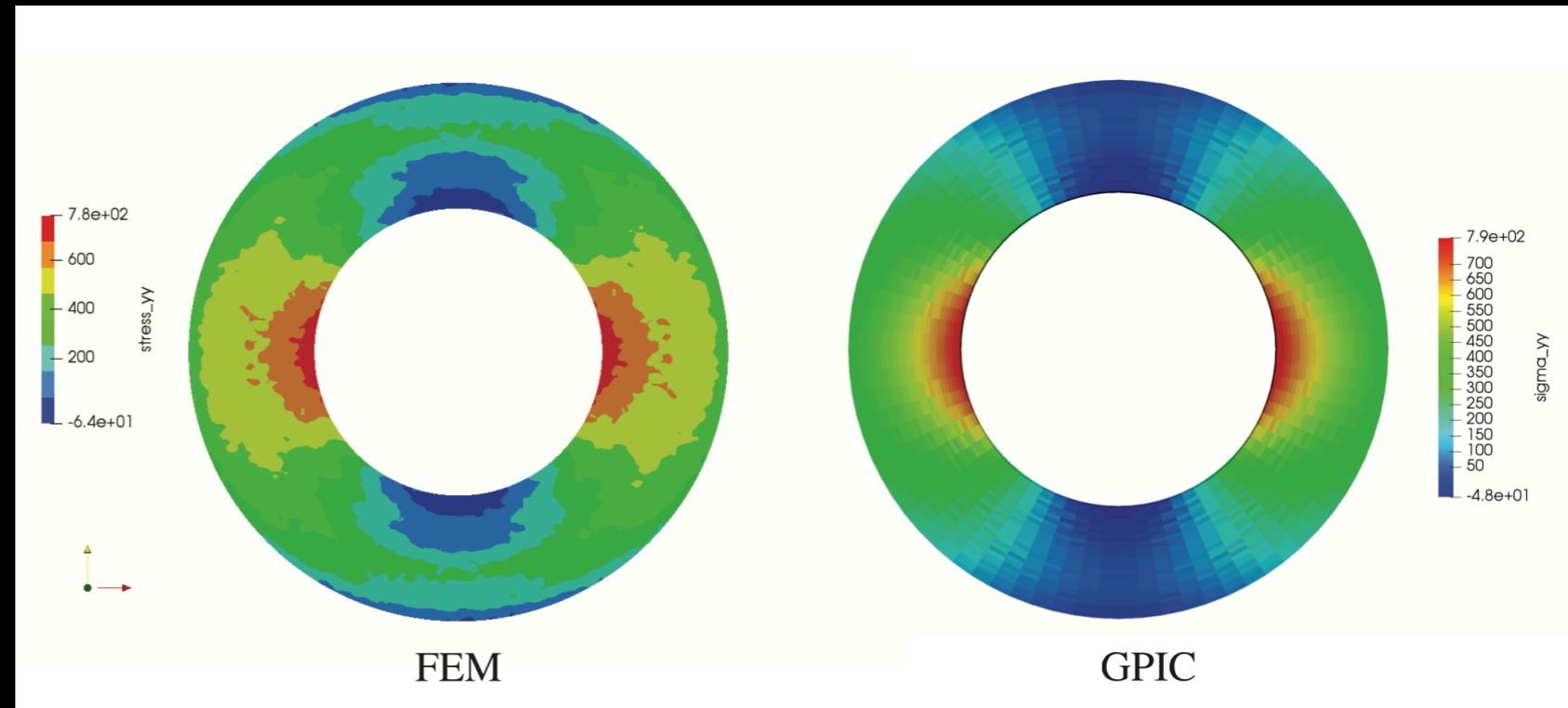
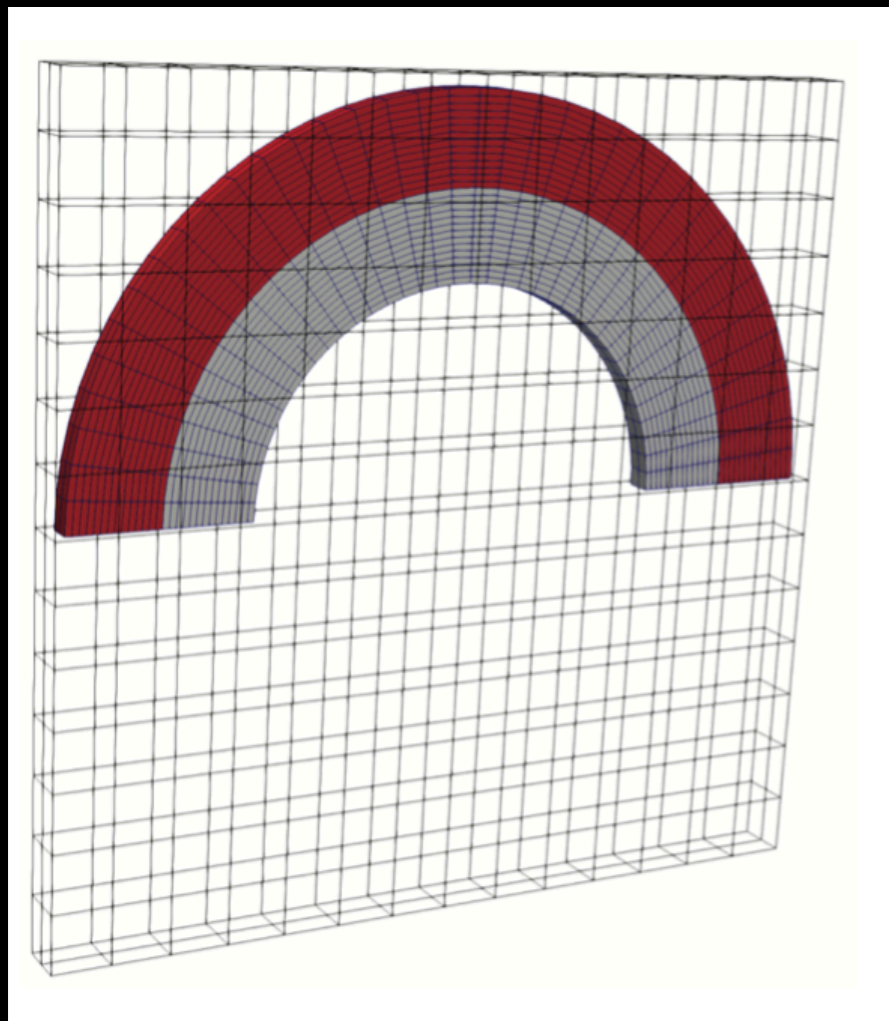
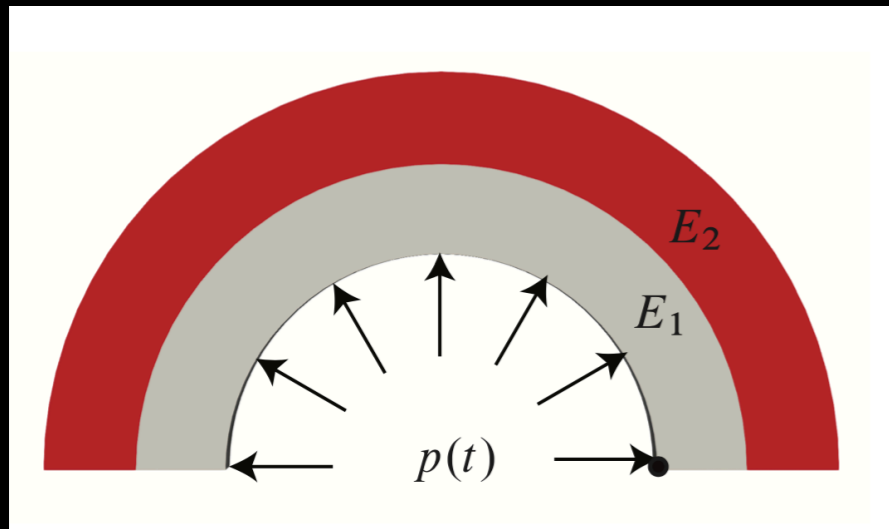
## ABAQUS



## GPIC

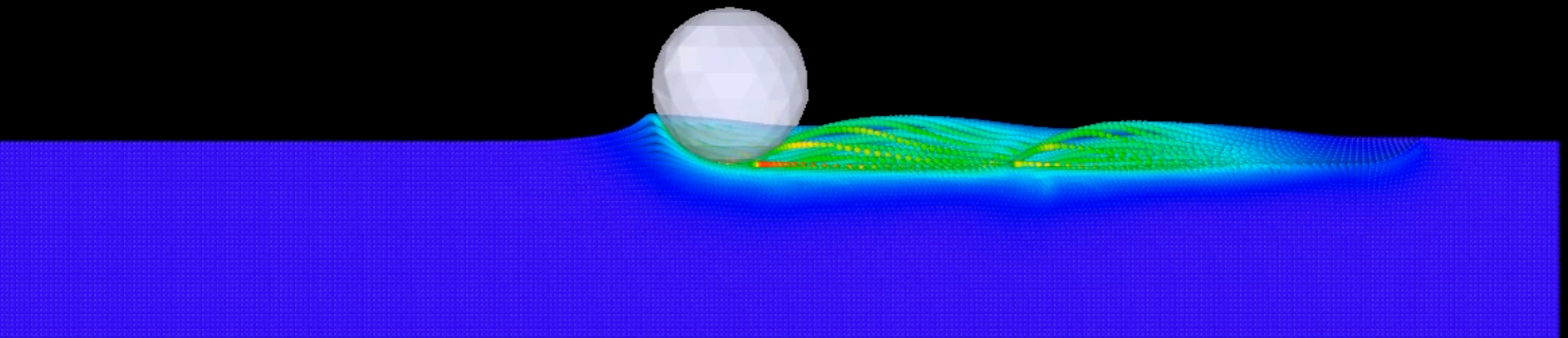
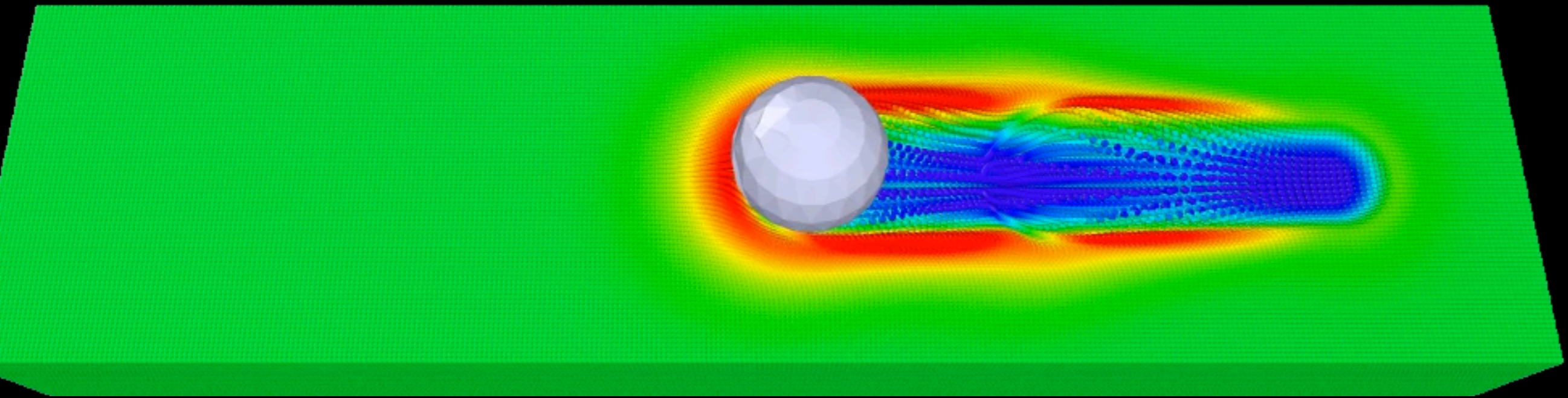


# Generalised Particle in Cell (GPIC): example 2



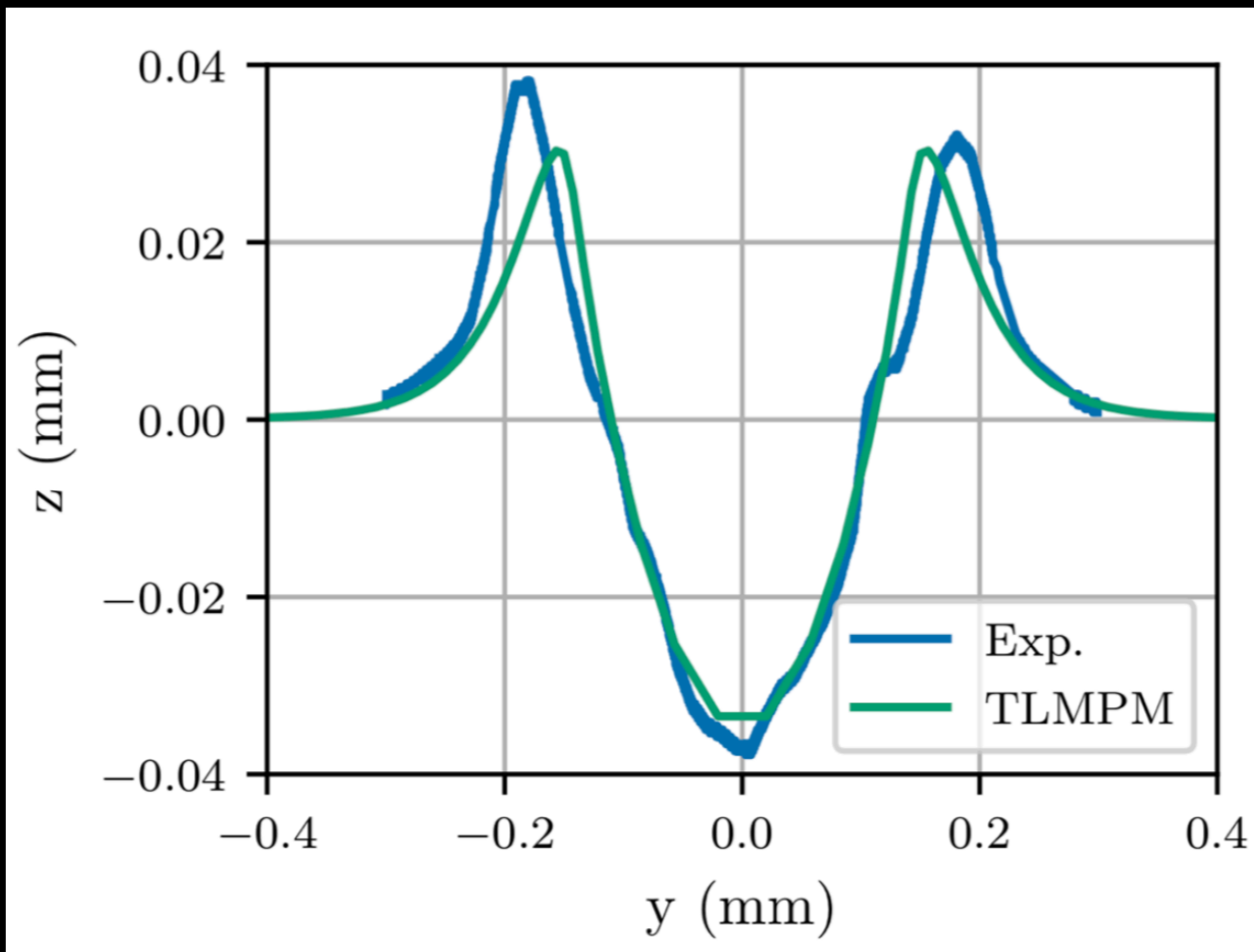
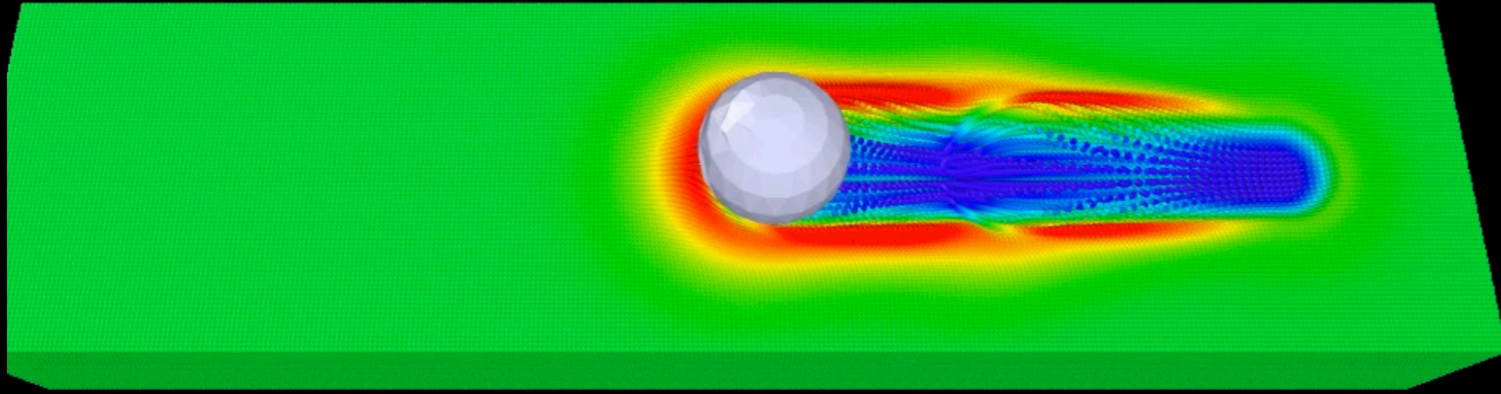
# Scratch test of copper: TLMPM, Johnson-Cook plastic model

---





# Scratch test of copper: groove topology



# Modelling fracture

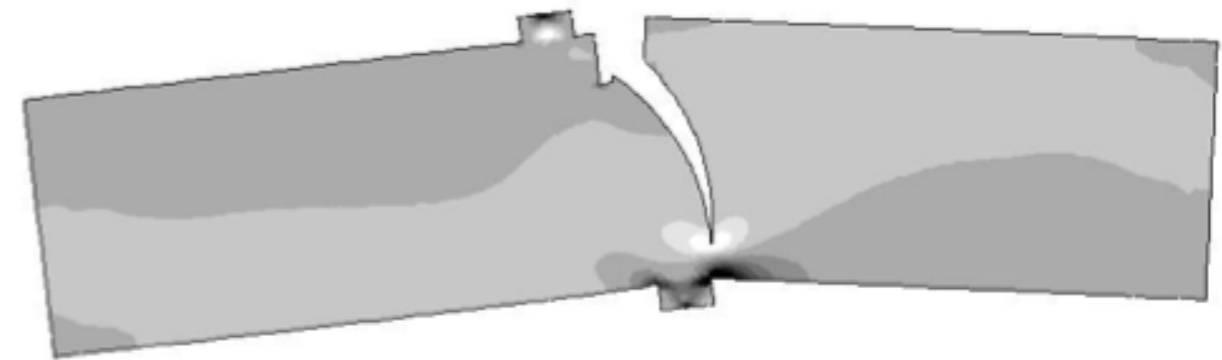
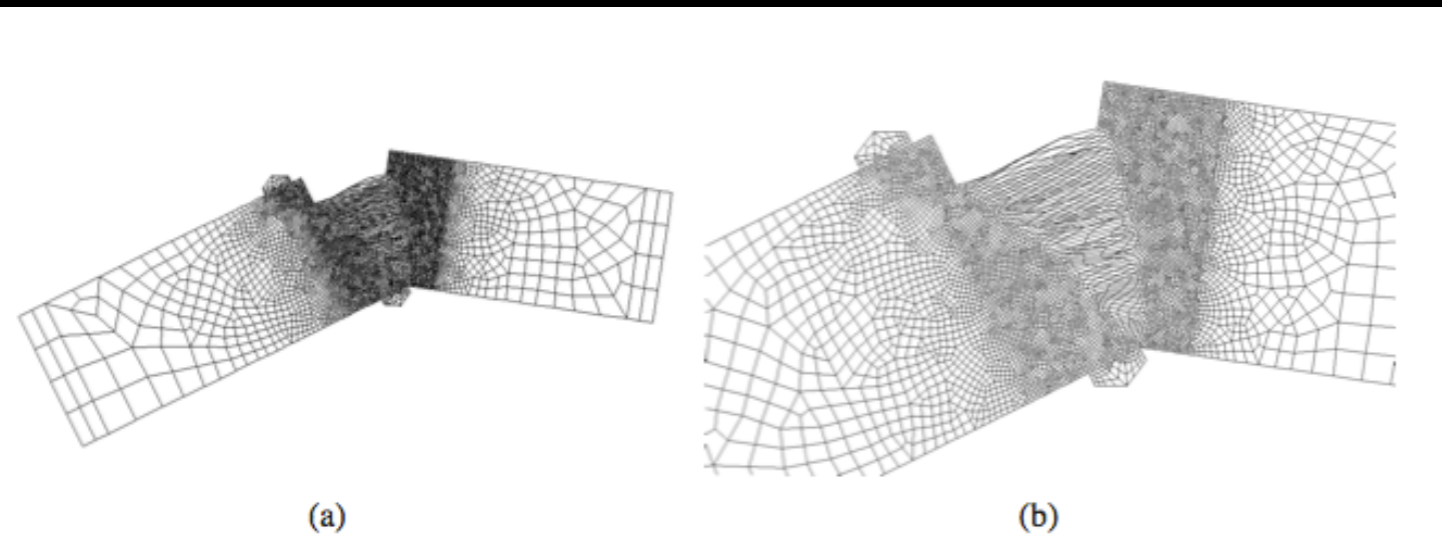
---

## Damage mechanics approach:

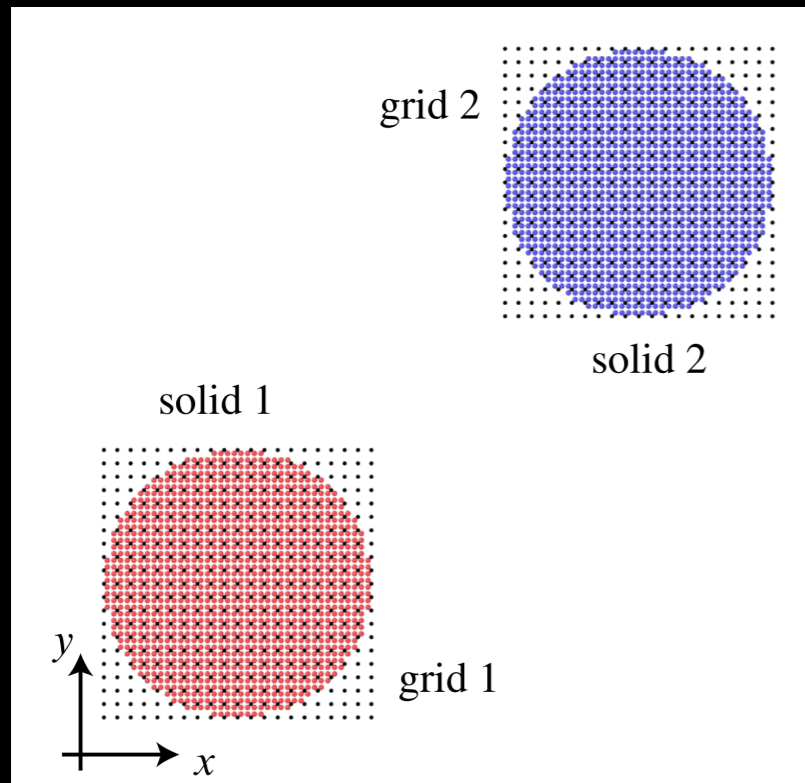
- + easy to code (2D, 3D)
- less efficient (refined meshes)
- hard to capture discontinuities
- required non locality

## Fracture mechanics approach:

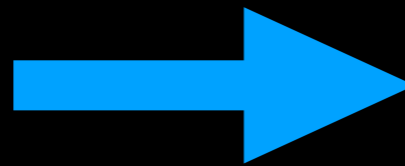
- + capture discontinuities
- + efficient
- hard to implement
- tracking 3D non-planar crack surfaces



# TLMPM: modeling ductile fracture



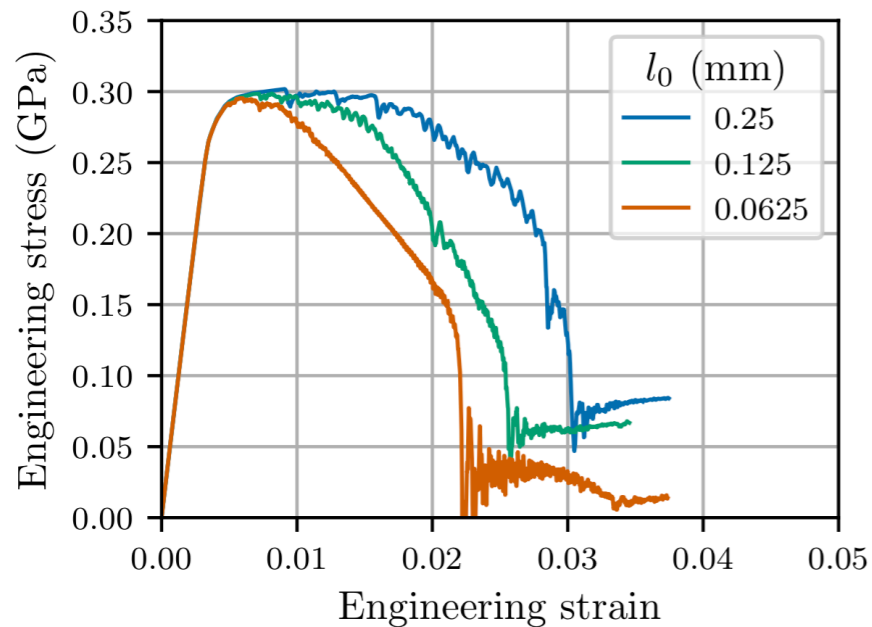
GRID



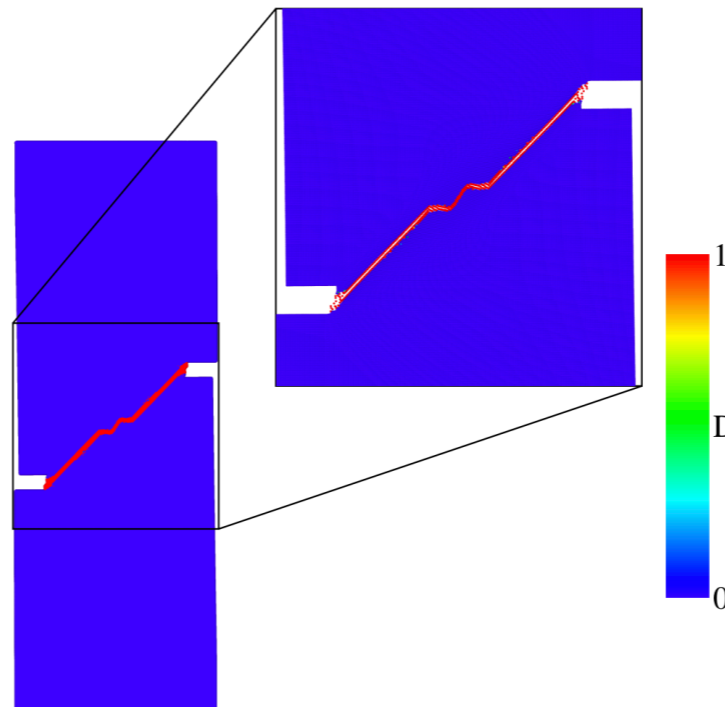
Damage mechanics approach

Local models

Nonlocal models



Local model



mesh dependent  
mesh biased

# Nonlocal gradient enhanced JC damage: length scale

---

$$\Delta D_{\text{init}} = \frac{\Delta \varepsilon_p}{\varepsilon_f} \quad \varepsilon_f = (D_1 + D_2 \exp(D_3 \sigma^*)) (1 + \dot{\varepsilon}_p^*)^{D_4}$$

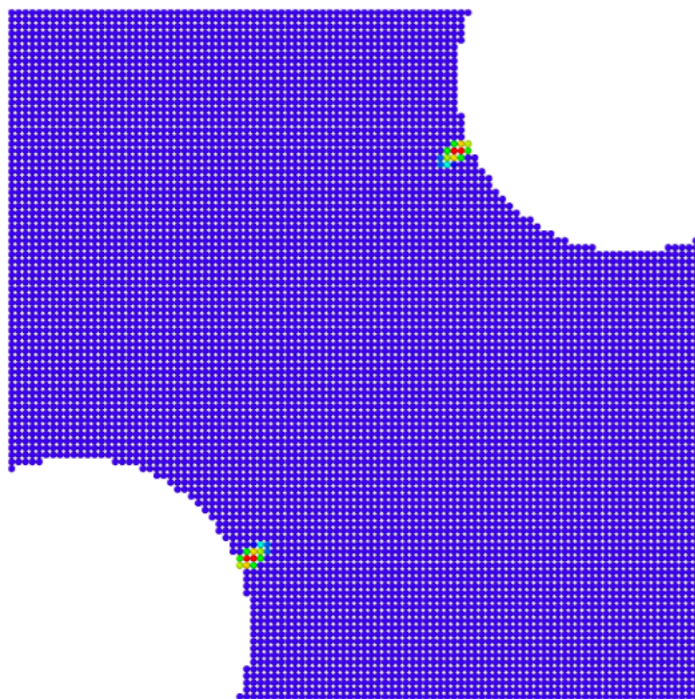
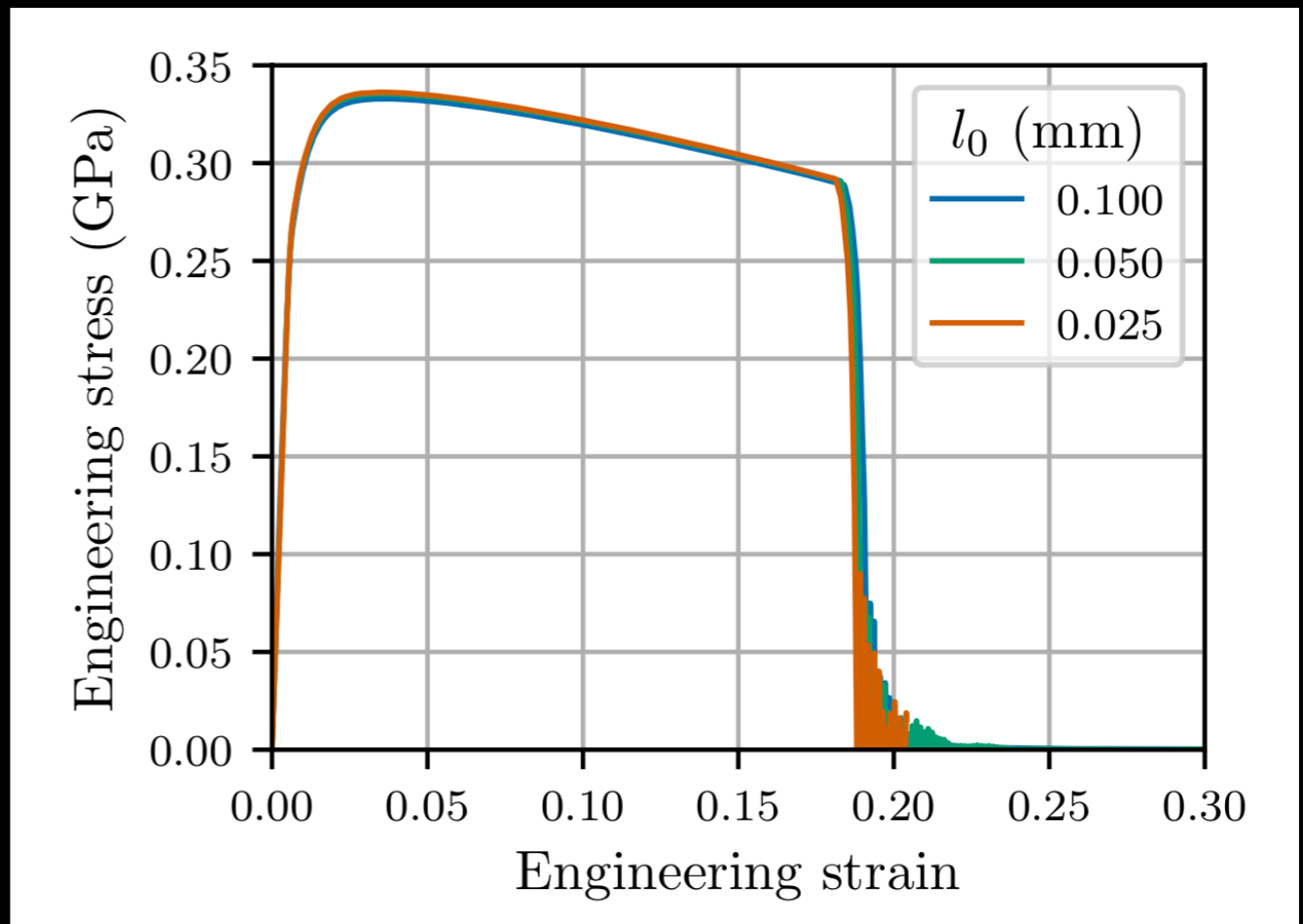
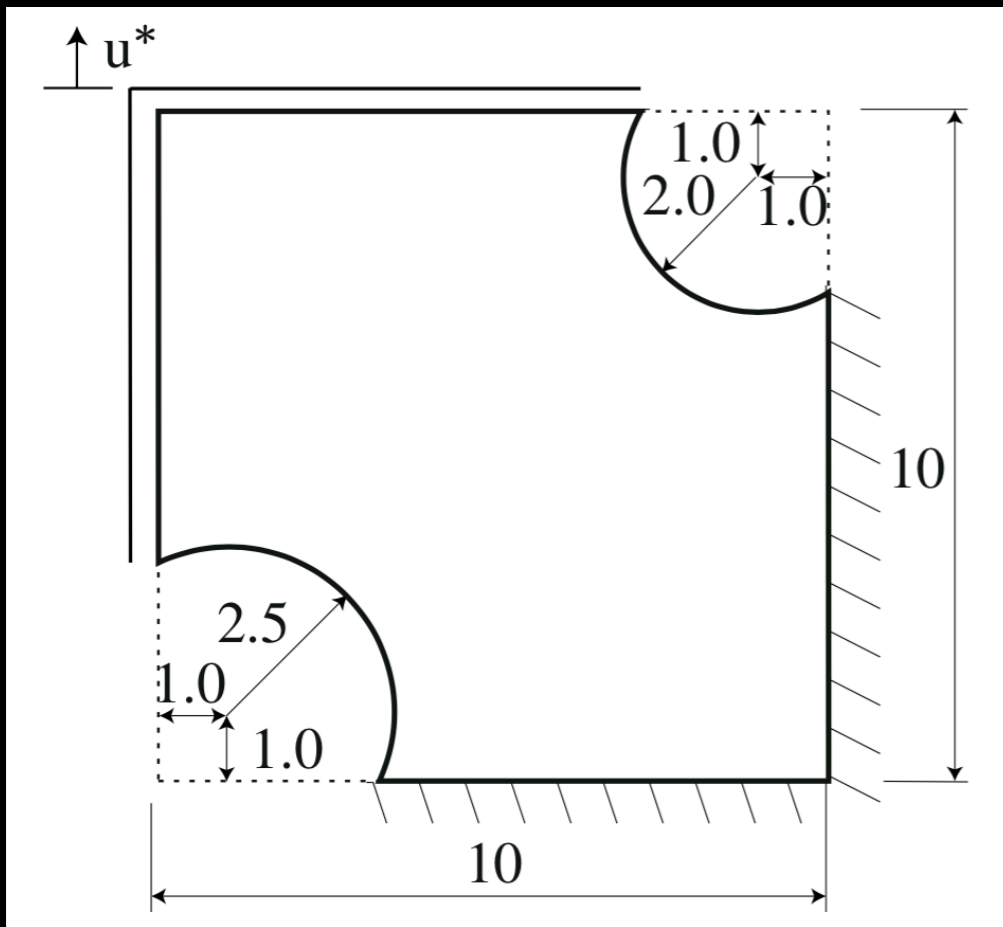
$$\langle \Delta D^{\text{init}} \rangle - c_0 \nabla_0^2 \langle \Delta D^{\text{init}} \rangle = \Delta D^{\text{init}} \quad c_0 \sim l_d^2$$

$$\langle D_{\text{init}} \rangle := \sum \langle \Delta D_{\text{init}} \rangle$$

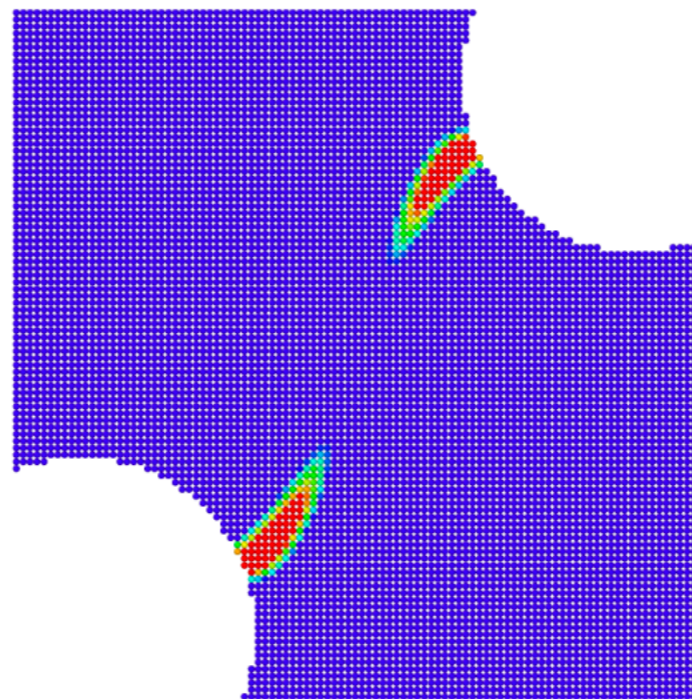
$$D = \begin{cases} 0 & \text{when } 0 \leq \langle D_{\text{init}} \rangle < 1 \\ 10 (\langle D_{\text{init}} \rangle - 1) & \text{when } \langle D_{\text{init}} \rangle \geq 1 \end{cases}$$

$$\sigma = (1 - D) \bar{\sigma}$$

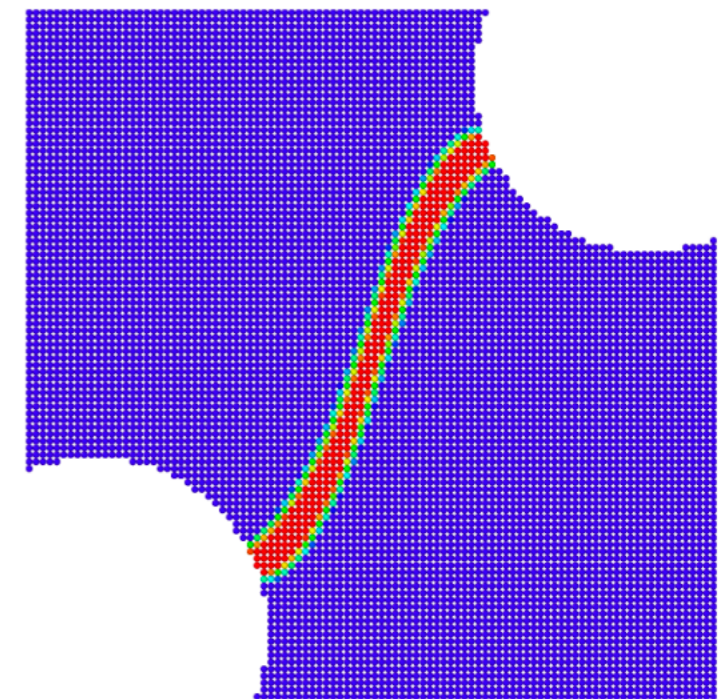
# TLMPM—modelling ductile fracture: result 1



(a)  $\epsilon_p = 1.069$

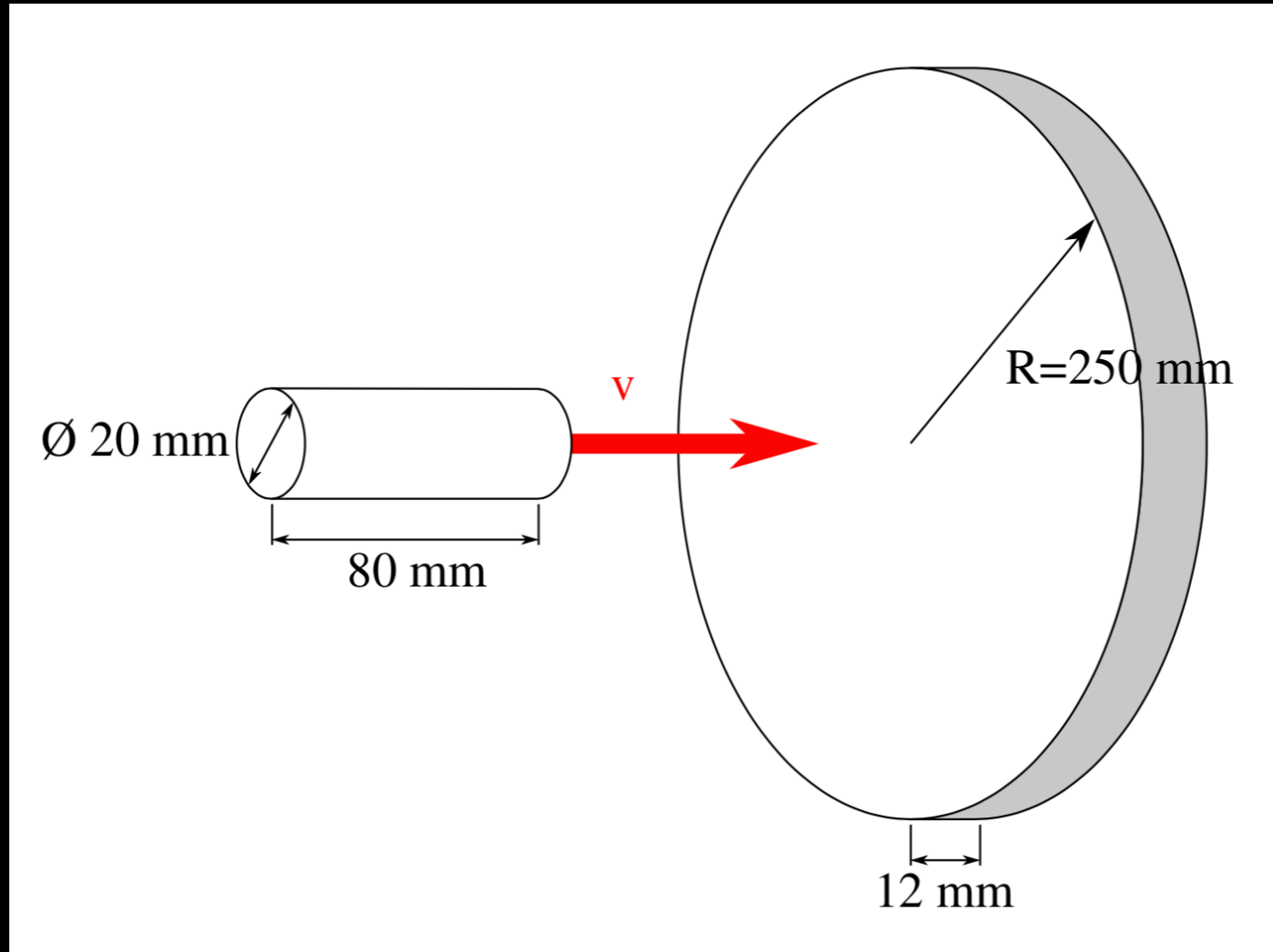


(b)  $\epsilon_p = 1.583$



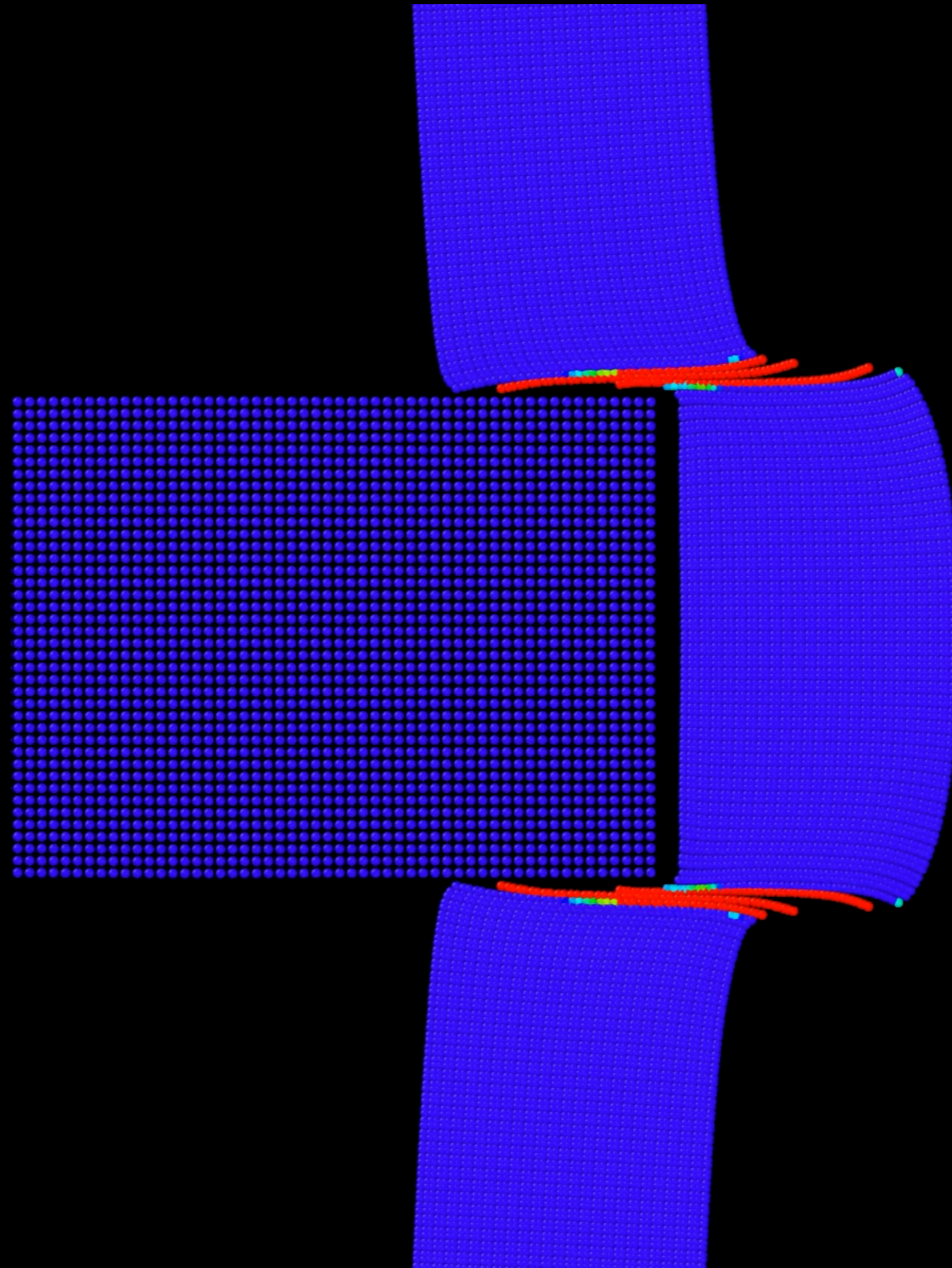
(c)  $\epsilon_p = 1.584$

# TLMPM—modelling ductile fracture: result 2

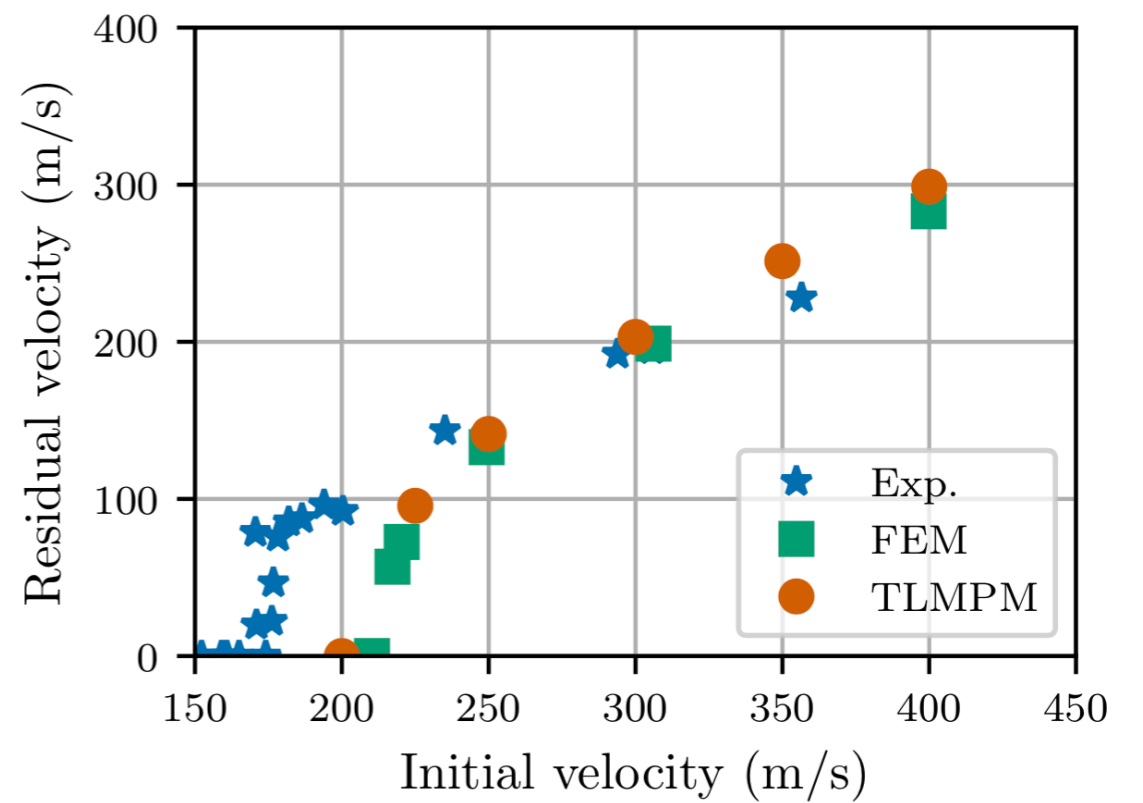
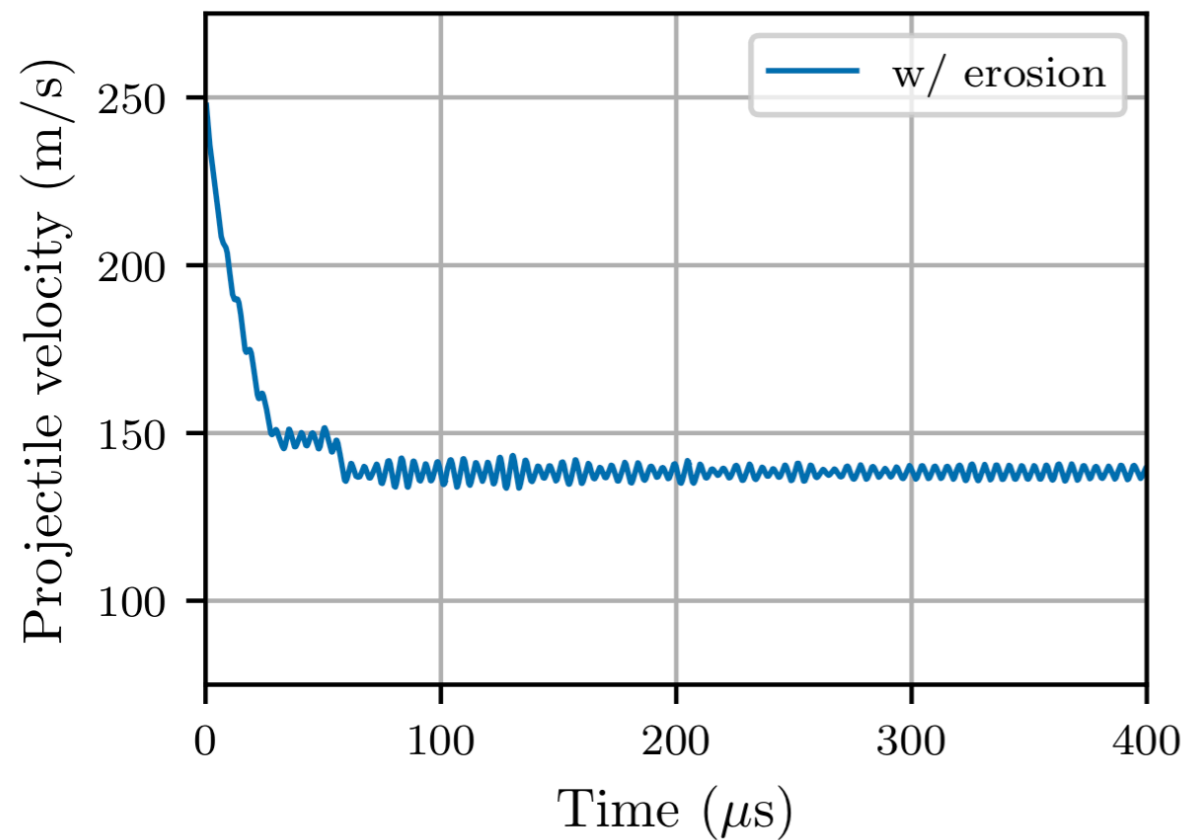


# TLMPM—modelling ductile fracture: result 2

---



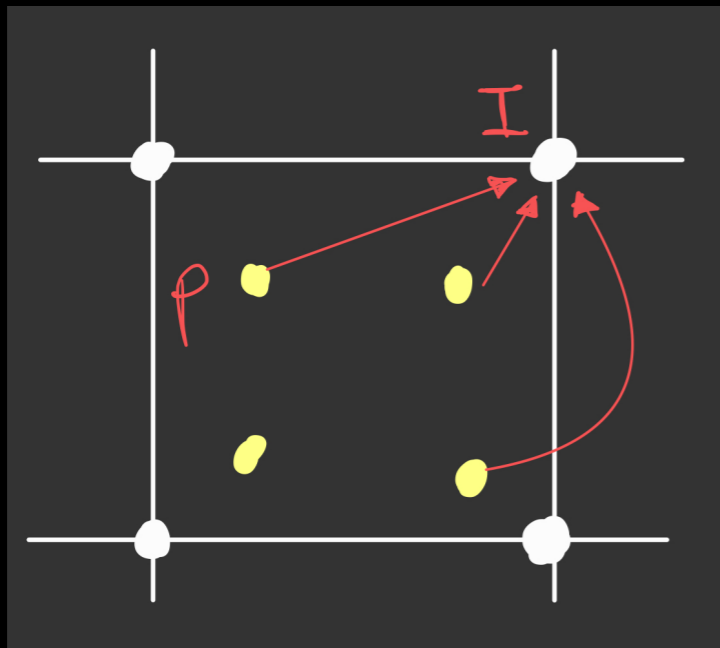
# TLMPM—modelling ductile fracture: result 2





# Summary

---



no mesh generation  
simple basis functions  
large deformation  
multiple contacts  
simple BC treatment  
simple implementation

approximate geometry  
blur material interfaces  
numerical fracture: ULMPM

# Effect of hardening

$$\begin{aligned} E_0 &= 1.4 \times 10^5 \\ \theta_c &= 2.5 \times 10^{-2} \\ \theta_s &= 7.5 \times 10^{-3} \\ \xi &= 5 \end{aligned}$$



$$\begin{aligned} E_0 &= 1.4 \times 10^5 \\ \theta_c &= 2.5 \times 10^{-2} \\ \theta_s &= 7.5 \times 10^{-3} \\ \xi &= 10 \end{aligned}$$



*The end*

Phu Nguyen

Department of Civil Engineering  
Monash University

[phu.nguyen@monash.edu](mailto:phu.nguyen@monash.edu)

<http://nvinhphu.wixsite.com/mysite>