LA-UR-19-32117
Approved for public release; distribution is unlimited.

Title: Modern Approaches in the Material Point Method

Author(s): Long, Christopher Curtis
Moutsanidis, Georgios
Bazilevs, Yuri

Intended for: Invited Seminar Talk at the University of Alabama

Issued: 2019-12-04

Modern Approaches in the Material Point Method

Christopher Long, LANL T-3
Georgios Moutsanidis, Brown University
Yuri Bazilevs, Brown University

Outline

- PART I -
  - Introduction to the fundamentals of MPM
  - Brief history of how it has historically been used, and the primary complications that it presents
  - Brief discussion of DDMP/GIMP/CPDI
- PART II
  - An Intro to Isogeometric Analysis
  - The usage of IGA and MPM together to solve the crossing noise problem
  - The usage of a single velocity field to model fracture and multi-body interactions

Part I
A brief MPM overview

What is MPM?

- The Material Point Method is a hybrid Lagrangian/Eulerian scheme
  - Advantages are that it can model extreme deformations without remeshing or entanglement
  - Disadvantages are that accuracy is hard to guarantee, and there are a few pathological issues
- The MPM is an advanced type of Particle in Cell (PIC) method, the essentials of which were first proposed by Frank Harlow in 1955
  - Harlow, F. H. (1955). “A Machine Calculation Method for Hydrodynamic Problems,” Los Alamos Scientific Laboratory, LAMS-1956, 94p, New Mexico.

PIC

- Particle in Cell methods have a Lagrangian particle and an Eulerian grid
- Particle data and grid data are exchanged by interpolatory schemes and the physics are solved on the Eulerian grid. The particles are used to model convection.
- The particle does not hold any material state data, as all parameters are overwritten at each time step with values from the grid

FLIP

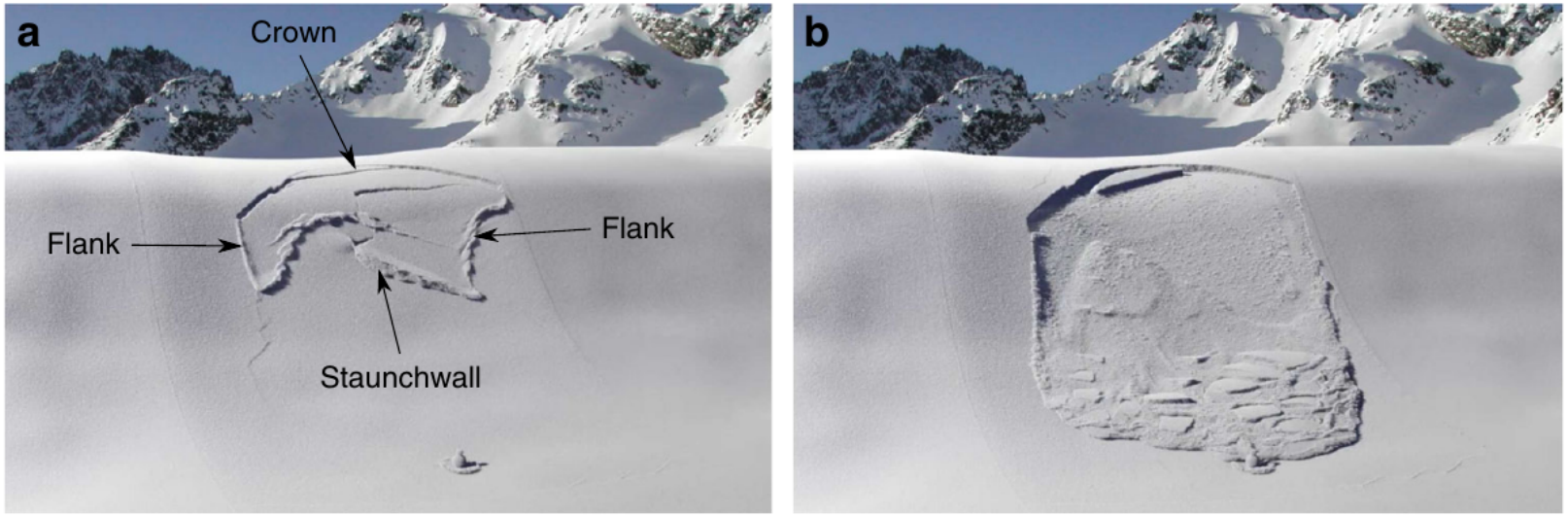
- The Fluid Implicit Particle Method (FLIP) was introduced in 1986 by Jerry Brackbill as a means to reduce the overly diffusive solutions inherent to PIC methods.
  - Brackbill, J.U. and Ruppel, H.M., 1986. FLIP: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. Journal of Computational physics, 65(2), pp.314-343.
- Introduces the concept of state variables residing primarily on the particles rather than the grid.
- Particle receives updates on velocity from the grid, but no data is overwritten.

MPM

- Sulsky, et al, developed the classic formulation of the Material Point Method in 1994
    - Sulsky, D.; Chen, Z.; Schreyer, H. L. (1994). "A particle method for history-dependent materials". Computer Methods in Applied Mechanics and Engineering. 118 (1): 179–196.
- The key innovation is the introduction of finite element shape function on the grid to perform data interpolation and integration to and from the grid/particle.
    - Previous methods had used cloud in cell or other earest grid point type schemes

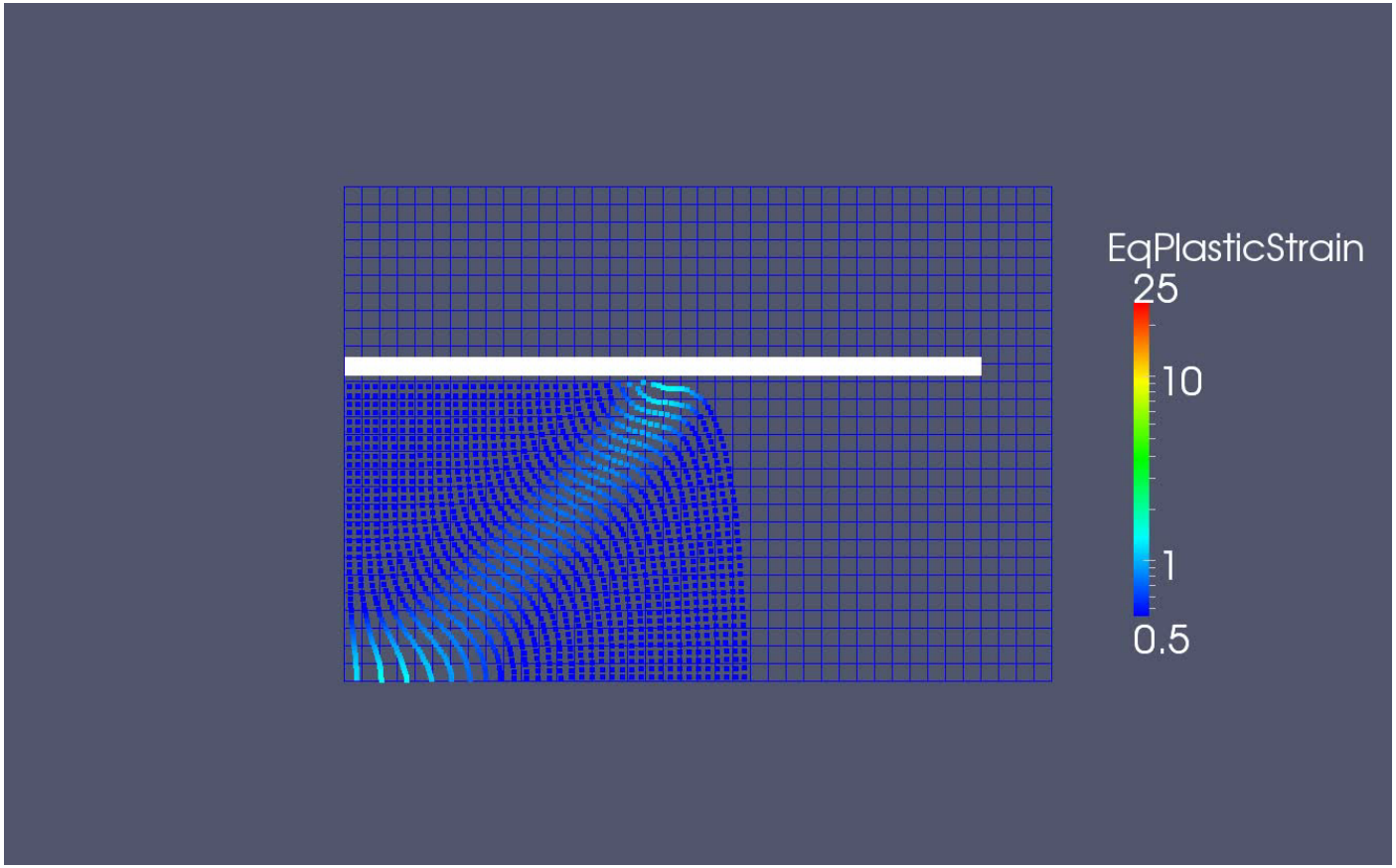
Uses of MPM

- Popular in computer animation and video game graphics
  - Fast and can perform large deformations

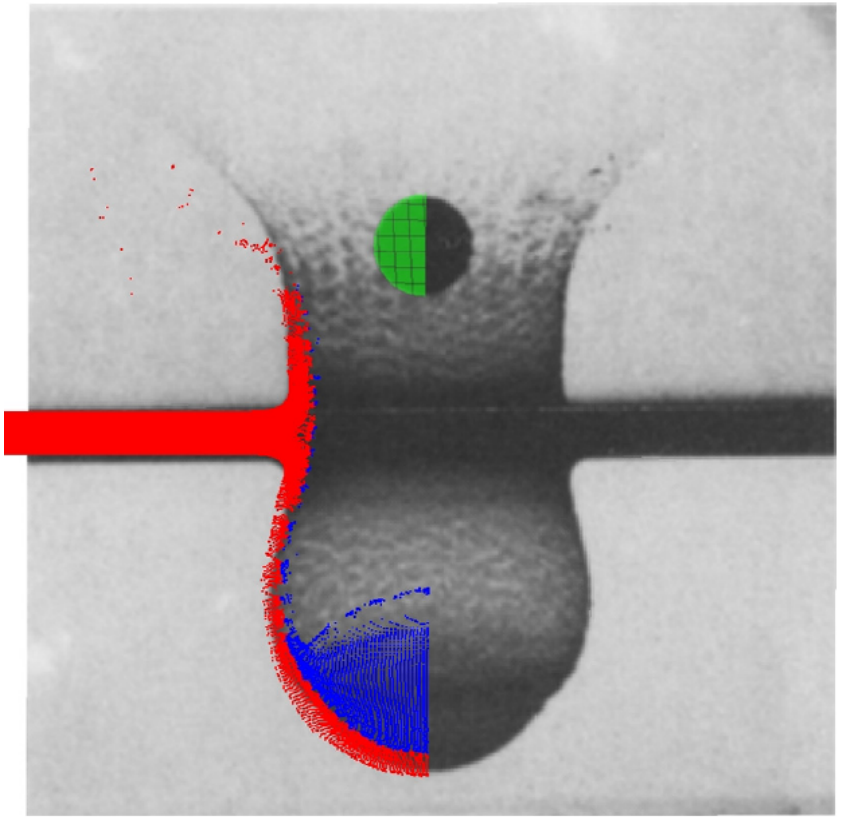


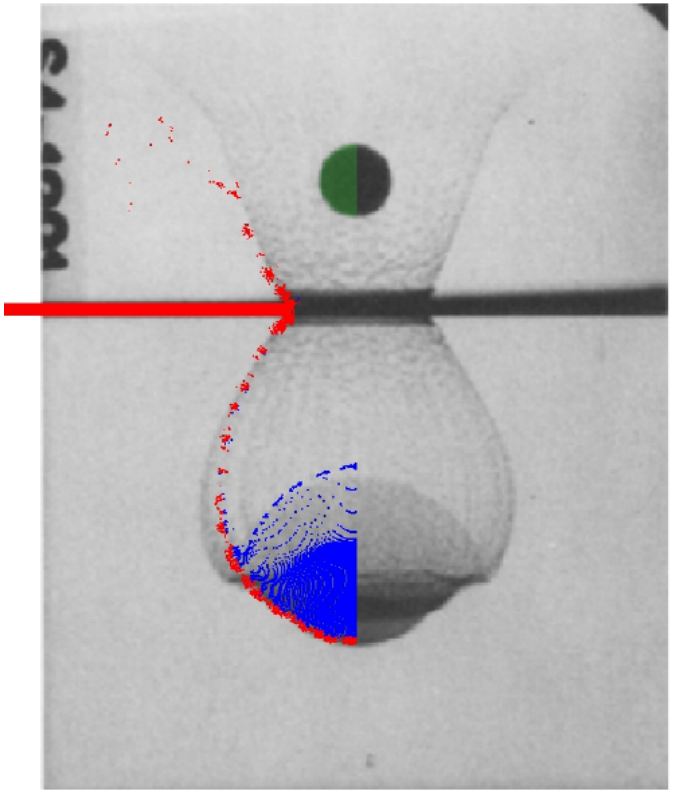
Gaume, J., Gast, T., Teran, J., van Herwijnen, A. and Jiang, C., 2018. Dynamic anticrack propagation in snow. Nature communications, 9(1), p.3047.

Large Deformations



Pulverizations

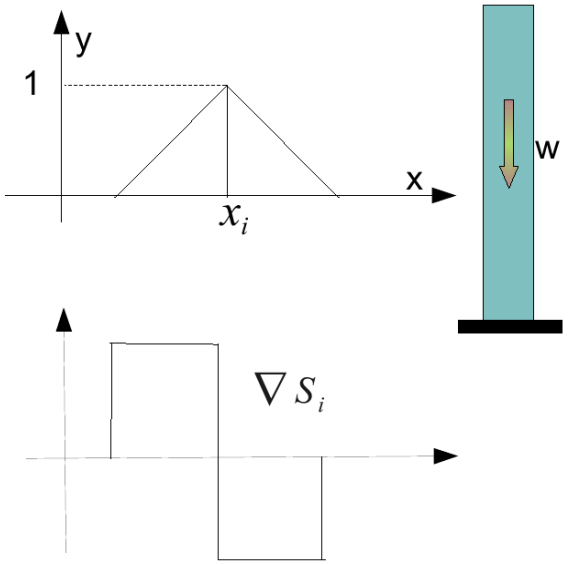


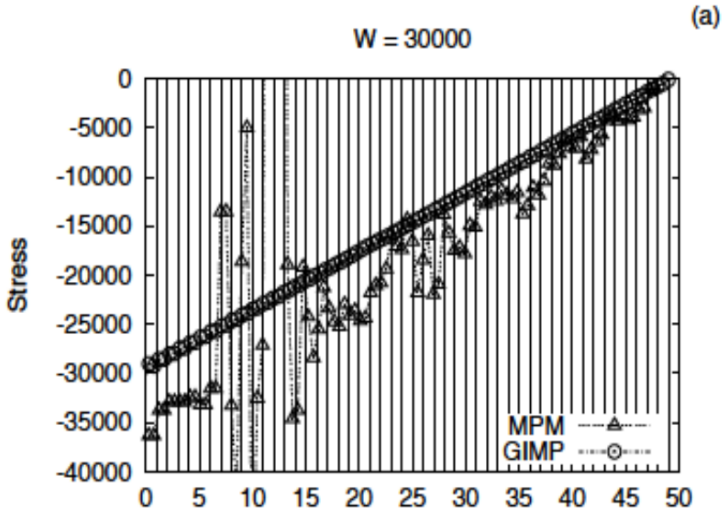


Crossing Noise Problem

$$m_i \frac{d\boldsymbol{v}_i}{dt} = -\sum_p V_p \boldsymbol{\sigma}_p \cdot \nabla S_i(\boldsymbol{x}_p) + \int \rho \boldsymbol{g} S_i(\boldsymbol{x}) dV + \int_{\partial v} S_i(\boldsymbol{x}) \boldsymbol{\sigma} \cdot \boldsymbol{n} dS ,$$

The discontinuity of shape function gradient causes an instability



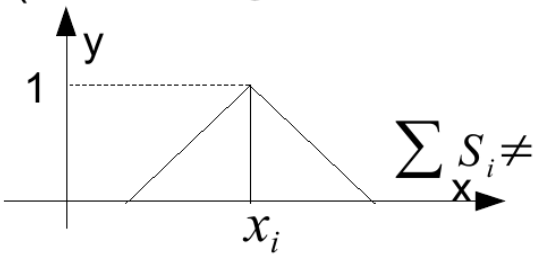


S.G. Bardenhagen, E.M. Kober, The generalized interpolation material point method, Comput. Model. Eng. Sci. 5 (6) (2004) 477–495.

GIMP, DDMP

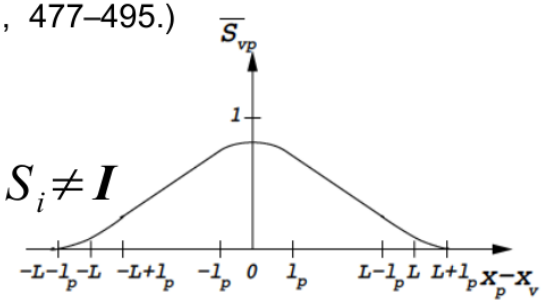
Generalized interpolation material point (GIMP) method

(S.G. Bardenhagen, E.M. Kober, 2004, Comput. Model. Eng. Sci. 5 (6), 477–495.)



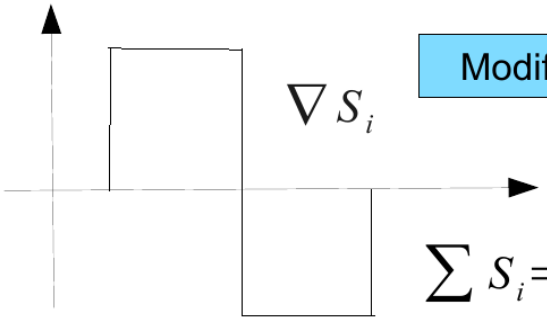
Modifies shape function

$$\sum S_i \neq 1, \quad \sum \nabla S_i \neq \mathbf{0}, \quad \sum x_i \nabla S_i \neq \boldsymbol{I}$$



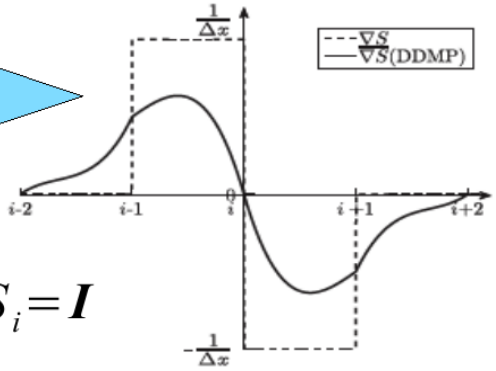
Dual domain material point (DDMP) method

(Zhang et al., 2011, JCP, 230, 6379-6398.)



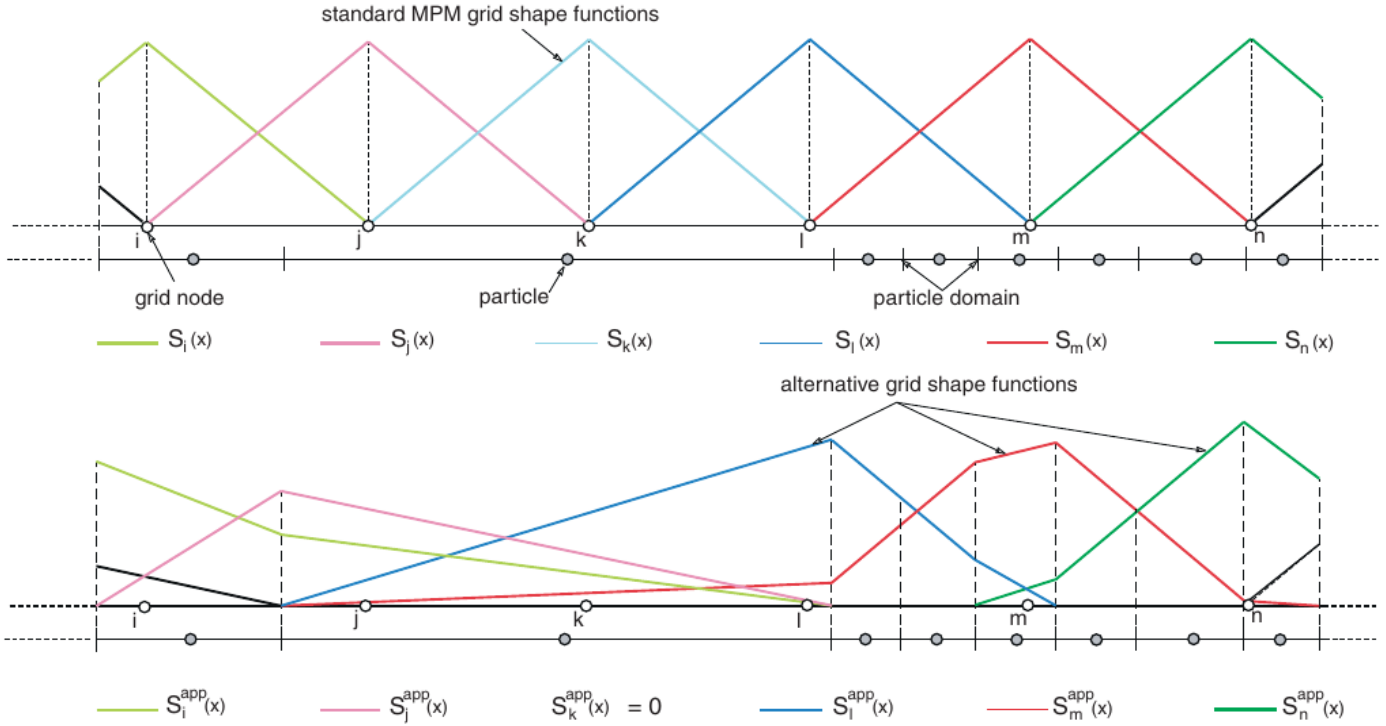
Modifies gradient of shape function

$$\sum S_i = 1, \quad \sum \nabla S_i = \mathbf{0}, \quad \sum x_i \nabla S_i = \boldsymbol{I}$$



CPDI

- Also modifies the underlying basis functions
- Retains linear completeness



Sadeghirad, A., Brannon, R.M. and Burghardt, J., 2011. A convected particle domain interpolation technique to extend applicability of the material point method for problems involving massive deformations. IJNME 86(12), pp.1435-1456.

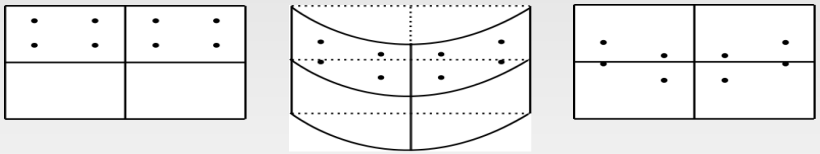
CartaBlanca MPM code used for modeling

Capable of running in thread-parallel and distributed-parallel modes

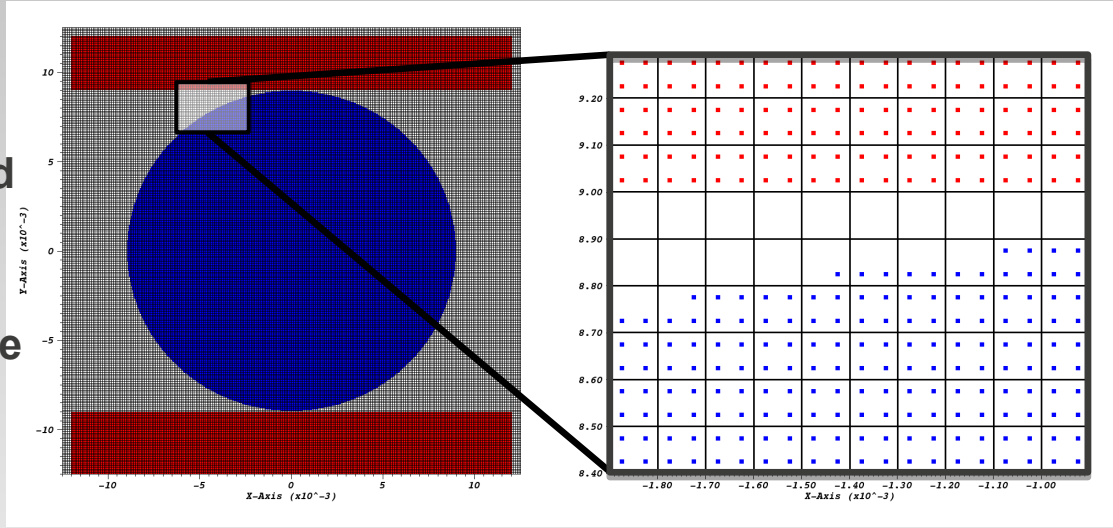
Uses Material Point Method (MPM), an advanced Particle-in-Cell Method

- **Multiphase Flow Framework**
- **Capable of strongly coupled Fluid-Structure Interaction simulations**

Well-suited for <u>fracture</u> problems and <u>large material deformations</u>



F.H. Harlow, *Methods Comput. Phys.* **3** (1964) 319.



- **Based off a Particle-in-Cell method**
- **Stationary grid is seeded with Lagrangian 'particles' which can move freely in and out of cells**
- **Grid quantities are advected, and particles move during time step (flow between cells)**
- **Grid 'snaps' back to original configuration during update**

Part II
Isogeometric Analysis, MPM, and the multi-body problem

What is IGA, and why do we care?

- It’s easier to understand than most people think
- Analysis fundamentally follows a finite element framework
  - Traditional FEA can be thought of as a special case of first-order IGA
- IGA has many advantages (and a few disadvantages) over FEA
  - C2 continuity of shape functions gives very smooth solution space
  - Complex geometry can be represented exactly (machine error only)

IGA concepts

- The mesh is constructed secondarily from the shape functions we create specifically to create it. I.E., we define the shape functions first, and the mesh and analysis capabilities both follow
- We define the number of shape functions and their behavior from a few “control points” and a recursion formula

Control Points and “knot” vectors

- We define a “knot vector” which includes the locations of the “knots”, which can be thought of as mesh boundaries.

$$\Xi = \{\xi_1, \xi_2, \ldots, \xi_{n+p+1}\}$$

- ‘n’ is the number of basis functions used to define the geometry, and ‘p’ is the desired polynomial order
- The vector must be non-decreasing, but can use repeated indices and may be non-uniform.
- The values correspond to the location of eventual element boundaries in 1-D

Control Points and “knot” vectors

- In 2 or more dimensions, we require ‘d’ knot vectors in an orthonormal framework.
- We use the Cox-de-Boor recursion formula to then define the shape functions as follows:

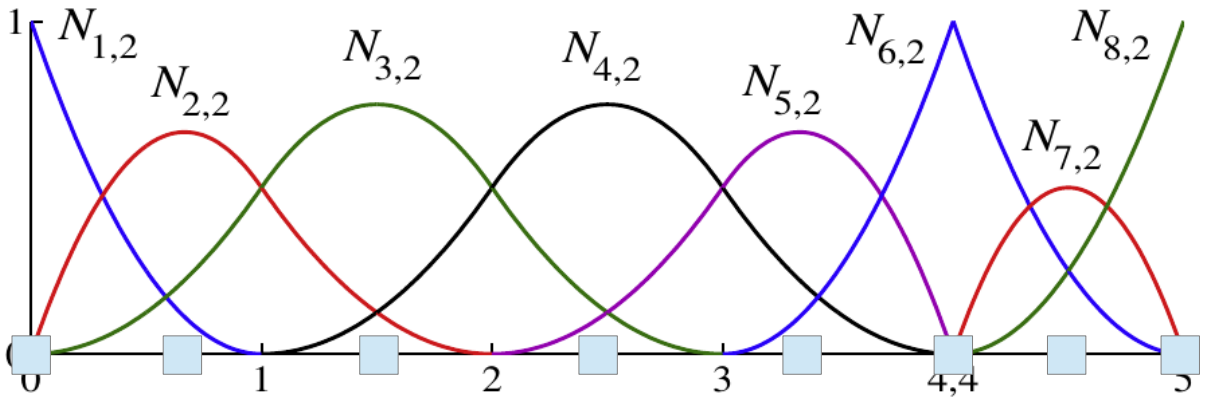
$$N_{i,0}(\xi) = \begin{cases} 1 & \text{if } \xi_i \leqslant \xi < \xi_{i+1}, \\ 0 & \text{otherwise.} \end{cases}$$

$$N_{i,p}(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} N_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} N_{i+1,p-1}(\xi)$$

1-D example

- This formula guarantees a partition of unity everywhere in the domain.

$$\Xi = \{0, 0, 0, 1, 2, 3, 4, 4, 5, 5, 5\}$$



Transformations to represent conical geometries in multi-D

- If no other adjustments are made, we may think of the ‘mesh’ generated by the previous constructions in multi-d as a rectilinear grid, with each ‘knot’ representing a point/line/plane for 1/2/3-D, respectively. Intersections are referred to as “knots”, and the “control points” may be considered as nodal points in traditional FEA.
- *But this does not have to be the case….*

Transformations to represent conical geometries

- We can transform our mesh using a series of weights on each shape function as follows to define a shape:

$$R_i^p(\xi) = \frac{N_{i,p}(\xi) w_i}{W(\xi)}$$

$$W(\xi) = \sum_{i=1}^{n} N_{i,p}(\xi) w_i$$

$$\mathbf{C}(\xi) = \sum_{i=1}^{n} R_i^p(\xi) \mathbf{B}_i$$

Multi-D transformations

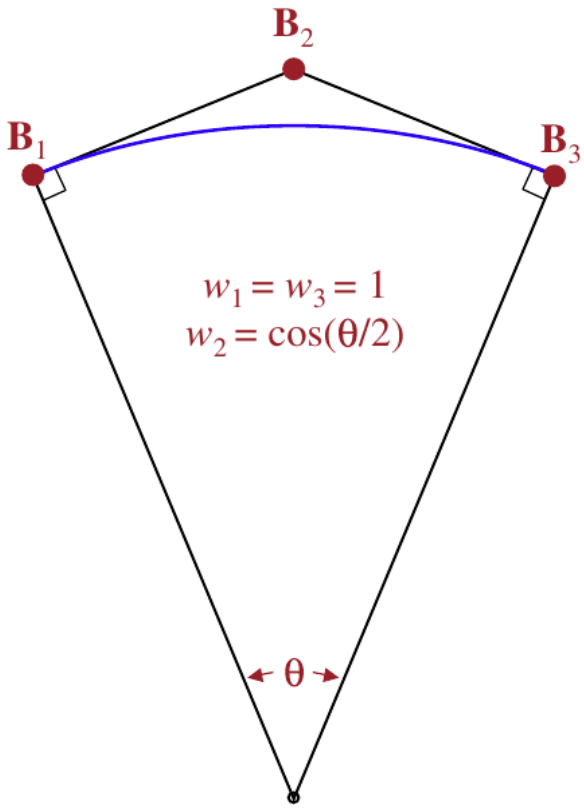
- Due to the nature of their construction, we can extend to multi dimensional analogs straightforwardly, and define new shape functions for surfaces and volumes:

$$R^{p,q}_{i,j}(\xi,\eta) = \frac{N_{i,p}(\xi)M_{j,q}(\eta)w_{i,j}}{\sum_{\hat{i}=1}^{n}\sum_{\hat{j}=1}^{m}N_{\hat{i},p}(\xi)M_{\hat{j},q}(\eta)w_{\hat{i},\hat{j}}},$$

$$R^{p,q,r}_{i,j,k}(\xi,\eta,\zeta) = \frac{N_{i,p}(\xi)M_{j,q}(\eta)L_{k,r}(\zeta)w_{i,j,k}}{\sum_{\hat{i}=1}^{n}\sum_{\hat{j}=1}^{m}\sum_{\hat{k}=1}^{l}N_{\hat{i},p}(\xi)M_{\hat{j},q}(\eta)L_{\hat{k},r}(\zeta)w_{\hat{i},\hat{j},\hat{k}}}$$

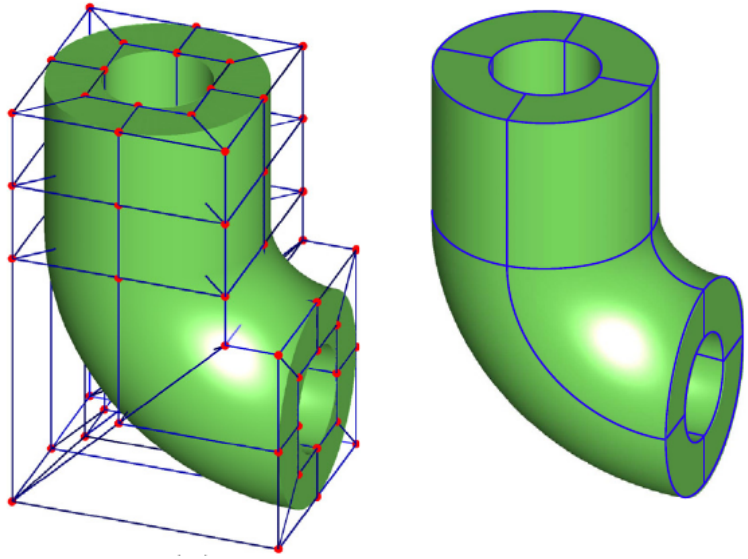
A 1-D example

- Notice that control point 2 does not lie on our arc
- The location of a control point and the location of the constructed mesh where analysis takes place are not generally the same, unless all weights are equal
- Control points can often be outside of the resulting “element”



3-D Example

- The control mesh is analogous to the mesh used for FEA. We can see on the left that the control mesh and the modeled object are radically different.
- Different sections of the resulting object are created uniquely and glued together in a process called ‘patching’



IGA- some practical issues

- Creating a complex mesh is hard to do by hand. However, CAD software generally uses NURBS or T-Splines to render, and so suitable domains for analysis may be created *graphically*.
- Meshing an area based on a rendered CAD volume has been a standard practice for years, however, the meshing step can be difficult, time consuming, and complex. IGA negates the necessity to mesh entirely, as the same functions used by CAD software to render the object can simultaneously be used for analysis!

IGA used in Lagrangian Hydro

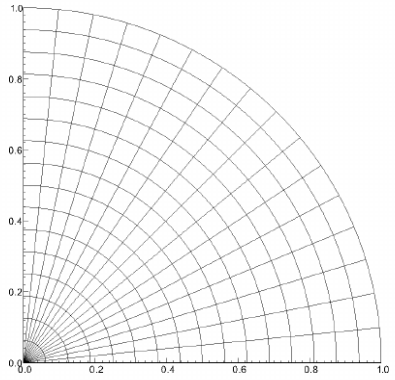
- “Coggeshall” problem

$$\rho = (1-t)^{-9/4},$$
$$e = \left(\frac{3z}{8(1-t)}\right)^2,$$

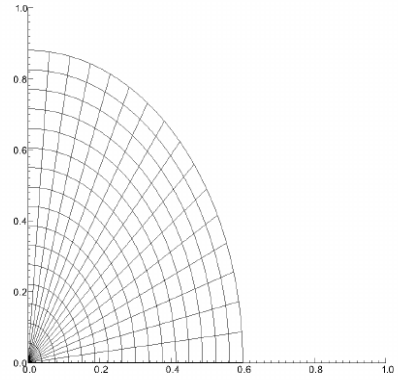
r = (1 − t)R,
z = (1 − t)1/4 Z,

$$v_r = -R = \frac{-(1-t)R}{(1-t)} = \frac{-r}{(1-t)},$$
$$v_z = \frac{-(1-t)^{-3/4}Z}{4} = \frac{-(1-t)^{1/4}Z}{4(1-t)} = \frac{-z}{4(1-t)}.$$

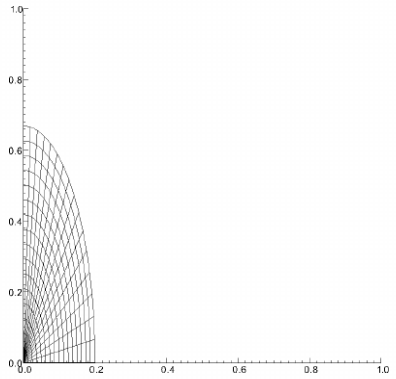
	ρ	p	v_r	v_z
3GP, Δt	7.205E-004	8.801E-005	1.676E-006	6.187E-005
4GP, $\Delta t/2$	8.630E-007	2.697E-007	3.616E-009	5.003E-008
5GP, $\Delta t/4$	5.447E-008	3.520E-008	9.118E-010	3.548E-010



(a)

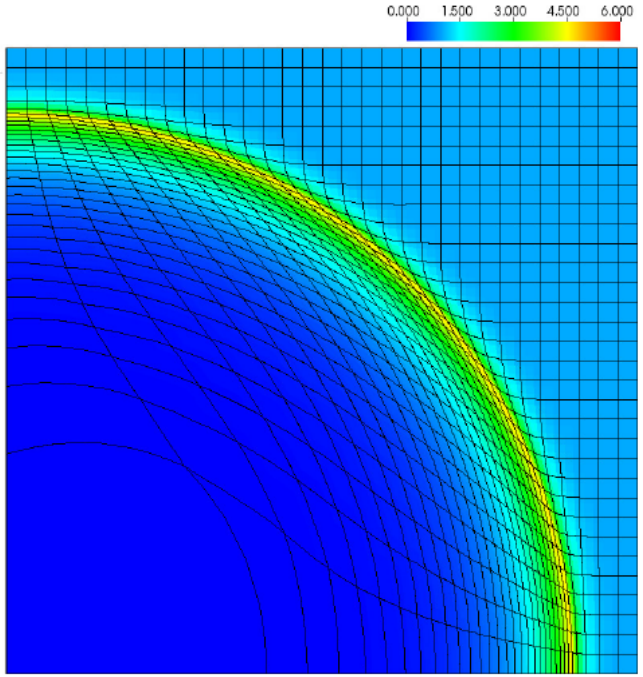


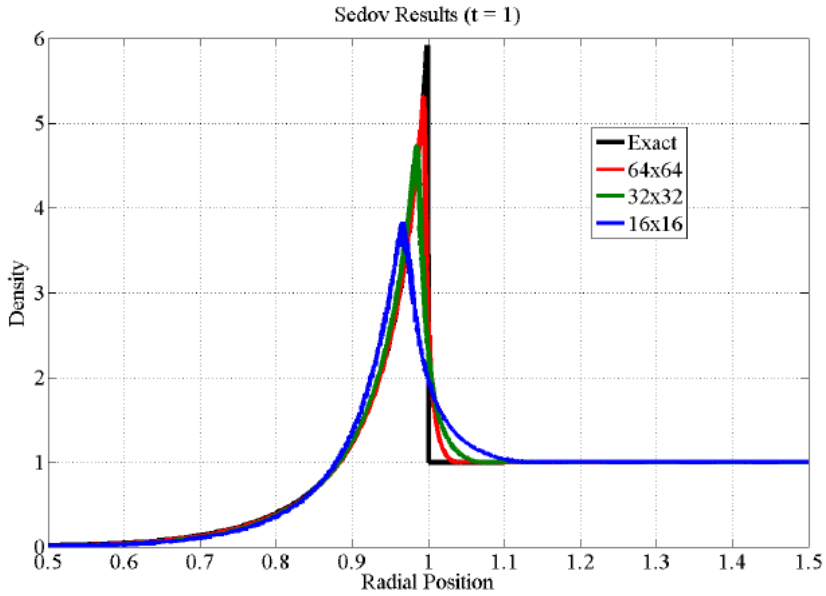
(b)



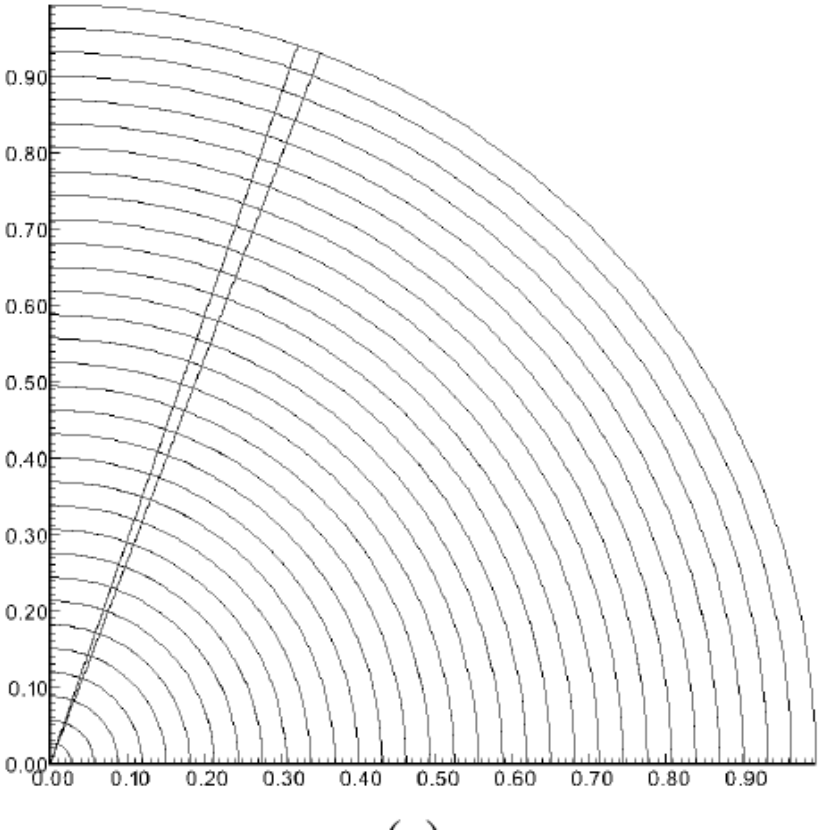
(c)

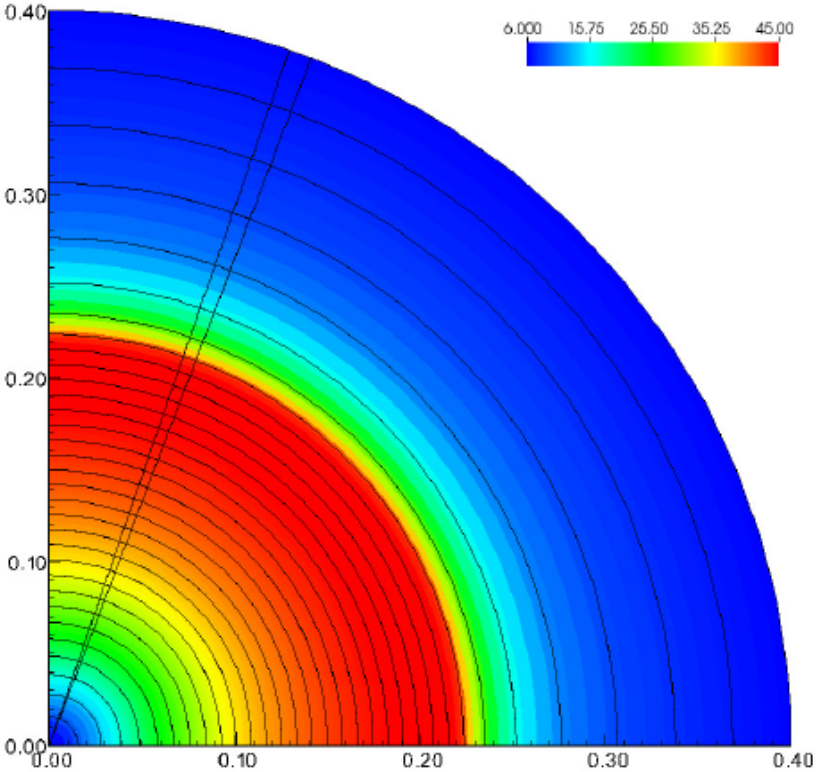
Sedov Blast


0.000
1.500
3.000
4.500
6.000


Sedov Results (t = 1)
Density
Radial Position
Exact
64x64
32x32
16x16
0
1
2
3
4
5
6
0.5
0.6
0.7
0.8
0.9
1
1.1
1.2
1.3
1.4
1.5

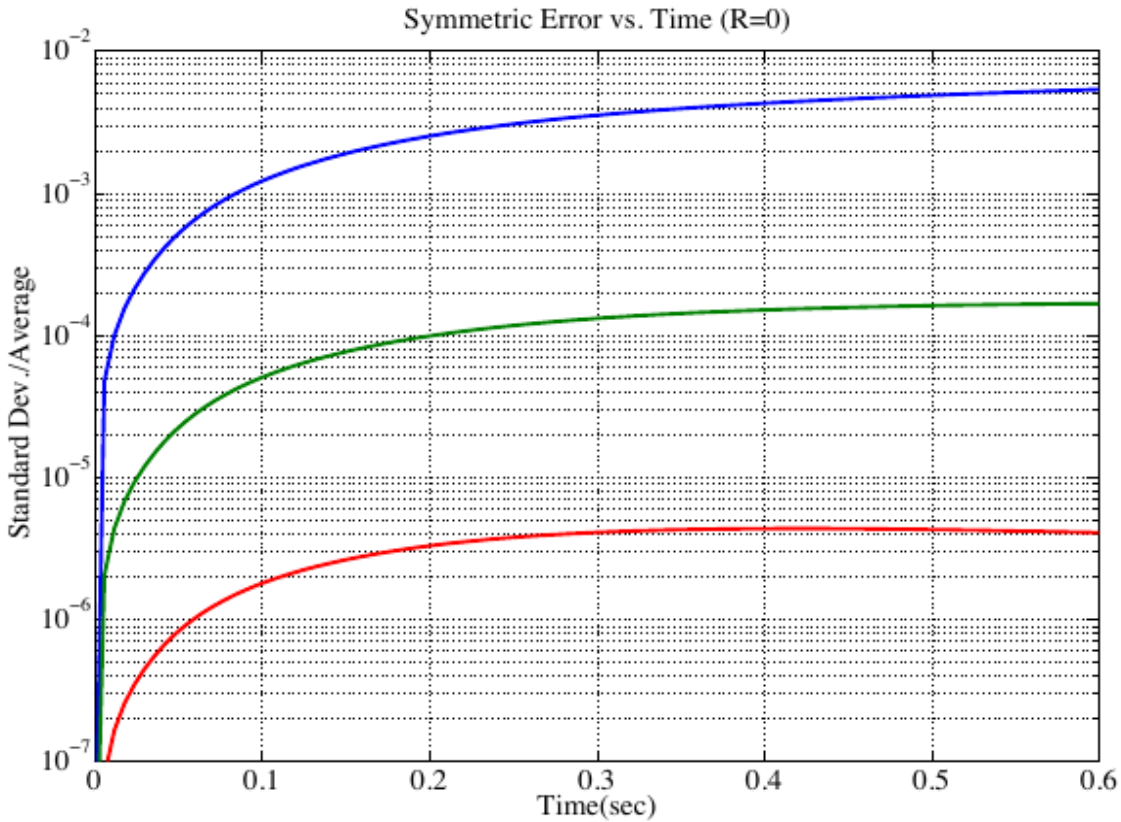
Noh on a TERRIBLE mesh





Noh, cont'd

- Poor gauss integration is the sole source of numerical error!

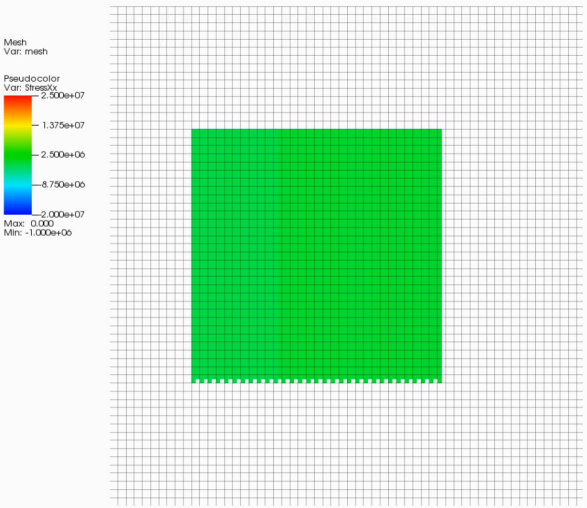


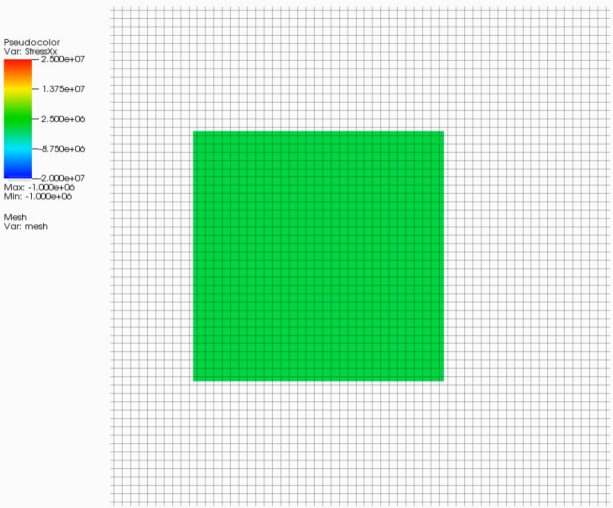
Discussion on MPM and IGA

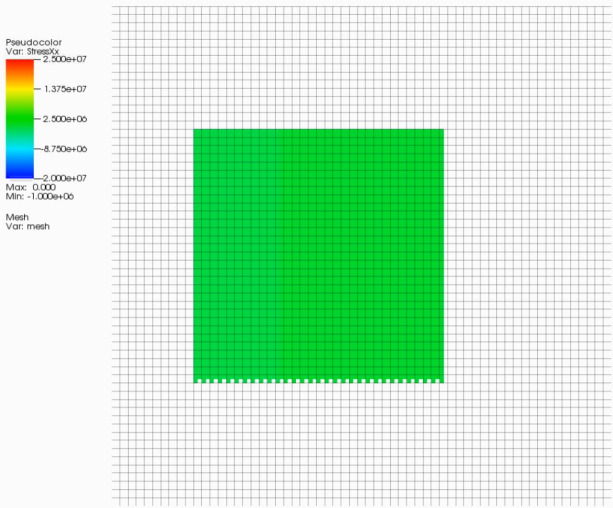
- We theorize that Gauss integration is not necessary for traditional IGA, and that 2nd order accuracy can be attained using far fewer integration points than GI requires.
- The derivatives of the shape functions are continuous, and thus the “crossing” instability is eliminated at a fundamental level
- For cylindrical problems, this has the potential to reduce mesh imprinting on the solution due to the fact that the shape functions respect the radial nature of the problem
- Stencil is still compact!

MPM and IGA

- Carta Blanca has been modified to use NURBS functions as basis functions
- For a rectilinear grid with no transformations, this is simply a matter of replacing the shape functions used and grid structure.
- For more complex geometries where the control mesh and ‘physical’ mesh are different, we will be required to transform the particles as well.

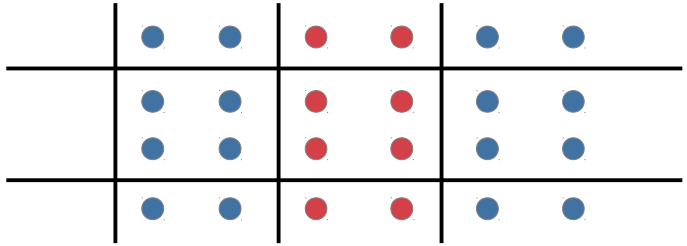

Mesh
Var: mesh
Pseudocolor
Var: StressXx
2.500e+07
1.375e+07
2.500e+06
-8.750e+06
-2.000e+07
Max: 0.000
Min: -1.000e+06


Pseudocolor
Var: StressXx
2.500e+07
1.375e+07
2.500e+06
-8.750e+06
-2.000e+07
Max: -1.000e+06
Min: -1.000e+06
Mesh
Var: mesh

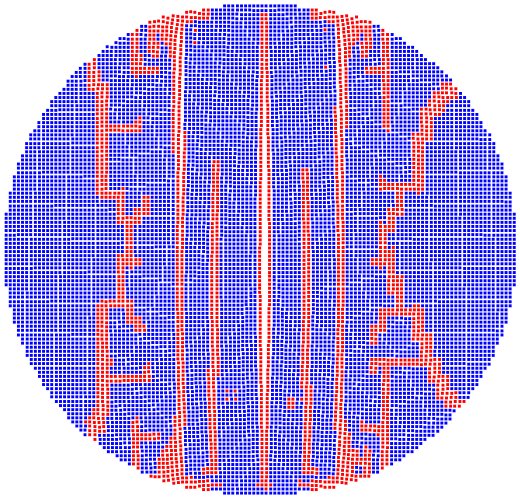

Pseudocolor
Var: StressXx
2.500e+07
1.375e+07
2.500e+06
-8.750e+06
-2.000e+07
Max: 0.000
Min: -1.000e+06
Mesh
Var: mesh

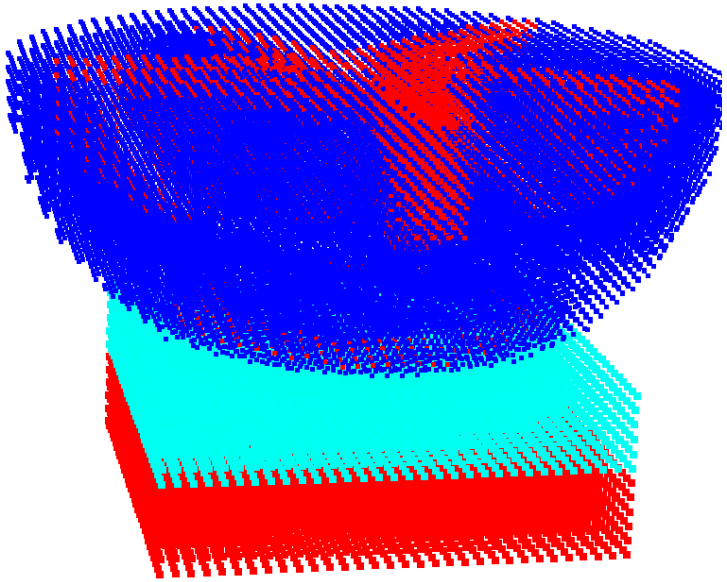
MPM crack and multi-body problem

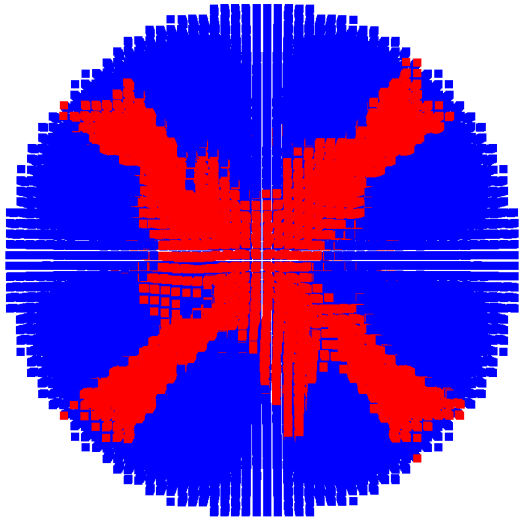
- Because particle information is integrated to the Eulerian grid, damage from a crack has to be at least one grid cell thick to naturally partition (2 grid cells in the case of DDMP)
- Two bodies sharing a velocity field will artificially `weld‘ upon contact



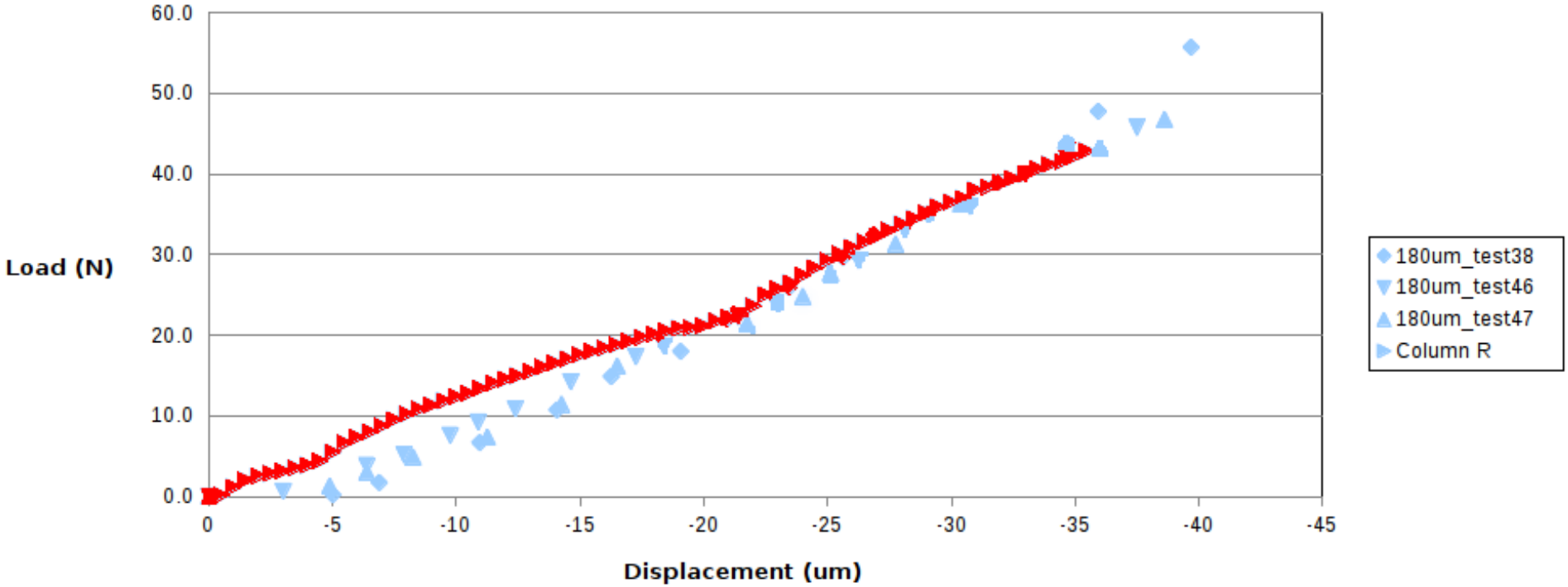
MPM Crack Thickness







Load to Failure

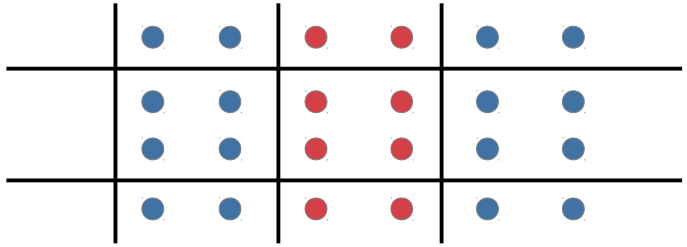


MPM crack problem

- This effect is due to the fact that the particles in a single sphere are all initialized as the same material, and are integrated as a unified field of velocity, stress, etc.
- Solution is to create an algorithmic “crack/surface finder“, which will both identify subgridscale cracks based on the damage field, and determine their orientation
  - Michael Homel at LLNL recently published an algorithm to do this within an MPM framework utilizing multiple velocity fields
  - We believe we can do this *within a single velocity field.*

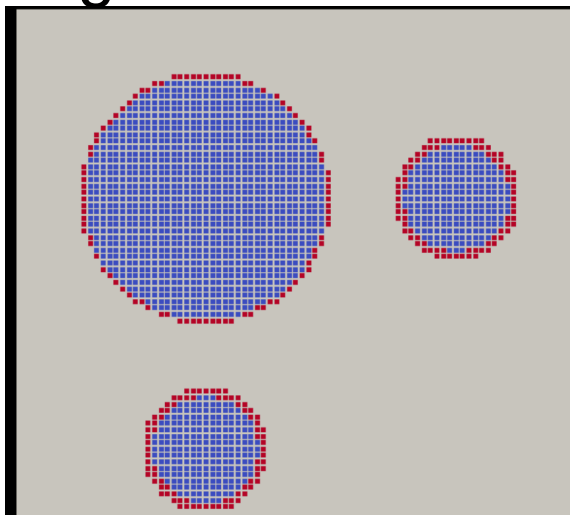
Crack finder

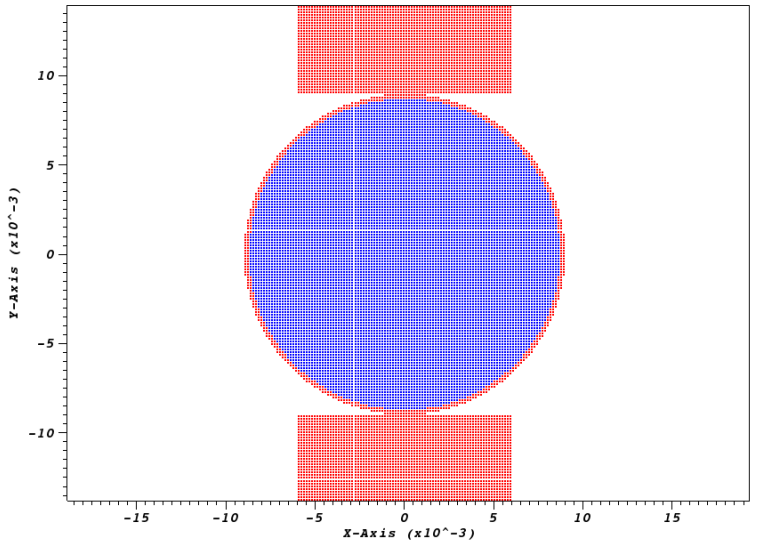
- We assign each particle its own shape functions, using SPH or RKPM
  - We currently use SPH, as it gives more consistent normals near object boundaries
  - For brittle problems, neighbors don‘t change very frequently
  - The general MPM or DDMP algorithm is unaltered, the specialty shape functions are solely used to integrate the damage field, as well as the gradient of the damage field.



Multi-Body Interactions

- Use both traditional Damage field as well as a new field called `interface‘. Interface particles are normal particles, but are on the surface of an object. This parameter gets used for determining crack position, but not in determining strength of a material.





Interface Particles

$$\overline{\nabla d}(\mathbf{x}_p) \cdot \overline{\nabla d}(\mathbf{y}_g) \begin{cases} \geq -\epsilon_{thresh} & \Rightarrow \text{particle } p \text{ and node } g \text{ are on the same side of } \Gamma_{t^0} \\ < -\epsilon_{thresh} & \Rightarrow \text{particle } p \text{ and node } g \text{ are on opposite sides of } \Gamma_{t^0} \end{cases}$$

$$I_p = \begin{cases} 1, & \text{particle p is located on the boundary of the solid} \\ 0, & \text{particle p is located in the interior of the solid} \end{cases}$$

$$\overline{\nabla I}(\mathbf{x}_p) \cdot \overline{\nabla I}(\mathbf{y}_g) \begin{cases} \geq -\epsilon_{thresh} & \Rightarrow \text{particle } p \text{ and node } g \text{ are located in the same body} \\ < -\epsilon_{thresh} & \Rightarrow \text{particle } p \text{ and node } g \text{ are located in different bodies} \end{cases}$$

$$D_p = \mathbf{max}(d_p, I_p)$$

General Approach

- Determine if particle is on or near a crack. If not, use traditional MPM/DDMP
  - Check for both magnitude of particle damage and magnitude of damage gradient field

$$\overline{\nabla d}(\mathbf{x}_p) \cdot \overline{\nabla d}(\mathbf{y}_g) \begin{cases} \geq -\epsilon_{thresh} & \Rightarrow \text{particle } p \text{ and node } g \text{ are on the same side of } \Gamma_{t^0} \\ < -\epsilon_{thresh} & \Rightarrow \text{particle } p \text{ and node } g \text{ are on opposite sides of } \Gamma_{t^0} \end{cases}$$

- If node and particle are on opposite sides of a crack, we alter the shape functions accordingly.

$$N_i^{\tau *}(\mathbf{x}) = \begin{cases} N_i(\mathbf{x}), \ \overline{\nabla D}(\mathbf{x}) \cdot \overline{\nabla D}(\mathbf{y}_i) \geq -\epsilon_{thresh} \\ 0, \text{ otherwise} \end{cases}$$

$$N_i^{\nu *}(\mathbf{x}) = \begin{cases} N_i(\mathbf{x}), \ (\overline{\nabla D}(\mathbf{x}) \cdot \overline{\nabla D}(\mathbf{y}_i) \geq -\epsilon_{thresh} \ \vee \ \widetilde{\sigma}_{11}(\mathbf{x}) \leq 0) \\ 0, \text{ otherwise} \end{cases}$$

Approach, cont‘d

- We rescale the modified shape functions to retain a partition of unity:

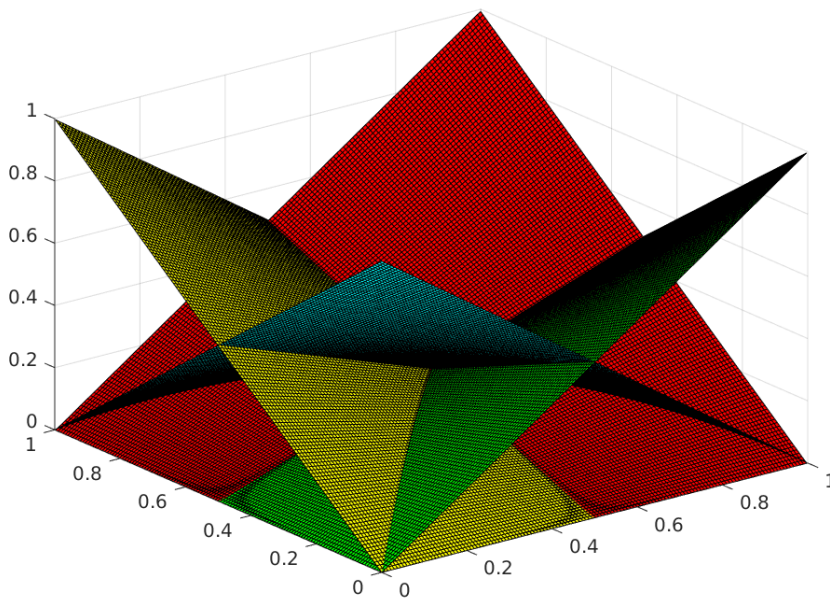
$$\widetilde{N}_i(\mathbf{x}) = \frac{N_i^*(\mathbf{x})}{\sum_{j=1}^{n_{\text{node}}} N_j^*(\mathbf{x})}$$

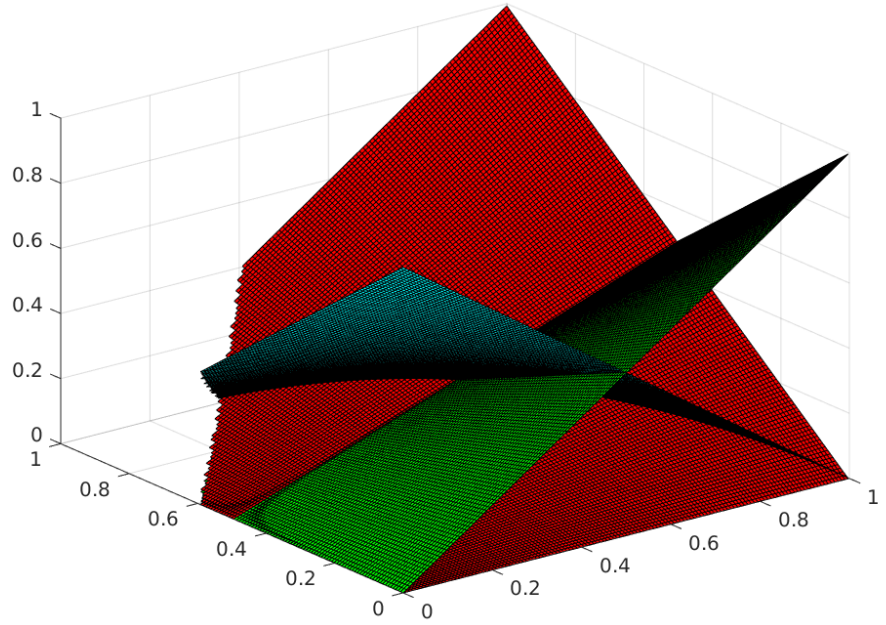
- Shape function gradients can then be computed as follows:

$$\nabla\widetilde{N}_i = \frac{\left(\sum_j N_j^*\right)\nabla N_i^* - N_i^*\left(\sum_j \nabla N_j^*\right)}{\left(\sum_j N_j^*\right)^2}$$

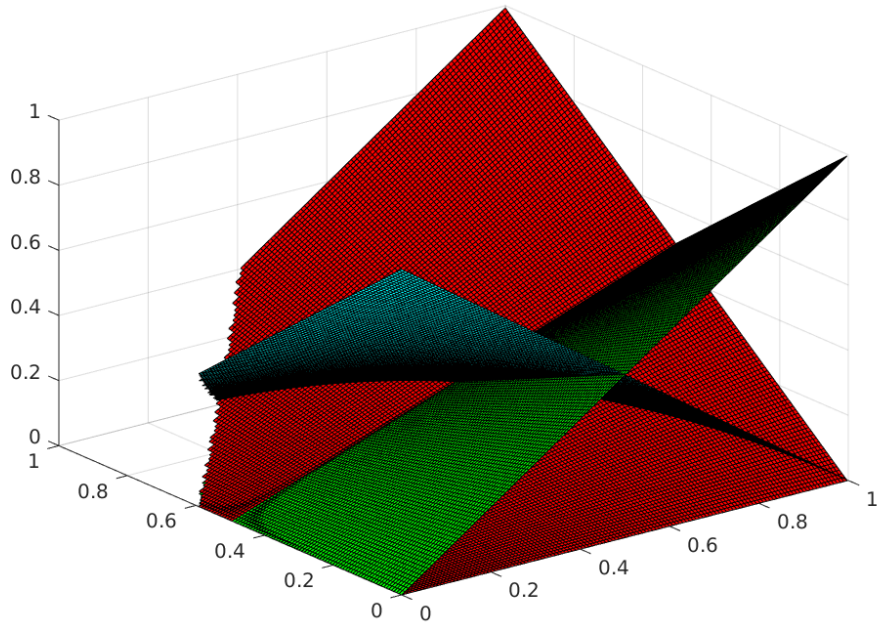
$$\sum_i \left(\nabla\widetilde{N}_i\right) = \nabla\left(\sum_i \widetilde{N}_i\right) = \nabla 1 = \mathbf{0}$$

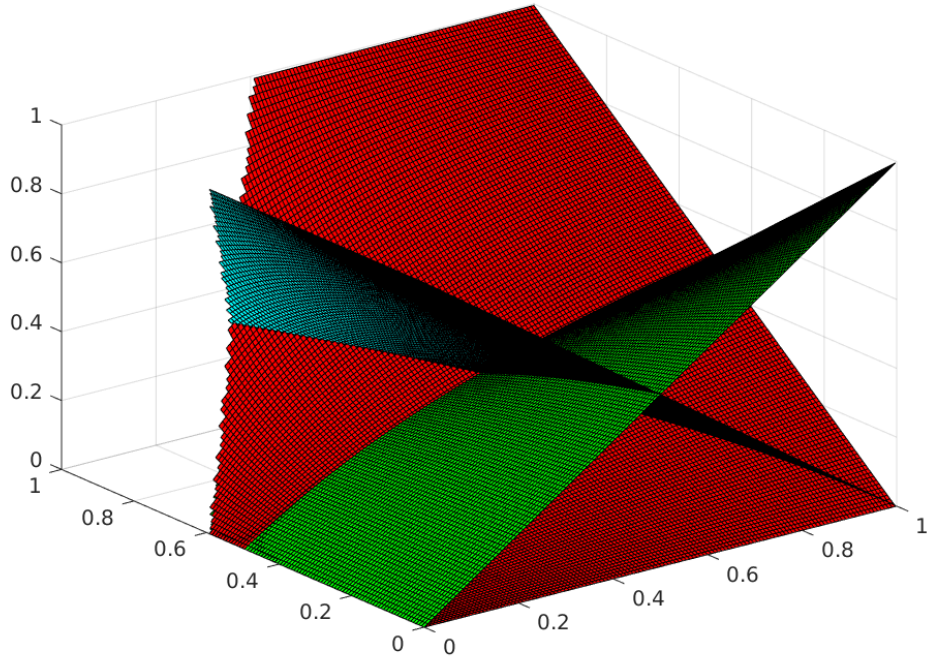
Shape functions



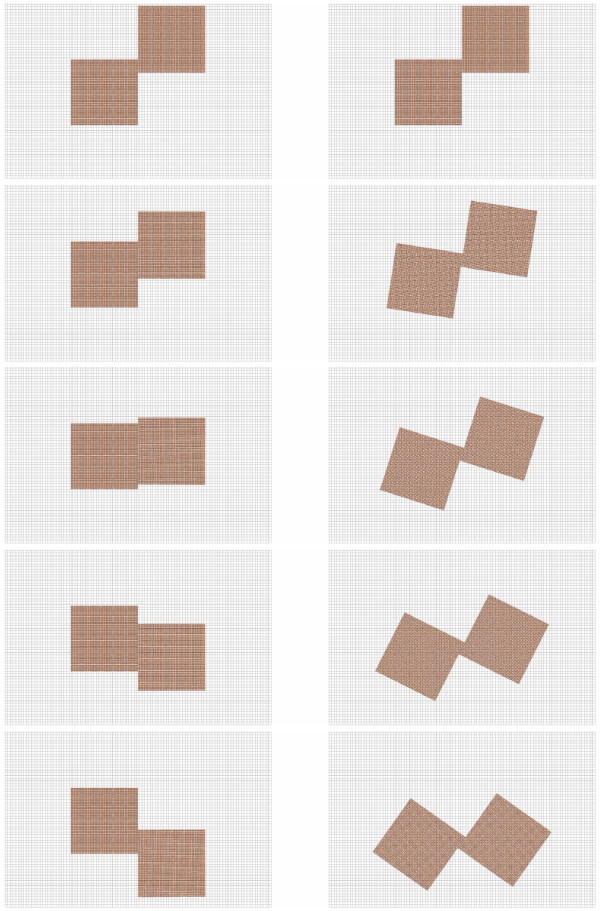


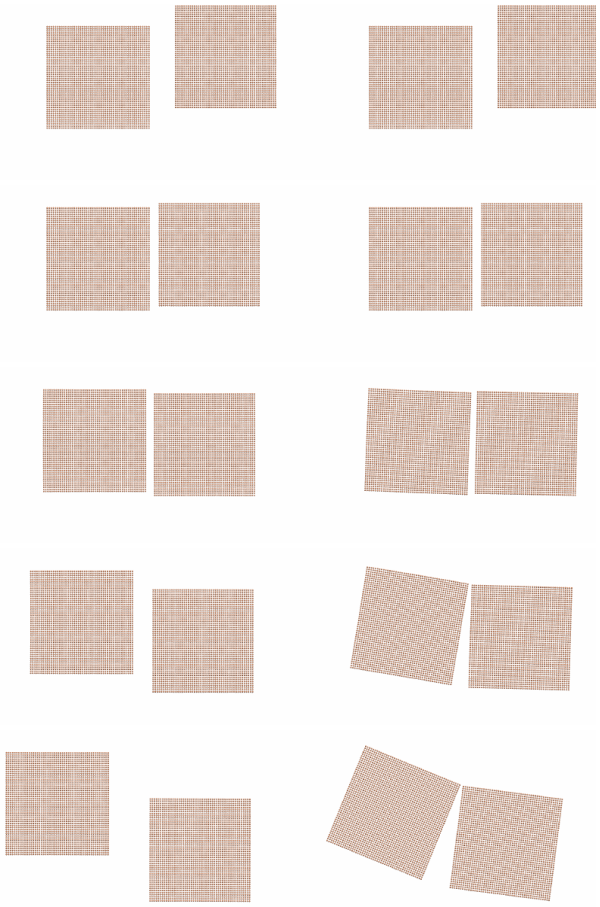
Shape functions

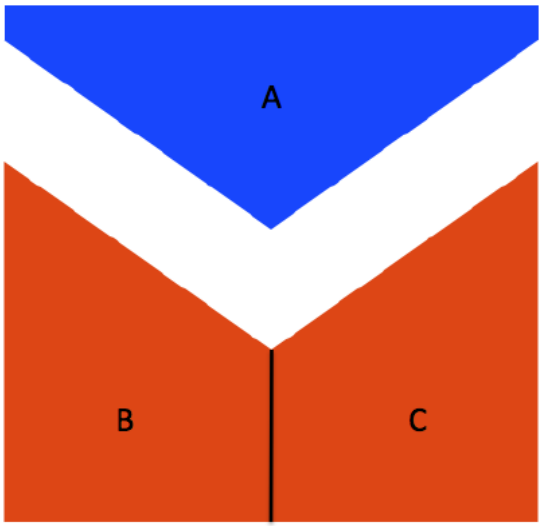


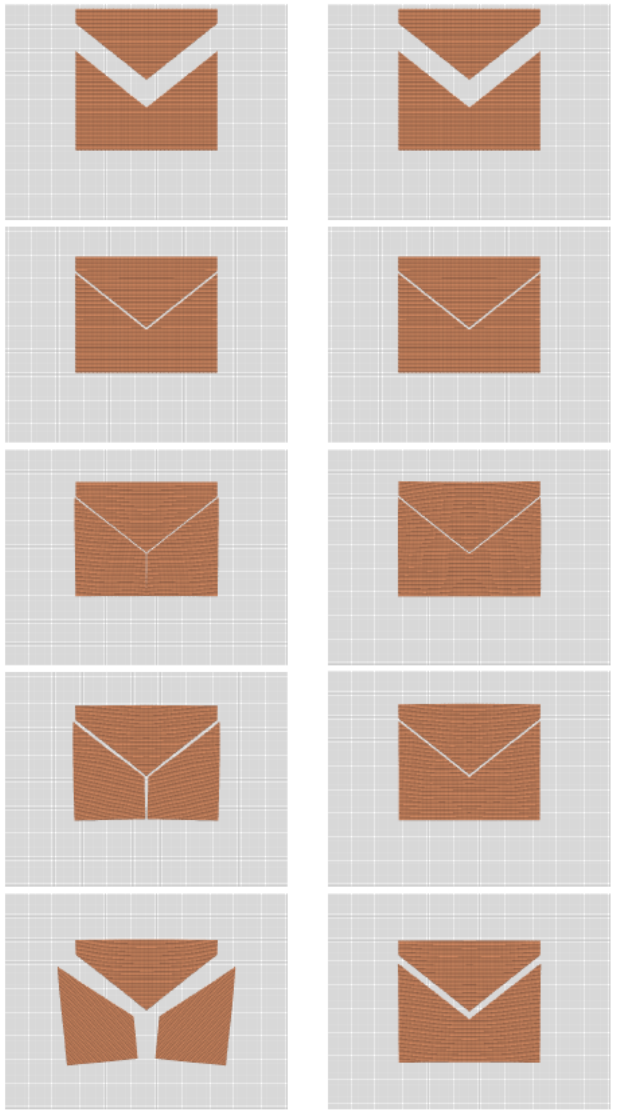


Examples

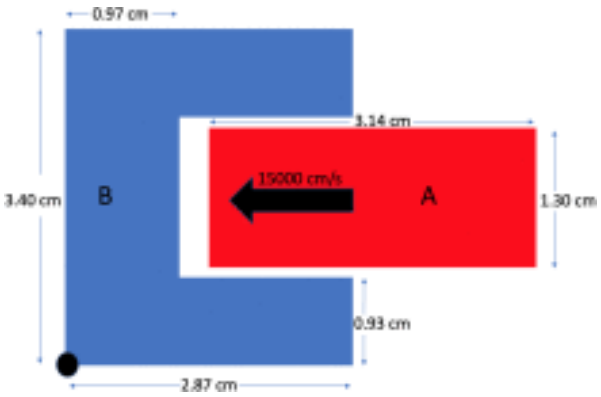


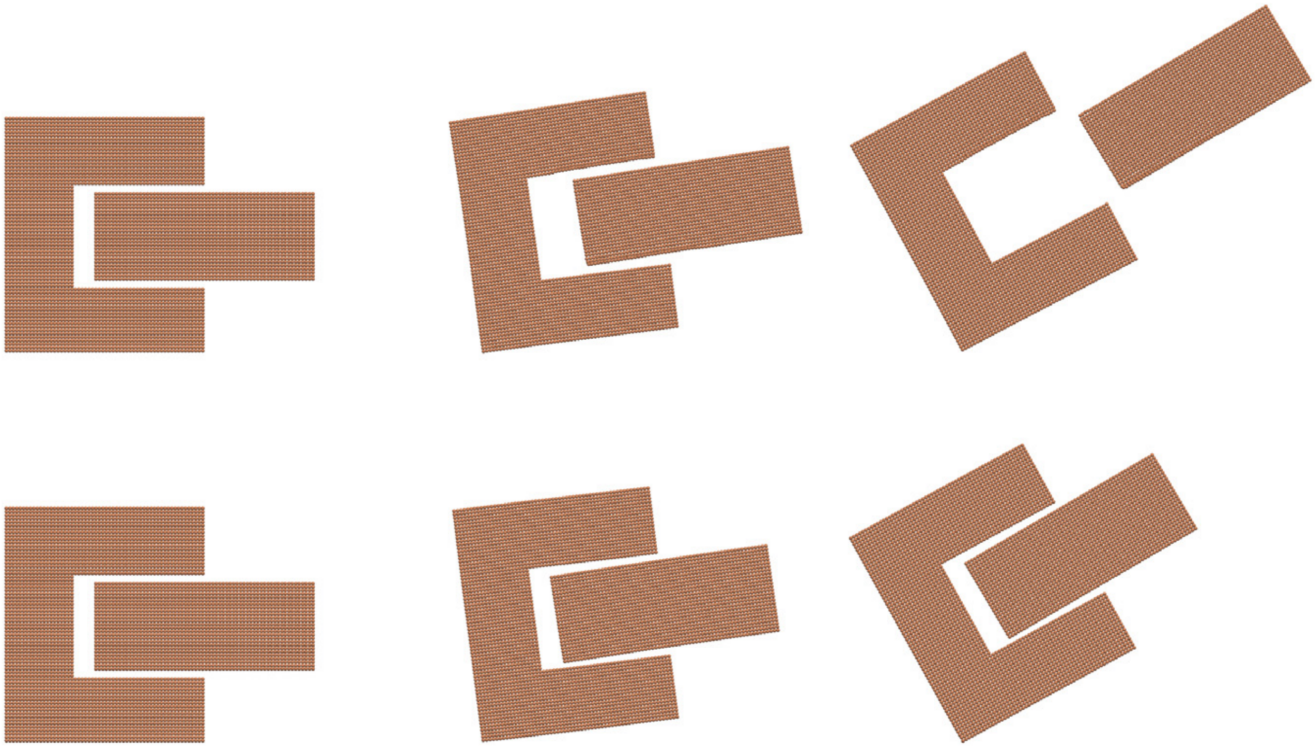



A
B
C

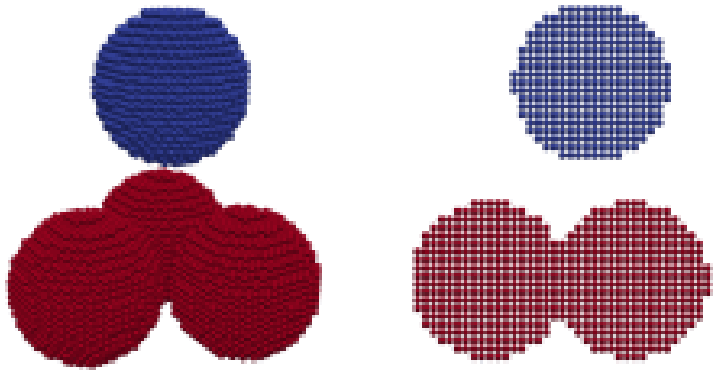


Rotating example





3D example



Conclusions/Ongoing work

- This works well for pre-prescribed cracks and appears to work for multiple bodies interacting
- We are currently working on our ability to dynamically grow a crack in a material using this method
- This feature is currently only available for an MPM approach, and we are working on the DDMP extension
- Has advantage of being able to simulate the intersection of several cracks without having to use mutliple velociy fields.