

GPU Acceleration of the Material Point Method

Fabian Meyer

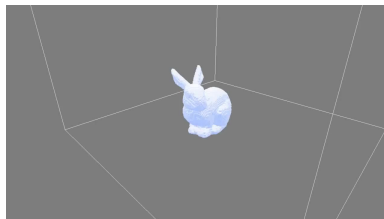
University of Koblenz

Kolloquium Computergrafik, 14 February 2019

A Brief MPM Overview: Do You Want to Build a Snowman?

A short historical summary of MPM:

- ▶ Belongs to family of particle-in-cell(PIC) techniques [EHB57].
- ▶ Initial application to solids [SZS95] → MPM
- ▶ From research to production in *Disney's* animation film *Frozen* [Sto+13].
- ▶ Avalanche research [Gau+18]



Video result of my bachelor thesis on the simulation of snow [Mey15].

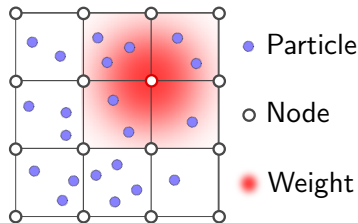
PIC ideas:

- ▶ Combine Lagrangian particles & Eulerian grid
- ▶ Particles store all information

Typical PIC/MPM roundtrip:

1. Particle-to-grid(P2G) transfer to an unmoving grid
2. Solve discretized governing equations on grid
3. Grid-to-particle(G2P) transfer back to particles & move them

⇒ **meshfree, non-empirical**



Transfers: Interpolation functions are defined over grid nodes.

GPGPU for performance enthusiasts

Why would('nt) you?

Drawbacks:

- ▶ Interactivity much easier on CPU, but slow PCI-Bus communication
- ▶ Code is mostly written against GPU architecture
- ▶ A lot of strain on the programmer

Benefits:

- ▶ Data is already on the GPU for rendering
- ▶ **Higher parallelization acceleration**

Governing Equations: Conservation of Mass & Momentum

Conservation of mass, continuum assumption holds.

Lagrangian (moving with a particle ${}_0\mathbf{x}$):

$${}_0J\rho({}_0\mathbf{x}, t) = \rho({}_0\mathbf{x}, 0). \quad (1)$$

Eulerian (outside observer ${}_t\mathbf{x}$):

$$\frac{\partial}{\partial t}\rho({}_t\mathbf{x}, t) = -\vec{\nabla} \cdot (\rho({}_t\mathbf{x}, t)\mathbf{v}({}_t\mathbf{x}, t)). \quad (2)$$

Lagrangian and Eulerian view measure differently but give same results. Equations are given in the strong form! [\[Jia+16\]](#)[\[Abe12\]](#)

Conservation of momentum:

Lagrangian (moving with a particle ${}_0\mathbf{x}$):

$$\rho({}_0\mathbf{x}, 0) \mathbf{a}({}_0\mathbf{x}, t) = \vec{\nabla} \cdot \mathbf{P}({}_0\mathbf{x}, t) + \mathbf{f}^{\text{body}}({}_0\mathbf{x}, t)_0 J. \quad (3)$$

Eulerian (outside observer ${}_t\mathbf{x}$):

$$\rho({}_t\mathbf{x}, t) \mathbf{a}({}_t\mathbf{x}, t) = \vec{\nabla} \cdot \boldsymbol{\sigma}({}_t\mathbf{x}, t) + \mathbf{f}^{\text{body}}({}_t\mathbf{x}, t) \quad (4)$$

Solving this equation will tell us how the velocity fields $\mathbf{v}({}_t\mathbf{x})$, $\mathbf{v}({}_0\mathbf{x})$ change on the whole domain due to acceleration \mathbf{a} . This is important to advect particles accounting for all forces.

[Jia+16][Abe12]

The Pretty Strong but Mathematically Weak Formulation

Weak Formulation (or Principle of Virtual Work):

Dot product equations with arbitrarily 'test functions' \mathbf{q} and apply divergence theorem:

$$\int_{\Omega^0} {}_0\mathbf{q} \cdot \left[({}_0\rho_0)({}_0\mathbf{a}) - {}_0\mathbf{f}^{\text{body}t} {}_0J \right] d{}_0\mathbf{x} = \int_{\partial\Omega^{t^n}} {}_t\mathbf{q} \cdot \boldsymbol{\sigma} d_t\mathbf{A} - \int_{\Omega^{t^n}} \nabla_t \mathbf{q} : \boldsymbol{\sigma} d_t\mathbf{x}. \quad (5)$$

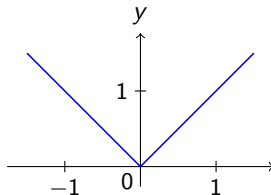
A strong solution is also a solution to the weak formulation. Leave out body forces (like gravity) and boundary condition (e.g. collisions) for now:

$$\int_{\Omega^0} {}_0\mathbf{q} \cdot ({}_0\rho_0)({}_0\mathbf{a}) d{}_0\mathbf{x} = \int_{\Omega^{t^n}} \nabla_t \mathbf{q} : \boldsymbol{\sigma} d_t\mathbf{x}. \quad (6)$$

Weak Derivative:

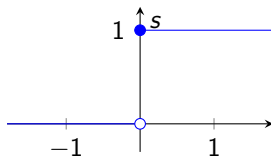
$y = |t|$ has weak derivative:

$$v = \begin{cases} -1, & \text{if } t < 0 \\ c, & \text{if } t = 0 \\ 1, & \text{if } t > 0 \end{cases}$$



Heaviside step function has no weak derivate:

$$s = \begin{cases} 0, & \text{if } t < 0 \\ 1, & \text{if } t \geq 0 \end{cases}$$



Allows for point loads, material discontinuities and more. [\[Bat06\]](#)

Discretization of Space and Time

Time discretization with implicit midpoint scheme:

$$\frac{y^{n+1} - y^n}{\Delta t} = f^{n+\frac{1}{2}} = f\left(t^n + \frac{\Delta t}{2}, \frac{1}{2}y^n + \frac{1}{2}y^{n+1}\right) \quad (7)$$

- **implicit** requires linear system solve \Rightarrow more stable, larger time steps
- **midpoint** as it conserves gov. equations

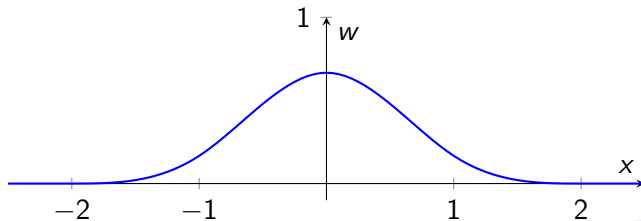
$$\Rightarrow \int_{\Omega^t} {}_t\mathbf{q} \cdot {}_t\rho({}_t\mathbf{v}^{n+1} - {}_t\mathbf{v}^n) d_t\mathbf{x} = \int_{\Omega^{t^n}} \nabla {}_t\mathbf{q} : \boldsymbol{\sigma}^{n+\frac{1}{2}} d_t\mathbf{x}. \quad (8)$$

Space Discretization is done in a Galerkin/FEM fashion with grid based interpolants w_i with limited support. Here dyadic products

$$w_i(\mathbf{x}) = w(\mathbf{x} - \mathbf{x}_i) = w\left(\frac{1}{h}(\mathbf{x} - \mathbf{x}_i)\right) = w\left(\frac{1}{h}(x - x_i)\right)w\left(\frac{1}{h}(y - y_i)\right)w\left(\frac{1}{h}(z - z_i)\right) \quad (9)$$

of cubic b-splines suffice:

$$w(x) = \begin{cases} \frac{1}{2}|x|^3 - |x|^2 + \frac{2}{3} & 0 \leq |x| < 1 \\ \frac{1}{6}(2 - |x|)^3 & 1 \leq |x| < 2 \\ 0 & 2 \leq |x| \end{cases} \quad (10)$$



Thus set in:

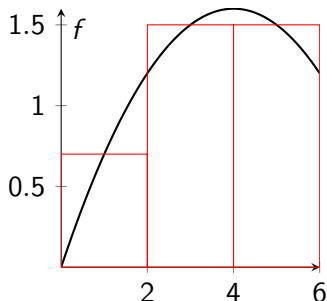
$${}_t\mathbf{q}(\mathbf{x}, t^n) = \sum_i \mathbf{e}_i w_i(\mathbf{x}), {}_t\mathbf{v}^{n(+1)}(\mathbf{x}) = \sum_j \mathbf{v}_j^{n(+1)} w_j(\mathbf{x}).$$

Combine it with numerical integration where the particles function as quadrature points [\[SKB08\]](#):

$$\begin{aligned} g_i &= \int_{\Omega} g(\mathbf{x}) w_i(\mathbf{x}) d\mathbf{x} \\ &\approx \sum_p g_p w_i(\mathbf{x}_p) V_p. \end{aligned} \quad (11)$$

due to integration by midpoint rule:

$$\int_{\Omega} f(\mathbf{x}) d\mathbf{x} \approx \sum_{i=1}^N f(\mathbf{x}_i) h_i. \quad (12)$$



Velocity Fields: APIC-Transfers

PIC-transfer

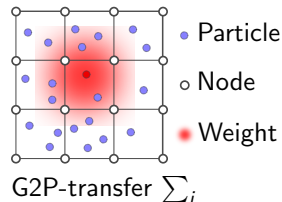
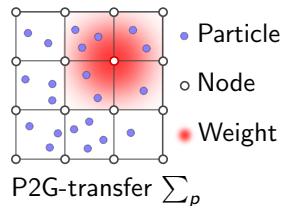
$$1. (m\mathbf{v})_i^n = \sum_p w_{ip}^n m_p \mathbf{v}_p^n$$

$$2. \mathbf{v}_i^n = \frac{(m\mathbf{v})_i^n}{m_i^n}$$

$$3. \mathbf{v}_{p,PIC}^{n+1} = \sum_i w_{ip}^n \mathbf{v}_i^{n+1}$$

APIC-transfers add a local velocity field \mathbf{C}_p^n around \mathbf{v}_p^n :

$$(m\mathbf{v})_i^n = \sum_p w_{ip}^n m_p (\mathbf{v}_p^n + \mathbf{C}_p^n (\mathbf{x}_i^n - \mathbf{x}_p^n))$$



Layout of the data: SoA vs. AoS

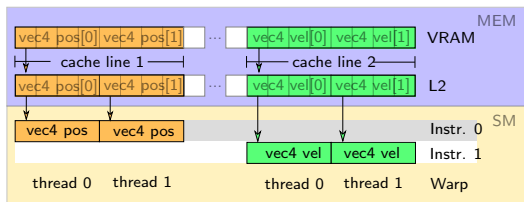
Coalescing:

SoA-Layout

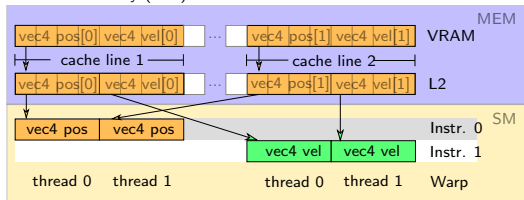
```
struct Container{
    vec4 positions[n];
    vec4 velocities[n];
} Particles;
```

AoS-Layout

```
struct Particle {
    vec4 position;
    vec4 velocity;
} Particles[n];
```



Structures of arrays (SoA)



Arrays of structures (AoS)

Nvidia Nsight[NVI] now offers metrics to identify bottlenecks:

| Metric | Description |
|-----------------|--|
| VRAM SOL% | memory throughput w.r.t. to hardware limit |
| SM SOL% | instruction throughput |
| L2 SOL% | L2-cache throughput |
| Tex SOL% | L1-cache throughput |
| SM Issue Util.% | amount of cycles an instr. was issued |

A simple map or copy shader with one instruction shows SoA vs. AoS differences:

| Layout | $\Delta t_c (\mu s)$ | Speedup | VRAM | SM | L2 | SM Issue Util. |
|--------|----------------------|---------|-------|--------------|-------|----------------|
| AoS | 243 | - | 77.7% | 7.3% | 30.3% | 6.8% |
| SoA | 120 | 2.26x | 75.4% | 14.3% | 29.4% | 14.0% |

⇒ SoA increases coalescing for non-random access.

Parallel Reduction & Scan

hi

1. [Jia+16]



Rohan Abeyaratne. *Volume II of Lecture Notes on, The Mechanics of Elastic Solids: Continuum Mechanics.*

http:

[//web.mit.edu/abeyaratne/Volumes/RCA_Vol_II.pdf](http://web.mit.edu/abeyaratne/Volumes/RCA_Vol_II.pdf).

[Online; accessed 08-November-2018]. MIT
Department of Mechanical Engineering, 2012.



Klaus-Jürgen Bathe. *Finite element procedures.*
Klaus-Jurgen Bathe, 2006.



Martha W Evans, Francis H Harlow, and
Eleazer Bromberg. *The particle-in-cell method for
hydrodynamic calculations.* Tech. rep. LOS ALAMOS
NATIONAL LAB NM, 1957.



Johan Gaume et al. “Unified modeling of the release
and flow of snow avalanches using the Material Point
Method”. In: Aug. 2018.



Chenfanfu Jiang et al. “The material point method for simulating continuum materials”. In: *ACM SIGGRAPH 2016 Courses*. ACM. 2016, p. 24.



Fabian Meyer. “Simulation von Schnee”. Bachelor’s Thesis. Universität Koblenz-Landau, Institut für Computervisualistik, 2015.



NVIDIA Corporation. *NVIDIA Nsight*.
https://docs.nvidia.com/nsight-visual-studio-edition/Content/Performance_Markers_OGL.htm.
[Online; accessed 5-December-2018].



Michael Steffen, Robert M Kirby, and Martin Berzins. “Analysis and reduction of quadrature errors in the material point method (MPM)”. In: *International journal for numerical methods in engineering* 76.6 (2008), pp. 922–948.



Alexey Stomakhin et al. “A material point method for snow simulation”. In: *ACM Transactions on Graphics (TOG)* 32.4 (2013), p. 102.



Deborah Sulsky, Shi-Jian Zhou, and Howard L Schreyer. “Application of a particle-in-cell method to solid mechanics”. In: *Computer physics communications* 87.1-2 (1995), pp. 236–252.