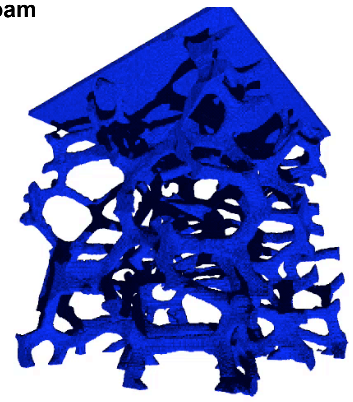
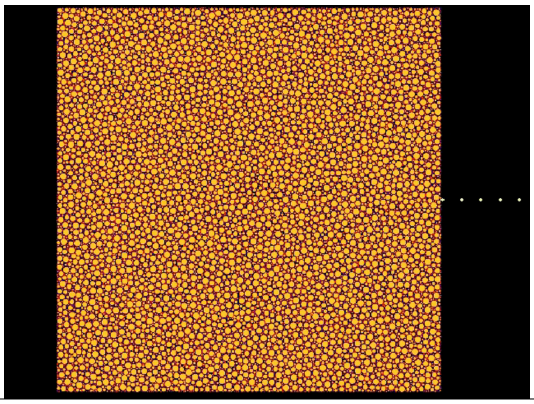
The Generalized Interpolation Material Point Method

Compaction of a foam microstructure



Tungsten Particle Impacting sandstone



The Material Point Method (MPM)

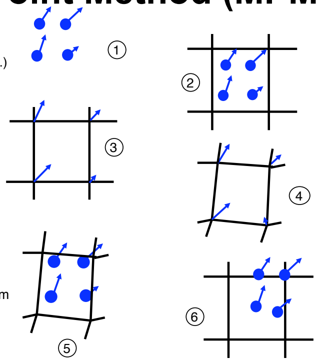
1. Lagrangian material points carry all state data (position, velocity, stress, etc.)
2. Overlying mesh defined
3. Particle state projected to mesh, e.g.:

$$v_g = \sum_p S_{gp} m_p v_p \Big/ \sum_p S_{gp} m_p$$

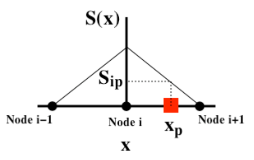
4. Conservation of momentum solved on mesh giving updated mesh velocity and (in principal) position.

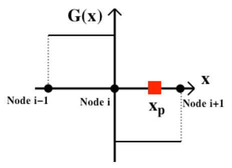
   Stress at particles computed based on gradient of the mesh velocity.

5. Particle positions/velocities updated from mesh solution.
6. Discard deformed mesh. Define new mesh and repeat



Interpolation function - MPM

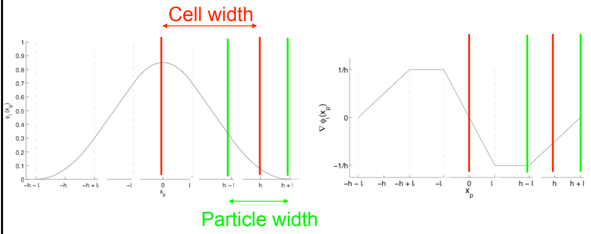




$$S_i(x_p) = \begin{cases} 1 + \frac{x_p - x_i}{h} & -h < x_p - x_i \le 0 \\ 1 - \frac{x_p - x_i}{h} & 0 < x_p - x_i \le h \\ 0 & \text{otherwise} \end{cases}$$

$$G_i(x_p) = \begin{cases} \frac{1}{h} & -h < x_p - x_i \le 0 \\ -\frac{1}{h} & 0 < x_p - x_i \le h \\ 0 & \text{otherwise} \end{cases}$$

Interpolation function - GIMP



$$S_i(x_p) = \begin{cases} \frac{(h+l_p+(x_p-x_i))^2}{4hl_p} & -h - l_p < x_p - x_i \le -h + l_p \\ 1 + \frac{(x_p-x_i)}{h} & -h + l_p < x_p - x_i \le -l_p \\ 1 - \frac{(x_p-x_i)^2+l_p^2}{2hl_p} & -l_p < x_p - x_i \le l_p \\ 1 - \frac{(x_p-x_i)}{h} & l_p < x_p - x_i \le h - l_p \\ \frac{(h+l_p-(x_p-x_i))^2}{4hl_p} & h - l_p < x_p - x_i \le h + l_p \\ 0 & \text{otherwise,} \end{cases}$$

$$\nabla S_i(x_p) = \begin{cases} \frac{h+l_p+(x_p-x_i)}{2hl_p} & -h - l_p < x_p - x_i \le -h + l_p \\ \frac{1}{h} & -h + l_p < x_p - x_i \le -l_p \\ -\frac{(x_p-x_i)}{hl_p} & -l_p < x_p - x_i \le l_p \\ -\frac{1}{h} & l_p < x_p - x_i \le h - l_p \\ -\frac{h+l_p-(x_p-x_i)}{2hl_p} & h - l_p < x_p - x_i \le h + l_p \\ 0 & \text{otherwise,} \end{cases}$$

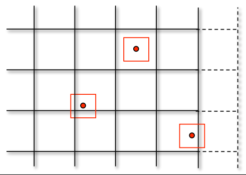
Interpolation function

In 2D, shape functions are formed by the products of the x and y function: $S_{ij}(x,y) = S_i(x_p)S_j(y_p)$

Gradients involve PARTIAL derivatives, so for instance:

$$\nabla S_{ip}(x,y) = \frac{dS_i(x_p)}{dx} S_i(y_p)\mathbf{i} + S_i(x_p)\frac{dS_i(y_p)}{dy}\mathbf{j}$$

$$\nabla S_{ip}(x,y) = Gx_{ip}(x,y)\mathbf{i} + Gy_{ip}(x,y)\mathbf{j}$$



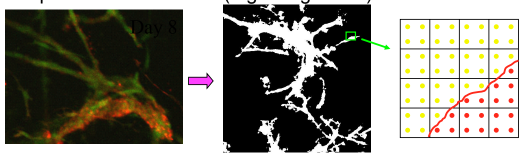
Steps in an MPM code

1. Initialize particles and create (logically) a grid
2. Project (integrate) particle data to grid (mass, velocity, etc.)
3. Set boundary conditions on velocity
4. Compute internal force from divergence of stress
5. Compute acceleration on the grid (a=F/m)
6. Integrate velocity on the grid (v* = v + a*dt)
7. Set boundary conditions on v* and a.
8. Compute Stress, update volume, compute dt_new
9. Update particle position and velocity, t = t+ dt, dt = dt_new
10. “Reset” grid and return to step 2 and repeat while t<t_final

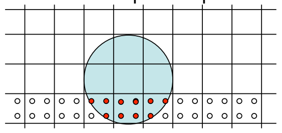
Step 1: Particle Initialization

There are two basic methods for determining particle locations

1. Acquire them from a file (e.g. image data)



2. Use geometric primitives to describe geometry, and inside/ outside tests to determine particle placement.



Step 1: Particle Initialization

At t=0, the particles need to be initialized with the following data

1. Position, x_p
2. Volume, v_p
3. Mass, (m_p=density*volume)
4. Velocity, V_p
5. Stress, $\sigma_p = 0$
6. Deformation Gradient (F_p =Identity)
7. Size $Lx_{p,}$ Ly_p (dx/PPC)

$$\sigma_p = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & 0 \\ \sigma_{yx} & \sigma_{yy} & 0 \\ 0 & 0 & \sigma_{zz} \end{bmatrix}$$

Step 2: Particle Data to Grid

Compute mass and velocity on the grid, m_i and v_i

$$m_i = \sum_p S_{ip} m_p + 1.0x10^{-100}$$

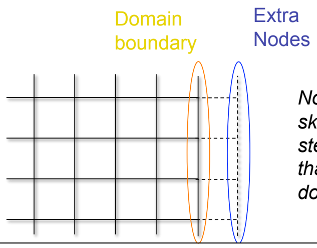
$$\mathbf{V}_i = \frac{\sum_p S_{ip} m_p \mathbf{V}_p}{m_i}$$

A quick check on your work:

$$\sum_i m_i = \sum_p m_p \qquad \sum_i m_i \mathbf{V}_i = \sum_p m_p \mathbf{V}_p$$

Step 3: Boundary Conditions

For simplicity, assume that the computational domain is a rigid box. If the velocity of the rigid walls is zero, then set the velocity on those computational nodes to be zero. Also need to set the velocity on the "extra" nodes to be zero as well.



Note, for starters, you can skip this (and the other BC step) if you solve problems that stay away from the domain boundaries.

Step 4: Compute Internal Force

The internal force is the volume integral of the divergence of the stress (stress is a second order tensor). The volume integral is approximated by summing the particle volumes. The divergence operation uses the gradients of the shape functions to give:

$$\mathbf{f}_i^{\text{int}} = -\sum_p \left(\nabla S_{ip} \cdot \sigma_p\right) v_p$$

Or, a bit more explicitly:

$$\left(\mathbf{f}_i^{\text{int}}\right)_x = -\sum_p \left(Gx_{ip}(x,y)\sigma_{xx} + Gy_{ip}(x,y)\sigma_{yx}\right) v_p$$
$$\left(\mathbf{f}_i^{\text{int}}\right)_x = -\sum_p \left(Gx_{ip}(x,y)\sigma_{xy} + Gy_{ip}(x,y)\sigma_{yy}\right) v_p$$

Step 5: Compute Acceleration

This is basically just inverting Newton's Second Law to get acceleration at each grid node:

$$\mathbf{a}_i = \frac{\left(\mathbf{f}_i^{\text{ext}} + \mathbf{f}_i^{\text{int}}\right)}{m_i}$$

This is also a convenient place to include gravity:

$$\mathbf{a}_i = \frac{\left(\mathbf{f}_i^{\text{ext}} + \mathbf{f}_i^{\text{int}}\right)}{m_i} + \mathbf{g}$$

Step 6: Integrate nodal Velocity

Using basic forward Euler integration, advance the velocity at the grid nodes:

$$\mathbf{V}_i^* = \mathbf{V}_i + \mathbf{a}_i \bullet \Delta t$$

Step 7: Boundary Conditions

As in Step 3, set all components of **V**$_i$***** and **a**$_i$ to zero, on both the domain boundary nodes and the "extra" nodes.

Step 8: Compute Particle Stress

The first part of this step is presented in a general manner. Namely, computing the kinematic behavior at the particle level. Then a specific "constitutive model" is given for getting an elastic stress from the deformation gradient, F.

First, compute the velocity gradient at each particle, based on nodal velocities:

$$\nabla \mathbf{V}_p = \sum_i \nabla S_{ip} \mathbf{V}_i^*$$

Note that this is creating a second order tensor from two vectors (first order tensors) via a dyadic product.

Step 8: Compute Particle Stress

Next, we'll use the velocity gradient tensor to update the deformation gradient **F**:

$$\mathbf{F}^{\mathbf{t}+\Delta t} = (\mathbf{I} + \nabla \mathbf{V}_p \cdot \Delta t)\mathbf{F}^{\mathbf{t}}$$

With **F** in hand, one can use any number of constitutive models. A simple one is given below:

$$\sigma = \lambda \frac{\ln(J)}{J}\mathbf{I} + \frac{\mu}{J}\left(\mathbf{F}\mathbf{F}^{\mathbf{T}} - \mathbf{I}\right)$$

Where J=det(**F**) and μ and λ and material specific properties.

Step 8: Compute Particle Stress

Finally, update the particle volume according to:

$$v_p^{\,t+\Delta t} = \det(\mathbf{I} + \nabla \mathbf{V}_p \cdot \Delta t) v_p^{\,t}$$

And compute a new timestep size that will satisfy the CFL stability condition:

$$dt_{new} \le CFL \frac{\left(\mathbf{V}_p + C\right)}{dx},$$

$$\text{where } CFL \text{ should be .5 or smaller,}$$

$$C = \frac{Ev_p}{m_p},$$

$$\text{and } E = \frac{\mu\left(3\lambda + 2\mu\right)}{\lambda + \mu}$$

Step 9: Update Particle State

Update particle velocity according to:

$$\mathbf{V}_p = \mathbf{V}_p + \sum_i S_{ip}\mathbf{a}_i \cdot \Delta t$$

$$\mathbf{X}_p = \mathbf{X}_p + \sum_i S_{ip}\mathbf{V}_i^* \cdot \Delta t$$

Lastly, update the time:

$$time = time + \Delta t$$

$$\Delta t = \Delta t_{new}$$

Step 10: Return to Step 2

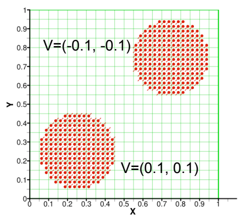
Repeat steps 2 through 9 until the time reaches the desired simulation time.

A first Simulation

Consider replicating the results from Section 7.3 of Sulsky, Chen and Schreyer, 1995. There, two cylinder of diameter (approximately) 0.5 are given initial velocities towards each other, and they collide and bounce away.

Properties given are:
Density = 1000
E = 1000
Poisson's ratio = 0.3
Velocity = (0.1, 0.1)
and (-0.1, -0.1)

These values of E and Poisson's ratio correspond to:
λ = 577
μ = 385



A first Simulation

Energy plots such as that shown in Figure 5a can be obtained by summing up the kinetic energy of the particles:

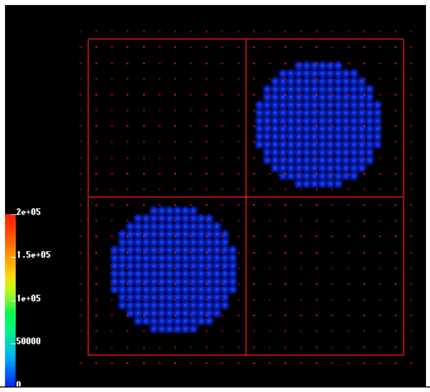
$$KE = 0.5 * m_p * v_p^2$$

The strain energy is a little more complicated. It is easiest to compute during the stress calculation, and is given by:

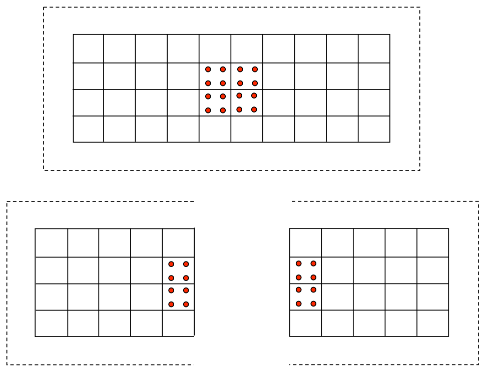
$$SE = 0.5*\lambda*(\ln(J))-\mu*\ln(J)+0.5*\mu*(\mathrm{trace}(F^TF-3))$$

Again, J in this equation is the determinant of F.

A first Simulation



Parallelism - Domain Decomposition



Parallelism - Domain Decomposition