# Generalized Interpolation Material Point (GIMP) Method

John Nairn

February 19, 2015

## 1 Introduction

These notes provide a detailed derivation of the GIMP method for MPM, which was first described by Bardenhagen and Kober. The notes pay attention to large deformation and give extensions to axisymmetry. Finally, GIMP methods for implementing transport equations are derived.

## 2 Virtual Work or Power

Let $\delta\boldsymbol{u}$ be a virtual displacement. Virtual work (or power) gives the following starting equation:

$$\int_V \rho\boldsymbol{b}\cdot\delta\boldsymbol{u}\,dV + \int_{S_T}\boldsymbol{T}\cdot\delta\boldsymbol{u}\,dS + \sum_p \boldsymbol{F_p}\cdot\delta\boldsymbol{u} = \int_V \rho\boldsymbol{a}\cdot\delta\boldsymbol{u}\,dV + \int_V \boldsymbol{\sigma}\cdot\nabla\delta\boldsymbol{u}\,dV \tag{1}$$

where $\boldsymbol{b}$ is specific body force (*e.g.*, gravity), $\boldsymbol{T}$ is traction applied on the surface $S_T$, $\boldsymbol{F_p}$ is a force applied directly to a particle (an extra turn added for MPM modeling), $\boldsymbol{a}$ is acceleration, $\rho$ is density, and $\boldsymbol{\sigma}$ is the Cauchy stress.

## 3 Particle Basis Expansion

First, we expand $\rho\boldsymbol{b}$, $\rho\boldsymbol{a}$, and $\boldsymbol{\sigma}$ in a particle basis, where any function of $\boldsymbol{x}$ is written as:

$$f(\boldsymbol{x}) = \sum_p f_p\chi_p(\boldsymbol{x}) \tag{2}$$

where $f_p$ is the value of the function on particle $p$ and $\chi_p(\boldsymbol{x})$ is a particle basis function. The most common function is for $\chi_p(\boldsymbol{x}) = 1$ inside the particle domain and 0 outside the domain. The property expansions are:

$$\rho\boldsymbol{b} = \sum_p \rho_p\boldsymbol{b_p}\chi_p(\boldsymbol{x}) = \sum_p \frac{m_p}{V_p}\boldsymbol{b_p}\chi_p(\boldsymbol{x}) \tag{3}$$

$$\rho\boldsymbol{a} = \sum_p \frac{m_p}{V_p}\frac{d\boldsymbol{v_p}}{dt}\chi_p(\boldsymbol{x}) = \sum_p \frac{\dot{\boldsymbol{p}}_{\boldsymbol{p}}}{V_p}\chi_p(\boldsymbol{x}) \tag{4}$$

$$\boldsymbol{\sigma} = \sum_p \boldsymbol{\sigma}_p\chi_p(\boldsymbol{x}) \tag{5}$$

where $m_p$ is particle mass, $V_p$ is current particle volume, $\boldsymbol{v_p}$ is particle velocity, $\boldsymbol{p_p}$ is particle momentum, and $\boldsymbol{\sigma}_p$ is particle stress. The particle force term (a useful MPM term), can be expanded as

$$\boldsymbol{F}_p = \frac{\boldsymbol{F}_p}{V_p}\int_V \chi_p(\boldsymbol{x})\,dV \tag{6}$$

This results is making used of normalized particle shape function such that:

$$V_p = \int_V \chi_p(\boldsymbol{x})\,dV \tag{7}$$

The new virtual work equation becomes:

$$\sum_p \int_V \frac{m_p}{V_p}\chi_p(\boldsymbol{x})\boldsymbol{b_p}\cdot\delta\boldsymbol{u}\,dV + \int_{S_T}\boldsymbol{T}\cdot\delta\boldsymbol{u}\,dS + \sum_p \frac{\boldsymbol{F_p}}{\boldsymbol{V_p}}\int_V \chi_p(\boldsymbol{x})\cdot\delta\boldsymbol{u}\,dV \tag{8}$$

$$= \sum_p \int_V \frac{\dot{\boldsymbol{p}}_{\boldsymbol{p}}}{V_p}\chi_p(\boldsymbol{x})\cdot\delta\boldsymbol{u}\,dV + \sum_p \int_V \boldsymbol{\sigma}_p\chi_p(\boldsymbol{x})\cdot\nabla\delta\boldsymbol{u}\,dV \tag{9}$$

## 4 Grid Expansion

Next, we expand the virtual displacements in grid-based shape functions using:

$$\delta\boldsymbol{u} = \sum_i \delta\boldsymbol{u_i}N_i(\boldsymbol{x}) \tag{10}$$

where the sum is over nodes ($i$), $\delta\boldsymbol{u_i}$ is the virtual displacement on node $i$, and $N_i(\boldsymbol{x})$ are the nodal shape functions (standard finite element shape functions). Each term is revised as follows:

$$\sum_i\sum_p\int_V \frac{m_p}{V_p}\chi_p(\boldsymbol{x})\boldsymbol{b_p}\cdot\delta\boldsymbol{u_i}N_i(\boldsymbol{x})\,dV = \sum_i\sum_p S_{ip}m_p\boldsymbol{b}_p\cdot\delta\boldsymbol{u_i} \tag{11}$$

$$\int_{S_T}\boldsymbol{T}\cdot\delta\boldsymbol{u} = \sum_i \delta\boldsymbol{u_i}\cdot\int_{S_T}\boldsymbol{N}_i(\boldsymbol{x})\boldsymbol{T}\,dS \tag{12}$$

$$\sum_i\sum_p\frac{\boldsymbol{F_p}}{\boldsymbol{V_p}}\int_V \chi_p(\boldsymbol{x})\cdot\delta\boldsymbol{u_i}N_i(\boldsymbol{x})\,dV = \sum_i\sum_p S_{ip}\boldsymbol{F}_p\cdot\delta\boldsymbol{u_i} \tag{13}$$

$$\sum_i\sum_p\int_V \frac{\dot{\boldsymbol{p}}_{\boldsymbol{p}}}{V_p}\chi_p(\boldsymbol{x})\cdot\delta\boldsymbol{u_i}N_i(\boldsymbol{x})\,dV = \sum_i\sum_p S_{ip}\dot{\boldsymbol{p}}_{\boldsymbol{p}}\cdot\delta\boldsymbol{u_i} \tag{14}$$

$$\sum_i\sum_p\int_V \boldsymbol{\sigma}_p\chi_p(\boldsymbol{x})\cdot\delta\boldsymbol{u_i}\nabla N_i(\boldsymbol{x})\,dV = \sum_i\sum_p (V_p\boldsymbol{\sigma}_p\boldsymbol{G}_{ip})\cdot\delta\boldsymbol{u_i} \tag{15}$$

where the GIMP shape functions are:

$$S_{ip} = \frac{1}{V_p}\int_V \chi_p(\boldsymbol{x})N_i(\boldsymbol{x})\,dV \tag{16}$$

$$\boldsymbol{G}_{ip} = \frac{1}{V_p}\int_V \chi_p(\boldsymbol{x})\nabla N_i(\boldsymbol{x})\,dV \tag{17}$$

The particle stress term can be revised to use particle mass as follows:

$$V_p\boldsymbol{\sigma}_p = m_p\frac{\boldsymbol{\sigma}_p}{\rho_p} = m_p\frac{\boldsymbol{\sigma}_p}{\rho_p}\frac{\rho_0}{\rho_0} = m_p\frac{J\sigma_p}{\rho_0} = m_p\frac{\tau}{\rho_0} \tag{18}$$

Here $J = V_p/V_0 = \rho_0/\rho_p$ is the relative volume and $\boldsymbol{\tau}$ is the Kirchhoff stress. In this form the stress term depends most naturally on Kirchhoff stresses normalized to initial density (and this approach avoids the need to find current particle density or volume when calculating that term on each time step). For this reason, `NairnMPM` always tracks stress as $\boldsymbol{\tau}/\rho_0$ (and all material models are expected to calculate this stress). When archiving stress, `NairnMPM` writes Cauchy stress using $\boldsymbol{\sigma}_p = (\rho_0/J)*(\tau/\rho_0)$. Note that when doing energy calculations that $\boldsymbol{\tau}\cdot \mathrm{du}$, where du is incremental deformation gradient, is energy per unit initial volume and therefore the energy calculated in code using $(\tau/\rho_0)\cdot \mathrm{du}$ is energy per unit mass or $dU/(\rho_0V_0)$.

## 5 MPM Equation

Making use of the fact that $\delta\boldsymbol{u_i}$ is arbitrary, the summand of all terms can be equated to arrive at the controlling MPM equation on the background grid of:

$$\sum_p S_{ip}^{(n)}\frac{d\boldsymbol{p_p^{(n)}}}{dt} = \frac{d\boldsymbol{p}_i^{(n)}}{dt} = \boldsymbol{f}_i^{(n)} + \boldsymbol{f}_{i,T}^{(n)} \tag{19}$$

where

$$\boldsymbol{f}_i^{(n)} = \sum_p\left(-m_p\frac{\boldsymbol{\tau}_p^{(n)}\cdot\boldsymbol{G}_{ip}^{(n)}}{\rho_0} + m_pS_{ip}^{(n)}\boldsymbol{b}_p + \boldsymbol{F}_p^{(n)}S_{ip}^{(n)}\right) \tag{20}$$

$$\boldsymbol{f}_{i,T}^{(n)} = \int_{S_T}\boldsymbol{N}_i(\boldsymbol{x})\boldsymbol{T}\,dS \tag{21}$$

where superscript $(n)$ has been added to mean terms calculated from the state of all particles at the start of time step $n$. Commonly the body force, $\boldsymbol{b}_p$, will be independent of the particle state (*e.g.,* gravity, which is a constant, or a general body force that depends only on position and time and not particle state). When that holds, it can be removed from the sum. The total internal force them becomes

$$\boldsymbol{f}_i^{(n)} = \sum_p\left(-m_p\frac{\boldsymbol{\tau}_p^{(n)}\cdot\boldsymbol{G}_{ip}^{(n)}}{\rho_0} + \boldsymbol{F}_p^{(n)}S_{ip}^{(n)}\right) + m_i^{(n)}\boldsymbol{b}(\boldsymbol{x}_i,t) \tag{22}$$

where $\boldsymbol{b}(\boldsymbol{x}_i,t)$ is body force at the nodal location and

$$m_i^{(n)} = \sum_p m_pS_{ip}^{(n)} \tag{23}$$

is total nodal mass.

Once the forces are found, the momentum can be updated on the grid using

$$\boldsymbol{p}_i^{(n+1)} = \boldsymbol{p}_i^{(n)} + (\boldsymbol{f}_i^{(n)} + \boldsymbol{f}_{i,T}^{(n)})\Delta t \tag{24}$$

where $\Delta t$ is the time step and:

$$\boldsymbol{p}_i^{(n)} = \sum_p\boldsymbol{p}_p^{(n)}S_{ip}^{(n)} \tag{25}$$

## 5.1 Velocity Approach to Strain and Particle Updates

### 5.1.1 Incremental Displacement Gradient

For strain updates, the algorithm needs to evaluate the spatial velocity gradient, $\nabla \boldsymbol{v}(x,t)$, and use that to find the incremental deformation gradient, $\mathsf{dF}$, defined by

$$\mathsf{dF}^{(n)} = \exp(\nabla \boldsymbol{v} \Delta t) \tag{26}$$

which assumes $\nabla \boldsymbol{v}$ is constant over the time step. The gradient is found by extrapolating spatial velocity on the grid:

$$\nabla \boldsymbol{v} = \sum_i \boldsymbol{G}_{ip} \otimes \boldsymbol{v}_i^{(n)} \tag{27}$$

and the spatial velocity is found from

$$\boldsymbol{v}_i^{(n)} = \frac{\boldsymbol{p}_i^{(n)}}{m_i^{(n)}} \tag{28}$$

### 5.1.2 Particle Updates with Damping

In each time step, the Eulerian update on the grid is used to update the Lagrangian velocities and positions on the particles, and it is useful to include damping options in both the grid and particle updates. Two global damping strategies are to add damping terms proportional to either the grid velocity or the particle velocity. In most cases, these two approaches to damping should have similar results. By including them both, however, it is possible to propose several different types of damping schemes. If damping is based on grid velocity, then the nodal momentum update changes to:

$$p_i^{*(n+1)} = p_i^{(n)} + m_i^{(n)} \boldsymbol{a}_i^{*(n)} \Delta t \tag{29}$$

where "*" indicates a term that is revised to including damping affects in this time step. The effective nodal acceleration is

$$\boldsymbol{a}_i^{*(n)} = \frac{\boldsymbol{p}_i^{*(n+1)} - \boldsymbol{p}_i^{(n)}}{m_i^{(n)} \Delta t} = \frac{\boldsymbol{f}_i^{(n)} + \boldsymbol{f}_{i,T}^{(n)} - \alpha_g(t) \boldsymbol{p}_i^{(n)}}{m_i^{(n)}} = \frac{\boldsymbol{f}_i^{(n)} + \boldsymbol{f}_{i,T}^{(n)}}{m_i^{(n)}} - \alpha_g(t) \boldsymbol{v}_i^{(n)} \tag{30}$$

where $\alpha_g(t)$ is the grid damping constant (with units 1/sec) that applies to all nodes (*i.e.*, independent of $\boldsymbol{x}_i$), but may evolve in time (such as a kinetic energy thermostat). See below for how to deal with position-dependent damping.

We can write two updates for the particles — a FLIP update that increments particle velocity using accelerations extrapolated to the grid and a PIC update that extrapolates velocity directly to the particle. After adding a particle damping constant, $\alpha_p(t)$, these two updates become:

$$\begin{aligned} \boldsymbol{v}_{p,FLIP}^{(n+1)} &= \boldsymbol{v}_p^{(n)} + \big(\boldsymbol{a}_{g\to p}^{*(n)} - \alpha_p(t) \boldsymbol{v}_p^{(n)}\big) \Delta t & (31) \\ \boldsymbol{v}_{p,PIC}^{(n+1)} &= \boldsymbol{v}_{g\to p}^{*(n+1)} - \alpha_p(t) \boldsymbol{v}_p^{(n)} \Delta t = \boldsymbol{v}_{g\to p}^{(n)} + \big(\boldsymbol{a}_{g\to p}^{*(n)} - \alpha_p(t) \boldsymbol{p}_p^{(n)}\big) \Delta t & (32) \end{aligned}$$

where $\boldsymbol{a}_{g\to p}^{*(n)}$ is effective acceleration extrapolated from the grid to the particle:

$$\boldsymbol{a}_{g\to p}^{*(n)} = \sum_i \boldsymbol{a}_i^{*(n)} S_{ip}^{(n)} = \boldsymbol{a}_{g\to p}^{(n)} - \alpha_g(t) \boldsymbol{v}_{g\to p}^{(n)} \tag{33}$$

and

$$\boldsymbol{a}^{(n)}_{g\to p} = \sum_i \frac{\boldsymbol{f}^{(n)}_i + \boldsymbol{f}^{(n)}_{i,T}}{m^{(n)}_i} S^{(n)}_{ip} \qquad \text{and} \qquad \boldsymbol{v}^{(n)}_{g\to p} = \sum_i \boldsymbol{v}^{(n)}_i S^{(n)}_{ip} \tag{34}$$

The extrapolated velocities are

$$\boldsymbol{v}^{*(n+1)}_{g\to p} = \sum_i \frac{\boldsymbol{p}^{*(n+1)}_i}{m^{(n)}_i} S^{(n)}_{ip} = \boldsymbol{v}^{(n+1)}_{g\to p} - \alpha_g(t)\boldsymbol{v}^{(n)}_{g\to p}\Delta t = \boldsymbol{v}^{(n)}_{g\to p} + \boldsymbol{a}^{*(n)}_{g\to p}\Delta t \tag{35}$$

$$\boldsymbol{v}^{(n+1)}_{g\to p} = \sum_i \left( \frac{\boldsymbol{p}^{(n)}_i}{m^{(n)}_i} + \frac{\boldsymbol{f}^{(n)}_i + \boldsymbol{f}^{(n)}_{i,T}}{m^{(n)}_i}\Delta t \right) S^{(n)}_{ip} = \boldsymbol{v}^{(n)}_{g\to p} + \boldsymbol{a}^{(n)}_{g\to p}\Delta t \tag{36}$$

Both damping strategies can be done entirely within the particle update. The two methods can be used individually or combined. Because grid and particle velocities should be similar, the total damping constant is $\alpha(t) = \alpha_g(t) + \alpha_p(t)$.

Next, the MPM position update with damping is found by integrating the extrapolated velocity over the time step, which comes from PIC velocity, by the midpoint rule:

$$\boldsymbol{x}^{(n+1)}_p = \boldsymbol{x}^{(n)}_p + \int_0^{\Delta t} \boldsymbol{v}^{(n+1)}_{p,PIC}(\Delta t = t)\,dt \approx \boldsymbol{x}^{(n)}_p + \frac{1}{2}\left(\boldsymbol{v}^{(n)}_{g\to p} + \boldsymbol{v}^{(n+1)}_{p,PIC}\right)\Delta t \tag{37}$$

$$= \boldsymbol{x}^{(n)}_p + \boldsymbol{v}^{(n)}_{g\to p}\Delta t + \frac{1}{2}\big(\boldsymbol{a}^{*(n)}_{g\to p} - \alpha_p(t)\boldsymbol{v}^{(n)}_p\big)(\Delta t)^2 \tag{38}$$

This update along with the FLIP velocity update (which is the one that should be used) can be written as:

$$\boldsymbol{x}^{(n+1)}_p = \boldsymbol{x}^{(n)}_p + \boldsymbol{v}^{*(n+1)}_{g\to p}\Delta t - \frac{1}{2}\big(\boldsymbol{a}^{*(n)}_{g\to p} + \alpha_p(t)\boldsymbol{v}^{(n)}_p\big)(\Delta t)^2 \tag{39}$$

$$\boldsymbol{v}^{(n+1)}_p = \boldsymbol{v}^{(n)}_p + \big(\boldsymbol{a}^{*(n)}_{g\to p} - \alpha_p(t)\boldsymbol{v}^{(n)}_p\big)\Delta t \tag{40}$$

It is interesting to redefine the grid and particle damping terms using:

$$\alpha_g(t) \to -\frac{1-\beta}{\Delta t} + \alpha'_g \qquad \text{and} \qquad \alpha_p(t) \to \frac{1-\beta}{\Delta t} + \alpha'_p \tag{41}$$

The time dependence of primed damping terms have been dropped, but they can all still depend on time. By unraveling all effective terms, the net updates become

$$\boldsymbol{x}^{(n+1)}_p = \boldsymbol{x}^{(n)}_p + \boldsymbol{v}^{(n+1)}_{g\to p}\Delta t - \frac{1}{2}\big(\boldsymbol{a}^{(n)}_{g\to p} + \alpha_{PIC}(\beta)\big(\boldsymbol{v}^{(n)}_p - \boldsymbol{v}^{(n)}_{g\to p}\big) + \alpha'_g\boldsymbol{v}^{(n)}_{g\to p} + \alpha'_p\boldsymbol{v}^{(n)}_p\big)(\Delta t)^2 \tag{42}$$

$$\boldsymbol{v}^{(n+1)}_p = \boldsymbol{v}^{(n)}_p + \big(\boldsymbol{a}^{(n)}_{g\to p} - \alpha_{PIC}(\beta)\big(\boldsymbol{v}^{(n)}_p - \boldsymbol{v}^{(n)}_{g\to p}\big) - \alpha'_g\boldsymbol{v}^{(n)}_{g\to p} - \alpha'_p\boldsymbol{v}^{(n)}_p\big)\Delta t \tag{43}$$

If any damping terms depend on position, however, the updates will need to be modified differently. The "PIC Damping" term is

$$\alpha_{PIC}(\beta) = \frac{1-\beta}{\Delta t} \tag{44}$$

It is called PIC damping, because in the absence of other damping terms ($\alpha'_g = \alpha'_p = 0$), the velocity update becomes

$$\boldsymbol{v}^{(n+1)}_p = \boldsymbol{v}^{(n)}_p - (1-\beta)\big(\boldsymbol{v}^{(n)}_p - \boldsymbol{v}^{(n)}_{g\to p}\big) + \boldsymbol{a}^{(n)}_{g\to p}\Delta t \tag{45}$$

If $\beta = 1$, the update is an undamped FLIP update. If $\beta = 0$, the update is an undamped, PIC update. Thus $\beta$ can be interpreted as the fraction FLIP in the velocity update. But the above analysis was formally all a FLIP analysis. A better interpretation is that $\beta$ adds an new form of artificial damping that is proportional to the inversion error between the particle velocity and the grid velocity extrapolated to the particle $(\boldsymbol{v}_p^{(n)} - \boldsymbol{v}_{g\to p}^{(n)})$. The net damping effect will be to damp out this error.

There are two approaches to coding updates with damping. First, it can be done entirely within the particle update task. The input to this task from the nodes will be $p_i^{(n+1)}$ (which will have replaced $p_i^{(n)}$ in the momentum update), $\boldsymbol{f}_i^{(n)} + \boldsymbol{f}_{i,T}^{(n)}$, and $m_i^{(n)}$. We can define two new effective terms

$$\begin{aligned}
\boldsymbol{a}_{g\to p}^{(n)'} &= \boldsymbol{a}_{g\to p}^{(n)} - \alpha_{PIC}(\beta)\big(\boldsymbol{v}_p^{(n)} - \boldsymbol{v}_{g\to p}^{(n)}\big) - \alpha_g' \boldsymbol{v}_{g\to p}^{(n)} - \alpha_p' \boldsymbol{v}_p^{(n)} && (46)\\
&= \boldsymbol{a}_{g\to p}^{(n)} - \alpha_g(t)\boldsymbol{v}_{g\to p}^{(n)} - \alpha_p(t)\boldsymbol{v}_p^{(n)} && (47)\\
\boldsymbol{v}_{g\to p}^{(n+1)'} &= \boldsymbol{v}_{g\to p}^{(n+1)} - \left(\alpha_{PIC}(\beta)\big(\boldsymbol{v}_p^{(n)} - \boldsymbol{v}_{g\to p}^{(n)}\big) + \alpha_g' \boldsymbol{v}_{g\to p}^{(n)} + \alpha_p' \boldsymbol{v}_p^{(n)}\right)\Delta t && (48)\\
&= \boldsymbol{v}_{g\to p}^{(n+1)} - \left(\alpha_g(t)\boldsymbol{v}_{g\to p}^{(n)} + \alpha_p(t)\boldsymbol{v}_p^{(n)}\right)\Delta t && (49)
\end{aligned}$$

where $\boldsymbol{v}_{g\to p}^{(n)}$ is found from available input properties using

$$\boldsymbol{v}_{g\to p}^{(n)} = \boldsymbol{v}_{g\to p}^{(n+1)} - \boldsymbol{a}_{g\to p}^{(n)}\Delta t \qquad (50)$$

The final updates become

$$\begin{aligned}
\boldsymbol{x}_p^{(n+1)} &= \boldsymbol{x}_p^{(n)} + \boldsymbol{v}_{g\to p}^{(n+1)'}\Delta t - \frac{1}{2}\boldsymbol{a}_{g\to p}^{(n)'}(\Delta t)^2 && (51)\\
\boldsymbol{v}_p^{(n+1)} &= \boldsymbol{v}_p^{(n)} + \boldsymbol{a}_{g\to p}^{(n)'}\Delta t && (52)
\end{aligned}$$

which are standard position and velocity updates to second order in position. Notice that it was important to keep the second order term on the position update in order to end up with consistent definitions for extrapolated velocity and acceleration. In object oriented code, the grid damping is added in the update particles task, but the particle damping is not added until calling the material point class. Separating them out gives

$$\begin{aligned}
\boldsymbol{x}_p^{(n+1)} &= \boldsymbol{x}_p^{(n)} + \big(\boldsymbol{v}_* - \alpha_p(t)\boldsymbol{v}_p^{(n)}\Delta t\big)\Delta t - \frac{1}{2}\big(\boldsymbol{a}_* - \alpha_p(t)\boldsymbol{v}_p^{(n)}\big)(\Delta t)^2 && (53)\\
&= \boldsymbol{x}_p^{(n)} + \Delta t\left(\boldsymbol{v}_* - \frac{\Delta t}{2}\big(\boldsymbol{a}_* + \alpha_p(t)\boldsymbol{v}_p^{(n)}\big)\right) && (54)\\
\boldsymbol{v}_p^{(n+1)} &= \boldsymbol{v}_p^{(n)} + \big(\boldsymbol{a}_* - \alpha_p(t)\boldsymbol{v}_p^{(n)}\big)\Delta t = \boldsymbol{v}_p^{(n)}\big(1 - \alpha_p(t)\Delta t\big) + \boldsymbol{a}_*\Delta t && (55)
\end{aligned}$$

where

$$\begin{aligned}
\boldsymbol{v}_* &= \boldsymbol{v}_{g\to p}^{(n+1)} - \alpha_g(t)\boldsymbol{v}_{g\to p}^{(n)}\Delta t && (56)\\
\boldsymbol{a}_* &= \boldsymbol{a}_{g\to p}^{(n)} - \alpha_g(t)\boldsymbol{v}_{g\to p}^{(n)} && (57)
\end{aligned}$$

The position update must be called first, because it needs $\boldsymbol{v}_p^{(n)}$, which is changed in the velocity update.

When all damping is done in the particle updates (as is possible from above equations), it is possible that damping will affect other sections of the code that use nodal momentum or force, such as contact, differently. To avoid these effects (if they matter), a second approach would be to apply the grid damping to the force calculation. But, when PIC damping is used, this could cause very unrealistic forces (because that term can be large when $\Delta t$ is small). A potential solution is to apply only $\alpha'_g$ to force calculations and then apply PIC damping during the particle updates. At the time of the particle update, the nodal input values would be

$$\begin{aligned} \boldsymbol{f}_i^{**(n)} &= \boldsymbol{f}_i^{(n)} + \boldsymbol{f}_{i,T}^{(n)} - \alpha'_g \boldsymbol{p}_i^{(n)} && (58) \\ \boldsymbol{p}_i^{**(n+1)} &= p_i^{(n)} + m_i^{(n)} \boldsymbol{a}_i^{**(n)} \Delta t && (59) \\ \boldsymbol{a}_i^{**(n)} &= \frac{\boldsymbol{f}_i^{(n)} + \boldsymbol{f}_{i,T}^{(n)}}{m_i^{(n)}} - \alpha'_g \boldsymbol{v}_i^{(n)} && (60) \end{aligned}$$

The effective terms become

$$\begin{aligned} \boldsymbol{a}_{g\to p}^{(n)'} &= \boldsymbol{a}_{g\to p}^{**(n)} - \alpha_{PIC}(\beta)\big(\boldsymbol{v}_p^{(n)} - \boldsymbol{v}_{g\to p}^{(n)}\big) - \alpha'_p \boldsymbol{v}_p^{(n)} && (61) \\ \boldsymbol{v}_{g\to p}^{(n+1)'} &= \boldsymbol{v}_{g\to p}^{**(n+1)} - \Big(\alpha_{PIC}(\beta)\big(\boldsymbol{v}_p^{(n)} - \boldsymbol{v}_{g\to p}^{(n)}\big) + \alpha'_p \boldsymbol{v}_p^{(n)}\Big)\Delta t && (62) \end{aligned}$$

where $\boldsymbol{v}_{g\to p}^{(n)}$ is found from available input properties using

$$\boldsymbol{v}_{g\to p}^{(n)} = \boldsymbol{v}_{g\to p}^{**(n+1)} - \boldsymbol{a}_{g\to p}^{**(n)} \Delta t \qquad (63)$$

The final updates are the same as above except when coding

$$\begin{aligned} \boldsymbol{v}_* &= \boldsymbol{v}_{g\to p}^{**(n+1)} + \alpha_{PIC} \boldsymbol{v}_{g\to p}^{(n)} \Delta t && (64) \\ \boldsymbol{a}_* &= \boldsymbol{a}_{g\to p}^{**(n)} + \alpha_{PIC} \boldsymbol{v}_{g\to p}^{(n)} && (65) \end{aligned}$$

## 5.2 Momentum Approach to Strain and Particle Updates

### 5.2.1 Incremental Displacement Gradient

For strain updates, the algorithm needs to evaluate the spatial velocity gradient, $\nabla \boldsymbol{v}(x,t)$, and use that to find the incremental deformation gradient, $\mathsf{dF}$, defined by

$$\mathsf{dF}^{(n)} = \exp(\nabla \boldsymbol{v} \Delta t) \qquad (66)$$

which assumes $\nabla \boldsymbol{v}$ is constant over the time step. The gradient is found by extrapolating spatial velocity on the grid:

$$\nabla \boldsymbol{v} = \frac{\sum_i \boldsymbol{G}_{ip} \otimes \boldsymbol{p}_i^{(n)}}{\sum_i S_{ip} m_i^{(n)}} \qquad (67)$$

### 5.2.2 Particle Updates with Damping

In each time step, the Eulerian update on the grid is used to update the Lagrangian momenta and positions on the particles, and it is useful to include damping options in both the grid and particle updates. Two global damping strategies are to add damping terms proportional to either the grid momentum or the particle momentum. In most cases, these two approaches to damping should have similar results. By including them both, however, it is possible to propose several different types of damping schemes. If damping is based on grid momentum, then the nodal momentum update changes to:

$$p_i^{*(n+1)} = p_i^{(n)} + \boldsymbol{f}_i^{*(n)} \Delta t \qquad (68)$$

where "*" indicates a term that is revised to including damping affects in this time step. The effective nodal force is

$$\boldsymbol{f}_i^{*(n)} = \frac{\boldsymbol{p}_i^{*(n+1)} - \boldsymbol{p}_i^{(n)}}{\Delta t} = (\boldsymbol{f}_i^{(n)} + \boldsymbol{f}_{i,T}^{(n)}) - \alpha_g(t)\boldsymbol{p}_i^{(n)} \tag{69}$$

where $\alpha_g(t)$ is the grid damping constant (with units 1/sec) that applies to all nodes (*i.e.*, independent of $\boldsymbol{x}_i$), but may evolve in time (such as a kinetic energy thermostat). See below for how to deal with position-dependent damping.

We can write two updates for the particles — a FLIP update that increments particle momentum using forces extrapolated to the grid and a PIC update that extrapolates momentum directly to the particle. After adding a particle damping constant, $\alpha_p(t)$, these two updates become:

$$\begin{aligned}
\boldsymbol{p}_{p,FLIP}^{(n+1)} &= \boldsymbol{p}_p^{(n)} + (\boldsymbol{f}_{g\to p}^{*(n)} - \alpha_p(t)\boldsymbol{p}_p^{(n)})\Delta t && (70)\\
\boldsymbol{p}_{p,PIC}^{(n+1)} &= \boldsymbol{p}_{g\to p}^{*(n+1)} - \alpha_p(t)\boldsymbol{p}_p^{(n)}\Delta t = \boldsymbol{p}_{g\to p}^{(n)} + (\boldsymbol{f}_{g\to p}^{*(n)} - \alpha_p(t)\boldsymbol{p}_p^{(n)})\Delta t && (71)
\end{aligned}$$

where $\boldsymbol{f}_{g\to p}^{*(n)}$ is effective force extrapolated from the grid to the particle:

$$\boldsymbol{f}_{g\to p}^{*(n)} = \sum_i \boldsymbol{f}_i^{*(n)} S_{ip}^{(n)} = \boldsymbol{f}_{g\to p}^{(n)} - \alpha_g(t)\boldsymbol{p}_{g\to p}^{(n)} \tag{72}$$

and

$$\boldsymbol{f}_{g\to p}^{(n)} = \sum_i (\boldsymbol{f}_i^{(n)} + \boldsymbol{f}_{i,T}^{(n)}) S_{ip}^{(n)} \qquad \text{and} \qquad \boldsymbol{p}_{g\to p}^{(n)} = \sum_i \boldsymbol{p}_i^{(n)} S_{ip}^{(n)} \tag{73}$$

The extrapolated mometa are

$$\begin{aligned}
\boldsymbol{p}_{g\to p}^{*(n+1)} &= \sum_i \boldsymbol{p}_i^{*(n+1)} S_{ip}^{(n)} = \boldsymbol{p}_{g\to p}^{(n+1)} - \alpha_g(t)\boldsymbol{p}_{g\to p}^{(n)}\Delta t = \boldsymbol{p}_{g\to p}^{(n)} + \boldsymbol{f}_{g\to p}^{*(n)}\Delta t && (74)\\
\boldsymbol{p}_{g\to p}^{(n+1)} &= \sum_i \Big(\boldsymbol{p}_i^{(n)} + (\boldsymbol{f}_i^{(n)} + \boldsymbol{f}_{i,T}^{(n)})\Delta t\Big) S_{ip}^{(n)} = \boldsymbol{p}_{g\to p}^{(n)} + \boldsymbol{f}_{g\to p}^{(n)}\Delta t && (75)
\end{aligned}$$

Both damping strategies can be done entirely within the particle update. The two methods can be used individually or combined. Because grid and particle momenta should be similar, the total damping constant is $\alpha(t) = \alpha_g(t) + \alpha_p(t)$.

Next, the MPM position update with damping is found by integrating the extrapolated velocity over the time step, which comes from PIC momentum, by the midpoint rule:

$$\begin{aligned}
\boldsymbol{x}_p^{(n+1)} &= \boldsymbol{x}_p^{(n)} + \int_0^{\Delta t} \frac{\boldsymbol{p}_{p,PIC}^{(n+1)}(\Delta t = t)}{m_{g\to p}^{(n)}}\, dt \approx \boldsymbol{x}_p^{(n)} + \frac{1}{2m_{g\to p}^{(n)}}\Big(\boldsymbol{p}_{g\to p}^{(n)} + \boldsymbol{p}_{p,PIC}^{(n+1)}\Big)\Delta t && (76)\\
&= \boldsymbol{x}_p^{(n)} + \frac{\boldsymbol{p}_{g\to p}^{(n)}}{m_{g\to p}^{(n)}}\Delta t + \frac{1}{2m_{g\to p}^{(n)}}(\boldsymbol{f}_{g\to p}^{*(n)} - \alpha_p(t)\boldsymbol{p}_p^{(n)})(\Delta t)^2 && (77)
\end{aligned}$$

This update along with the FLIP momentum update (which is the one that should be used) can be written as:

$$\begin{aligned}
\boldsymbol{x}_p^{(n+1)} &= \boldsymbol{x}_p^{(n)} + \frac{\boldsymbol{p}_{g\to p}^{*(n+1)}}{m_{g\to p}^{(n)}}\Delta t - \frac{1}{2m_{g\to p}^{(n)}}(\boldsymbol{f}_{g\to p}^{*(n)} + \alpha_p(t)\boldsymbol{p}_p^{(n)})(\Delta t)^2 && (78)\\
\boldsymbol{p}_p^{(n+1)} &= \boldsymbol{p}_p^{(n)} + (\boldsymbol{f}_{g\to p}^{*(n)} - \alpha_p(t)\boldsymbol{p}_p^{(n)})\Delta t && (79)
\end{aligned}$$

It is interesting to redefine the grid and particle damping terms using:

$$\alpha_g(t) \to -\frac{1-\beta}{\Delta t} + \alpha'_g \qquad \text{and} \qquad \alpha_p(t) \to \frac{1-\beta}{\Delta t} + \alpha'_p \tag{80}$$

The time dependence of primed damping terms have been dropped, but they can all still depend on time. By unraveling all effective terms, the net updates become

$$\begin{aligned}
\boldsymbol{x}_p^{(n+1)} &= \boldsymbol{x}_p^{(n)} + \frac{\boldsymbol{p}_{g\to p}^{(n+1)}}{m_{g\to p}^{(n)}}\Delta t - \frac{\big(\boldsymbol{f}_{g\to p}^{(n)} + \alpha_{PIC}(\beta)\big(\boldsymbol{p}_p^{(n)} - \boldsymbol{p}_{g\to p}^{(n)}\big) + \alpha'_g \boldsymbol{p}_{g\to p}^{(n)} + \alpha'_p \boldsymbol{p}_p^{(n)}\big)(\Delta t)^2}{2m_{g\to p}^{(n)}} && (81)\\
\boldsymbol{p}_p^{(n+1)} &= \boldsymbol{p}_p^{(n)} + \big(\boldsymbol{f}_{g\to p}^{(n)} - \alpha_{PIC}(\beta)\big(\boldsymbol{p}_p^{(n)} - \boldsymbol{p}_{g\to p}^{(n)}\big) - \alpha_g \boldsymbol{p}_{g\to p}^{(n)} - \alpha'_p \boldsymbol{p}_p^{(n)}\big)\Delta t && (82)
\end{aligned}$$

The "PIC Damping" term is

$$\alpha_{PIC}(\beta) = \frac{1-\beta}{\Delta t} \tag{83}$$

It is called PIC damping, because in the absence of other damping terms ($\alpha'_g = \alpha'_p = 0$), the momentum update becomes

$$\boldsymbol{p}_p^{(n+1)} = \boldsymbol{p}_p^{(n)} - (1-\beta)\big(\boldsymbol{p}_p^{(n)} - \boldsymbol{p}_{g\to p}^{(n)}\big) + \boldsymbol{f}_{g\to p}^{(n)}\Delta t \tag{84}$$

If $\beta = 1$, the update is an undamped FLIP update. If $\beta = 0$, the update is an undamped, PIC update. Thus $\beta$ can be interpreted as the fraction FLIP in the momentum update. But the above analysis was formally all a FLIP analysis. A better interpretation is that $\beta$ adds an new form of artificial damping that is proportional to the inversion error between the particle momentum and the grid momentum extrapolated to the particle ($\boldsymbol{p}_p^{(n)} - \boldsymbol{p}_{g\to p}^{(n)}$). The net damping effect will be to damp out this error.

There are two approaches to coding updates with damping. First, it can be done entirely within the particle update task. The input to this task from the nodes will be $p_i^{(n+1)}$ (which will have replaced $p_i^{(n)}$ in the momentum update), $\boldsymbol{f}_i^{(n)} + \boldsymbol{f}_{i,T}^{(n)}$, and $m_i^{(n)}$. We can define two new effective terms

$$\begin{aligned}
\boldsymbol{f}_{g\to p}^{(n)'} &= \boldsymbol{f}_{g\to p}^{(n)} - \alpha_{PIC}(\beta)\big(\boldsymbol{p}_p^{(n)} - \boldsymbol{p}_{g\to p}^{(n)}\big) - \alpha'_g \boldsymbol{p}_{g\to p}^{(n)} - \alpha'_p \boldsymbol{p}_p^{(n)} && (85)\\
&= \boldsymbol{f}_{g\to p}^{(n)} - \alpha_g(t)\boldsymbol{p}_{g\to p}^{(n)} - \alpha_p(t)\boldsymbol{p}_p^{(n)} && (86)\\
\boldsymbol{p}_{g\to p}^{(n+1)'} &= \boldsymbol{p}_{g\to p}^{(n+1)} - \left(\alpha_{PIC}(\beta)\big(\boldsymbol{p}_p^{(n)} - \boldsymbol{p}_{g\to p}^{(n)}\big) + \alpha'_g \boldsymbol{p}_{g\to p}^{(n)} + \alpha'_p \boldsymbol{p}_p^{(n)}\right)\Delta t && (87)\\
&= \boldsymbol{p}_{g\to p}^{(n+1)} - \left(\alpha_g(t)\boldsymbol{p}_{g\to p}^{(n)} + \alpha_p(t)\boldsymbol{p}_p^{(n)}\right)\Delta t && (88)
\end{aligned}$$

where $\boldsymbol{p}_{g\to p}^{(n)}$ is found from available input properties using

$$\boldsymbol{p}_{g\to p}^{(n)} = \boldsymbol{p}_{g\to p}^{(n+1)} - \boldsymbol{f}_{g\to p}^{(n)}\Delta t \tag{89}$$

The final updates become

$$\begin{aligned}
\boldsymbol{x}_p^{(n+1)} &= \boldsymbol{x}_p^{(n)} + \frac{\boldsymbol{p}_{g\to p}^{(n+1)'}}{m_{g\to p}^{(n)}}\Delta t - \frac{\boldsymbol{f}_{g\to p}^{(n)'}}{2m_{g\to p}^{(n)}}(\Delta t)^2 && (90)\\
\boldsymbol{p}_p^{(n+1)} &= \boldsymbol{p}_p^{(n)} + \boldsymbol{f}_{g\to p}^{(n)'}\Delta t && (91)
\end{aligned}$$

which are standard position and momentum updates to second order in position. Notice that it was important to keep the second order term on the position update in order to end up with consistent definitions for extrapolated momentum and force. In object oriented code, the grid damping is added in the update particles task, but the particle damping is not added until calling the material point class. Separating them out gives

$$\begin{aligned}
\boldsymbol{x}_p^{(n+1)} &= \boldsymbol{x}_p^{(n)} + \frac{\big(\boldsymbol{p}_* - \alpha_p(t)\boldsymbol{p}_p^{(n)}\Delta t\big)\Delta t}{m_{g\to p}^{(n)}} - \frac{1}{2}\frac{\big(\boldsymbol{a}_* - \alpha_p(t)\boldsymbol{v}_p^{(n)}\big)(\Delta t)^2}{m_{g\to p}^{(n)}} && (92)\\
&= \boldsymbol{x}_p^{(n)} + \frac{\Delta t}{m_{g\to p}^{(n)}}\left(\boldsymbol{p}_* - \frac{\Delta t}{2}\big(\boldsymbol{f}_* + \alpha_p(t)\boldsymbol{p}_p^{(n)}\big)\right) && (93)\\
\boldsymbol{p}_p^{(n+1)} &= \boldsymbol{p}_p^{(n)} + \big(\boldsymbol{f}_* - \alpha_p(t)\boldsymbol{p}_p^{(n)}\big)\Delta t = \boldsymbol{p}_p^{(n)}\big(1-\alpha_p(t)\Delta t\big) + \boldsymbol{f}_*\Delta t && (94)
\end{aligned}$$

where

$$\begin{aligned}
\boldsymbol{p}_* &= \boldsymbol{p}_{g\to p}^{(n+1)} - \alpha_g(t)\boldsymbol{p}_{g\to p}^{(n)}\Delta t && (95)\\
\boldsymbol{f}_* &= \boldsymbol{f}_{g\to p}^{(n)} - \alpha_g(t)\boldsymbol{p}_{g\to p}^{(n)} && (96)
\end{aligned}$$

The position update must be called first, because it needs $\boldsymbol{p}_p^{(n)}$, which is changed in the momentum update.

When all damping is done in the particle updates (as is possible from above equations), it is possible that damping will affect other sections of the code that use nodal momentum or force, such as contact, differently. To avoid these effects (if they matter), a second approach would be to apply the grid damping to the force calculation. But, when PIC damping is used, this could cause very unrealistic forces (because that term can be large when $\Delta t$ is small). A potential solution is to apply only $\alpha_g'$ to force calculations and then apply PIC damping during the particle updates. At the time of the particle update, the nodal input values would be

$$\begin{aligned}
\boldsymbol{f}_i^{**(n)} &= \boldsymbol{f}_i^{(n)} + \boldsymbol{f}_{i,T}^{(n)} - \alpha_g'\boldsymbol{p}_i^{(n)} && (97)\\
\boldsymbol{p}_i^{**(n+1)} &= p_i^{(n)} + \boldsymbol{f}_i^{**(n)}\Delta t && (98)
\end{aligned}$$

The effective terms become

$$\begin{aligned}
\boldsymbol{f}_{g\to p}^{(n)'} &= \boldsymbol{f}_{g\to p}^{**(n)} - \alpha_{PIC}(\beta)\big(\boldsymbol{p}_p^{(n)} - \boldsymbol{p}_{g\to p}^{(n)}\big) - \alpha_p'\boldsymbol{p}_p^{(n)} && (99)\\
\boldsymbol{p}_{g\to p}^{(n+1)'} &= \boldsymbol{p}_{g\to p}^{**(n+1)} - \left(\alpha_{PIC}(\beta)\big(\boldsymbol{p}_p^{(n)} - \boldsymbol{p}_{g\to p}^{(n)}\big) + \alpha_p'\boldsymbol{p}_p^{(n)}\right)\Delta t && (100)
\end{aligned}$$

where $\boldsymbol{p}_{g\to p}^{(n)}$ is found from available input properties using

$$\boldsymbol{p}_{g\to p}^{(n)} = \boldsymbol{p}_{g\to p}^{**(n+1)} - \boldsymbol{f}_{g\to p}^{**(n)}\Delta t \qquad (101)$$

The final updates are the same as above except when coding

$$\begin{aligned}
\boldsymbol{p}_* &= \boldsymbol{p}_{g\to p}^{**(n+1)} + \alpha_{PIC}\boldsymbol{p}_{g\to p}^{(n)}\Delta t && (102)\\
\boldsymbol{f}_* &= \boldsymbol{f}_{g\to p}^{**(n)} + \alpha_{PIC}\boldsymbol{p}_{g\to p}^{(n)} && (103)
\end{aligned}$$

## 5.3 Position Dependent Damping Strategies

Potentially it might be useful to damp certain regions differently than other regions. For example, damping could be used in a crack plane to absorbed released energy. This section considers changes when the damping terms depend on position. For damping based on grid velocity, the nodal momentum update changes to:

$$p_i^{*(n+1)} = p_i^{(n)} + \boldsymbol{a}_i^{*(n)}\Delta t \tag{104}$$

where "*" indicates a term that is revised to including damping affects in this time step. The effective nodal acceleration is

$$\boldsymbol{a}_i^{*(n)} = \frac{\boldsymbol{p}_i^{*(n+1)} - \boldsymbol{p}_i^{(n)}}{m_i^{(n)}\Delta t} = \frac{\boldsymbol{f}_i^{(n)} + \boldsymbol{f}_{i,T}^{(n)} - \alpha_g(t,\boldsymbol{x}_i)\boldsymbol{p}_i^{(n)}}{m_i^{(n)}} = \frac{\boldsymbol{f}_i^{(n)} + \boldsymbol{f}_{i,T}^{(n)}}{m_i^{(n)}} - \alpha_g(t,\boldsymbol{x}_i)\boldsymbol{v}_i^{(n)} \tag{105}$$

where $\boldsymbol{x}_i$ is the nodal position. The particle velocity update would be:

$$\boldsymbol{v}_{p,FLIP}^{(n+1)} = \boldsymbol{v}_p^{(n)} + \Big(\boldsymbol{a}_{g\to p}^{(n)} - \left(\sum_i \alpha_g(t,\boldsymbol{x}_i)\boldsymbol{v}_i^{(n)} S_{ip}\right) - \alpha_p(t,\boldsymbol{x}_p)\boldsymbol{v}_p^{(n)}\Big)\Delta t \tag{106}$$

The summation terms causes some problems. First, it is inefficient because it needs another extrapolation and likely repeated calculation of $\alpha_g(t,\boldsymbol{x}_i)$. Second, it loses connection to simple velocity terms. The situation can be improved by replacing $\alpha_g(t,\boldsymbol{x}_i)$ with $\alpha_g(t,\boldsymbol{x}_p)$, which is constant for the sum and can be removed from the sum to give

$$\boldsymbol{v}_{p,FLIP}^{(n+1)} = \boldsymbol{v}_p^{(n)} + \big(\boldsymbol{a}_{g\to p}^{(n)} - \alpha_g(t,\boldsymbol{x}_p)\boldsymbol{v}_{g\to p}^{(n)} - \alpha_p(t,\boldsymbol{x}_p)\boldsymbol{v}_p^{(n)}\big)\Delta t \tag{107}$$

The remainder of the update analysis can be followed as above. The only difference in code is that $\alpha_g(t,\boldsymbol{x}_p)$ and $\alpha_p(t,\boldsymbol{x}_p)$ need to be calculated for each particle in the update loop while for time dependence only, they only need to be calculated once in the entire particle update task. In this approach, the damping could be implemented as a particle property rather then as a function of Eulerian grid coordinates.

## 6 GIMP Derivation Comments

The nodal force $\boldsymbol{f}_i$ is sometimes interpreted as a sum of internal force, body force, and external force (for the three terms, respectively), but there is no need to separate them during calculations; they all add to the nodal force. The force $\boldsymbol{f}_{i,T}$ is a force due to tractions and needs extra work to integrate the tractions over traction-loaded surfaces in the extent of the grid shape function for node $i$.

Note that Eq. (19) is diagonal in the nodal system. Thus unlike other derivations of MPM, which get to here with a non-diagonal mass matrix, there is no need to invoke lumping of that mass matrix to make the problem tractable. The GIMP derivation naturally results in a lumped mass matrix.

Within this generalized framework, the various style of MPM are characterized by how they calculate $S_{ip}$ and $\boldsymbol{G}_{ip}$. Given particle functions, $\chi_p(\boldsymbol{x})$, the above equations give an "exact" result that is sometimes called "finite GIMP." In general, exact integration of arbitrary particle domains is difficult and is therefore not done. Instead various schemes have developed to approximate the integrals. The `NairnMPM` wiki page as one approach to classify the MPM family tree of methods.

## 7 Axisymmetric GIMP

In axisymmetric GIMP, the virtual work integrals are converted to cylindrical integrals:

$$\int_A \rho\boldsymbol{b}\cdot\delta\boldsymbol{u}\,r dA + \int_{L_T} \boldsymbol{T}\cdot\delta\boldsymbol{u}\,r dL + \sum_p \boldsymbol{F_p}\cdot\delta\boldsymbol{u} = \int_A \rho\boldsymbol{a}\cdot\delta\boldsymbol{u}\,r dA + \int_A \boldsymbol{\sigma}\cdot\nabla\delta\boldsymbol{u}\,r dA \tag{108}$$

where $A$ is the area of integration in the $r$-$z$ plane, $dA = dr\,dz$, $L_T$ is the path for surfaces having traction loads, and $\boldsymbol{T}$ and $\boldsymbol{F_p}$ have been redefined to be traction and force per radian. The result after expansion in the particle basis changes to:

$$\sum_p \int_A \frac{m_p}{A_p\langle r_p\rangle}\chi_p(\boldsymbol{x})\boldsymbol{b_p}\cdot\delta\boldsymbol{u}\,r\,dA + \int_{L_T}\boldsymbol{T}\cdot\delta\boldsymbol{u}\,dL + \sum_p \frac{\boldsymbol{F}_p}{A_p\langle r_p\rangle}\int_A \chi_p(\boldsymbol{x})\cdot\delta\boldsymbol{u}\,r\,dA \tag{109}$$

$$= \sum_p \int_A \frac{\dot{\boldsymbol{p}}_{\boldsymbol{p}}}{A_p\langle r_p\rangle}\chi_p(\boldsymbol{x})\cdot\delta\boldsymbol{u}\,r\,dA + \sum_p \int_A \boldsymbol{\sigma}_p\chi_p(\boldsymbol{x})\cdot\nabla\delta\boldsymbol{u}\,r\,dA \tag{110}$$

where $A_p$ is the particle area in the $r-z$ plane, $\langle r_p\rangle$ is the average radial position of the particle, and particle mass has been redefined to be the mass per radian or $m_p = \rho_p A_p\langle r_p\rangle$. The particle basis functions have the new normalization of

$$A_p\langle r_p\rangle = \int_A \chi_p(\boldsymbol{x})\,r\,dA \qquad \text{where} \qquad \langle r_p\rangle = \frac{1}{A_p}\int_{A_p} r\,dA \tag{111}$$

The first four results above after expansion in the grid-based shape functions are identical except that definition of $S_{ip}$ changes to

$$S_{ip} = \frac{1}{A_p\langle r_p\rangle}\int_A \chi_p(\boldsymbol{x})N_i(\boldsymbol{x})\,r\,dA \tag{112}$$

The stress term needs some extra work by evaluating $\nabla\delta\boldsymbol{u}$ in cylindrical coordinates, which is:

$$\nabla\delta\boldsymbol{u} = \begin{pmatrix} \frac{\partial\delta u_r}{\partial r} & \frac{\partial\delta u_r}{\partial z} & 0 \\ \frac{\partial\delta u_z}{\partial r} & \frac{\partial\delta u_z}{\partial z} & 0 \\ 0 & 0 & \frac{\delta u_r}{r} \end{pmatrix} \tag{113}$$

The stress term evaluates to

$$\begin{aligned}\sum_i\sum_p\int_A \boldsymbol{\sigma}_p\chi_p(\boldsymbol{x})\cdot\delta\boldsymbol{u_i}\nabla N_i(\boldsymbol{x})\,r\,dA &= \sum_i\sum_p A_p\langle r_p\rangle\begin{pmatrix}\sigma_{rr} & \sigma_{rz}\\ \sigma_{rz} & \sigma_{zz}\end{pmatrix}\boldsymbol{G}_{ip}\cdot(\delta u_{i,z},\delta u_{i,z}) \\ &\quad + \sum_i\sum_p A_p\langle r_p\rangle(\sigma_{\theta\theta},0)T_{ip}\cdot(\delta u_{i,z},\delta u_{i,z})\end{aligned} \tag{114}$$

where $\boldsymbol{G}_{ip}$ is redefined and $T_{ip}$ is a new shape function:

$$\boldsymbol{G}_{ip} = \frac{1}{A_p\langle r_p\rangle}\int_A \chi_p(\boldsymbol{x})\nabla N_i(\boldsymbol{x})\,r\,dA \tag{115}$$

$$T_{ip} = \frac{1}{A_p\langle r_p\rangle}\int_A \chi_p(\boldsymbol{x})N_i(\boldsymbol{x})\,dA \tag{116}$$

$$\tag{117}$$

For the particle stress term, we can revise to use particle mass as follows:

$$A_p\langle r_p\rangle\boldsymbol{\sigma}_p = m_p\frac{\boldsymbol{\sigma}_p}{\rho_p} = m_p\frac{\boldsymbol{\sigma}_p}{\rho_p}\frac{\rho_0}{\rho_0} = m_p\frac{J\sigma_p}{\rho_0} = m_p\frac{\tau}{\rho_0} \tag{118}$$

Making use of the fact that $\delta \boldsymbol{u_i}$ is arbitrary, the summands of all terms can be equated to arrive at the controlling MPM equation on the background grid in axisymmetric calculations:

$$\frac{d\boldsymbol{p}_i}{dt} = \boldsymbol{f}_i + \boldsymbol{f}_{i,T} \tag{119}$$

where

$$\begin{aligned}
\boldsymbol{f}_i &= \sum_p \left( -\frac{m_p}{\rho_0} \begin{pmatrix} \tau_{rr} & \tau_{rz} \\ \tau_{rz} & \tau_{zz} \end{pmatrix} \cdot \boldsymbol{G}_{ip} - \frac{m_p}{\rho_0} (\tau_{\theta\theta}, 0) T_{ip} + m_p S_{ip} \boldsymbol{b}_p + S_{ip} \boldsymbol{F}_p \right) && (120) \\
\boldsymbol{f}_{i,T} &= \int_{S_T} \boldsymbol{N}_i(\boldsymbol{x}) \boldsymbol{T}\, dS && (121)
\end{aligned}$$

This final result is very similar to MPM in Cartesian coordinates except it uses revised shape functions, $m_p$, $\boldsymbol{F}_p$, and $\boldsymbol{T}$ are redefined to be quantities per radian, and there is an extra stress term in $\boldsymbol{f}_i$.

## 8 GIMP Analysis for Conduction

The heat conduction equation is

$$\rho C_P \frac{\partial T}{\partial t} + \nabla \cdot \boldsymbol{q} = q_s(\boldsymbol{x}) \tag{122}$$

where $\rho$ is density, $C_P$ is constant pressure heat capacity (per unit mass), $T$ is temperature, $\boldsymbol{q}$ is heat flux (per unit area), and $q_s(\boldsymbol{x})$ is heat source (per unit volume). For heat conduction, the heat flux is $\boldsymbol{q} = -\mathsf{k}\nabla T$ where $\mathsf{k}$ is the thermal conductivity tensor and $T$ is temperature. Solving this equation in the MPM weak form gives

$$\int_V \left( \rho C_P \frac{\partial T}{\partial t} + \nabla \cdot \boldsymbol{q} - q_s(\boldsymbol{x}) \right) w(\boldsymbol{x})\, dV = 0 \tag{123}$$

where $w(\boldsymbol{x})$ is an arbitrary weighting function. Using the vector identity:

$$(\nabla \cdot \boldsymbol{q})\, w(\boldsymbol{x}) = \nabla \cdot \big(w(\boldsymbol{x})\boldsymbol{q}\big) - (\nabla w(\boldsymbol{x})) \cdot \boldsymbol{q} \tag{124}$$

and the divergence theorem, the weak form equation becomes:

$$\int_V \left( \rho C_P w(\boldsymbol{x}) \frac{\partial T}{\partial t} - \nabla w(\boldsymbol{x}) \cdot \boldsymbol{q} - q_s(\boldsymbol{x}) w(\boldsymbol{x}) \right) dV + \int_{\delta V} (w(\boldsymbol{x})\boldsymbol{q}) \cdot \hat{n}\, dS = 0 \tag{125}$$

where $\delta V$ is the border of $V$ and $\hat{n}$ is a surface normal vector.

As in GIMP for the momentum equation, we expand the particle quantities in the particle basis to get:

$$\begin{aligned}
\rho C_P \frac{\partial T}{dt} &= \sum_p \rho_p C_{P,p} \frac{\partial T_p}{dt} \chi_p(\boldsymbol{x}) && (126) \\
q_s(\boldsymbol{x}) &= \sum_p q_{s,p} \chi_p(\boldsymbol{x}) && (127) \\
\boldsymbol{q} &= \sum_p \boldsymbol{q}_p \chi_p(\boldsymbol{x}) && (128)
\end{aligned}$$

where subscript $p$ denotes a particle property and $\chi_p(\boldsymbol{x})$ is the particle basis function for particle $p$ (which is typically 1 within the deformed particle domain and zero elsewhere). Next expand weight function and its gradient in the grid shape functions:

$$w(\boldsymbol{x}) = \sum_i w_i N_i(\boldsymbol{x}) \qquad \text{and} \qquad \nabla w(\boldsymbol{x}) = \sum_i w_i \nabla N_i(\boldsymbol{x}) \tag{129}$$

After substituting all expansions, the weak form equation becomes

$$-\sum_i \int_{\delta V} (w_i N_i(\boldsymbol{x})\boldsymbol{q}) \cdot \hat{n}\, dS = \int_V \Bigg\{ -\sum_i \sum_p \Big[ \big(w_i \nabla N_i(\boldsymbol{x})\big) \cdot \boldsymbol{q}_p \chi_p(\boldsymbol{x}) \Big] \tag{130}$$

$$- \sum_i \sum_p q_{s,p} \chi_p(\boldsymbol{x}) w_i N_i(\boldsymbol{x}) + \sum_i \sum_p \rho_p C_{P,p} \chi_p(\boldsymbol{x}) w_i N_i(\boldsymbol{x}) \frac{\partial T_p}{dt} \Bigg\} dV \tag{131}$$

Using Eqs. (16) and (17) for GIMP shape functions and arbitrary nature of $w(\boldsymbol{x})$, this equation transforms to an equation for each node:

$$\sum_p m_p C_{P,p} \frac{\partial T_p}{dt} S_{ip} = \sum_p V_p \boldsymbol{q}_p \cdot \boldsymbol{G}_{ip} + \sum_p V_p q_{s,p} S_{ip} - \int_{\delta V} (N_i(\boldsymbol{x})\boldsymbol{q}) \cdot \hat{n}\, dS \tag{132}$$

where $m_p = \rho_p V_p = \rho_0 V_0$ is particle mass. This equation works for large deformation provided $V_P$ is the current particle volume and Eqs. (16) and (17) use finite GIMP methods. If Eqs. (16) and (17) are replaced by uniform GIMP (or CPDI), it still approximately accounts for large deformation by using current particle volume ($V_p$) in the sums.

## 8.1 MPM Equation

Defining $m_{Tp}^{(n)} = m_p C_{P,p}$ as a thermal mass on the particle in step $n$ (with SI units J/K when $C_P$ is heat capacity per unit mass or J/(K-kg)), a thermal momentum on the node can be defined as

$$p_{Ti}^{(n)} = \sum_p m_{Tp}^{(n)} T_p^{(n)} S_{ip}^{(n)} \tag{133}$$

The SI units for thermal momentum are J (Joules) (or N-m). In the momentum analog, temperature is replaced by velocity, thermal mass is replaced by actual mass, and the sum gives momentum with SI units N-sec. A thermal momentum equation, as scalar analog of the momentum equation, can be written as

$$\frac{dp_{Ti}^{(n)}}{dt} = Q_i^{(n)} + Q_{i,q}^{(n)} \tag{134}$$

where

$$Q_i^{(n)} = \sum_p V_p^{(n)} \left( \boldsymbol{q}_p^{(n)} \cdot \boldsymbol{G}_{ip}^{(n)} + q_{s,p}^{(n)} S_{ip}^{(n)} \right) \tag{135}$$

$$Q_{i,q}^{(n)} = -\int_{\delta V} (N_i(\boldsymbol{x})\boldsymbol{q}^{(n)}) \cdot \hat{n}\, dS \tag{136}$$

are total thermal flows (or thermal forces) with SI units J/sec or Watts. The first heat flow is internal heat flow while the second is heat flow at the boundaries due to flux boundary conditions. The $\boldsymbol{q}_p^{(n)}$ term is flux with SI units W/m$^2$. It is analogous to stress term in the momentum equation (using units

N/m$^2$), but rather then track and update $\boldsymbol{q}_p^{(n)}$ on the particle (as done for stress), it is calculated on each time step using

$$\boldsymbol{q}_p^{(n)} = -\mathsf{k}_p^{(n)} \nabla T_p^{(n)} = -\mathsf{k}_p^{(n)} \sum_i \mathsf{T}_i^{(n)} \boldsymbol{G}_{ip}^{(n)} \tag{137}$$

where nodal temperature is defined by

$$T_i^{(n)} = \frac{p_{Ti}^{(n)}}{m_{Ti}^{(n)}} \qquad \text{where} \qquad m_{Ti}^{(n)} = \sum_p m_{Tp}^{(n)} S_{ip}^{(n)} \tag{138}$$

Note that conductivity has SI units of W/(m K) and $q_{s,p}^{(n)}$ has SI units W/m$^3$.

## 8.2 Momentum and Particle Updates

The thermal momentum update on the node is

$$p_{Ti}^{(n+1)} = p_{Ti}^{(n)} + \left(Q_i^{(n)} + Q_{i,q}^{(n)}\right)\Delta t = p_{Ti}^{(n)} + m_{Ti}^{(n)} a_{Ti}^{(n)} \Delta t \tag{139}$$

where the nodal thermal acceleration is

$$a_{Ti}^{(n)} = \frac{p_{Ti}^{(n+1)} - p_{Ti}^{(n)}}{m_{Ti}^{(n)} \Delta t} = \frac{Q_i^{(n)} + Q_{i,q}^{(n)}}{m_{Ti}^{(n)}} \tag{140}$$

As done for velocity, we can write two updates for the particles — a FLIP update that increments particle temperature using thermal accelerations extrapolated to the grid and a PIC update that extrapolates temperature directly to the particle:

$$T_{p,FLIP}^{(n+1)} = T_p^{(n)} + a_{T,g\to p}^{(n)} \Delta t \tag{141}$$

$$T_{p,PIC}^{(n+1)} = T_{g\to p}^{(n+1)} \tag{142}$$

where

$$a_{T,g\to p}^{(n)} = \sum_i a_{Ti}^{(n)} S_{ip}^{(n)} \tag{143}$$

$$T_{g\to p}^{(n+1)} = \sum_i \left(T_i^{(n)} + a_{Ti}^{(n)} \Delta t\right) S_{ip}^{(n)} = T_{g\to p}^{(n)} + a_{T,g\to p}^{(n)} \Delta t \tag{144}$$

$$T_{g\to p}^{(n)} = \sum_i T_i^{(n)} S_{ip}^{(n)} \tag{145}$$

Unlike like particle velocity and position update, the addition of PIC character to temperature updates leads to thermal conduction, even if the conductivity of the material is set to zero. Thus, the particle temperature update is always done using FLIP. Like particle stresses, particle temperatures can develop variations within a cell. Because of these variations, particle temperature does not give the best measure of the local temperature when implementing features such as particle properties that depend on temperature. To implement temperature-dependent features on time step $n$, it is better to assume the temperature at the particle is equal to $T_{g\to p}^{(n)}$. In other words, the particle temperature evolves as described above, but whenever some other feature of the code needs to know the temperature at the particle, it should use the grid-based temperature and not the evolving particle temperature.

## 9 GIMP Analysis for Diffusion

We can derive GIMP diffusion analysis from the GIMP conduction analysis by replacing temperature with concentration potential $\mu$ (where $\mu$ is a dimensionless potential approximated as $\mu = c/c_{sat}$; *i.e.* from 0 to 1), and then by setting $\rho = 1$, and $C_P = 1$, and $\mathsf{k} = \mathsf{D}$, where $\mathsf{D}$ is the diffusion tensor. Defining $V_p^{(n)}$ as a solvent "mass" on the particle in step $n$ (with SI units m$^3$), a solvent momentum on the node can be defined as

$$p_{Di}^{(n)} = \sum_p V_p^{(n)} \mu_p^{(n)} S_{ip}^{(n)} \tag{146}$$

The SI units for solvent momentum are m$^3$-$\mu$. A solvent momentum equation, as scalar analog of the momentum equation, can be written as

$$\frac{dp_{Di}^{(n)}}{dt} = S_i^{(n)} + S_{i,q}^{(n)} \tag{147}$$

where

$$S_{Di}^{(n)} = \sum_p V_p^{(n)} \left( \boldsymbol{s}_p^{(n)} \cdot \boldsymbol{G}_{ip}^{(n)} + s_{s,p}^{(n)} S_{ip}^{(n)} \right) \tag{148}$$

$$S_{Di,q}^{(n)} = -\int_{\delta V} (N_i(\boldsymbol{x}) \boldsymbol{s}^{(n)}) \cdot \hat{n}\, dS \tag{149}$$

are total solvent flow (or solvent forces) with SI units m$^3$-$\mu$/sec. The first flow is internal flow while the second is flow across the surfaces due to flux boundary conditions. The $\boldsymbol{s}_p^{(n)}$ term is flux with SI units m-$\mu$/sec (or (solvent force)/m$^2$). It is analogous to stress term in the momentum equation (also with units of (force)/m$^2$), but rather then track and update $\boldsymbol{s}_p^{(n)}$ on the particle, it is calculated on each time step using

$$\boldsymbol{s}_p^{(n)} = -\mathsf{D}_p^{(n)} \nabla \mu_p^{(n)} = -\mathsf{D}_p^{(n)} \sum_i \boldsymbol{\mu}_i^{(n)} \boldsymbol{G}_{ip}^{(n)} \tag{150}$$

where nodal concentration potential is defined by

$$\mu_i^{(n)} = \frac{p_{Di}^{(n)}}{V_i^{(n)}} \qquad \text{where} \qquad V_i^{(n)} = \sum_p V_p^{(n)} S_{ip}^{(n)} \tag{151}$$

Note that diffusion tensor has SI units of m$^2$/sec and $s_{s,p}^{(n)}$ (which is solvent source term) has SI units $\mu$/sec.

The nodal update becomes

$$p_{Di}^{(n+1)} = p_{Di}^{(n)} + \big(S_i^{(n)} + S_{i,q}^{(n)}\big)\Delta t = p_{Di}^{(n)} + V_i^{(n)} a_{Di}^{(n)} \Delta t \quad \text{where} \quad a_{Di}^{(n)} = \frac{S_i^{(n)} + S_{i,q}^{(n)}}{V_i^{(n)}} \tag{152}$$

The FLIP particle update becomes

$$\mu_{p,FLIP}^{(n+1)} = \mu_p^{(n)} + a_{D,g\to p}^{(n)} \Delta t \tag{153}$$

$$a_{D,g\to p}^{(n)} = \sum_i a_{Di}^{(n)} S_{ip}^{(n)} \tag{154}$$

$$\tag{155}$$