Implementation of Crack and Material Velocity Fields with Contact and Nodal Velocity Boundary Conditions

John Nairn

February 3, 2023

Contents

Chapter 1

Velocity Fields

Each node has one to four crack velocity fields denoted [0], [1], [2], and [3]. If the problem has no cracks, then each node has just the single crack velocity field [0]. Each crack velocity field has any number of material velocity fields or just one velocity field if using single material mode.

1.1 Crack Velocity Fields

The crack velocity field is determined by drawing a line from the particle to the node. The crack velocity field number n is denoted by square brackets — $[n]$. If the line crosses no cracks, that material point uses field [0]. If the line crosses one crack, that material point uses field [1] (for the first crack found) or field [2] (if a second crack is found for the same node). If the line crosses two cracks, the material point uses field [3]. This scheme can handle one or two cracks at each node. A third crack (*i.e.*, a line that crosses a third crack or one of the cracks in field [3] being different than the cracks for fields [1] and [2]) is handled by picking one of the four fields, but is likely to not geting the physics correct. A warning is issued if a third crack is found.

Figure 1.1 shows schematic view of crack velocity fields for the indicated nodal point. Because field [2] is never allocated before field [1] (*i.e.*, the first single crack crossing is put into field [1] and the second into field [2]), the following crack field situations are all that are possible: [0], [1], [3], [0]&[1], [0]&[3], [1]&[2], [1]&[3], [0]&[1]&[2], [0]&[1]&[3], [1]&[2]&[3], and [0]&[1]&[2]&[3]. The combinations that never occur are the remaining ones or [2], [0]&[2], [2]&[3], and [0]&[2]&[3].

When doing crack contact, the possible field contacts are between [0]&[1] and [2]&[3] on the surface of the first crack (*i.e.*, crack in [1] and [3]) and between [0]&[2] and [1]&[3] on the surface of the second crack (*i.e.*, crack in [2] and [3]). Fields [0]&[3] and [1]&[2] do not contact. These can be visualized by looking at Figure 1.1 where non-contacting fields meet at a point instead of a surface.

1.2 Material Velocity Fields

When running in multimaterial mode (which is needed to model material contact), each crack velocity field may have 1 to n material velocity fields where n is the number of materials active in the simulations. When not using multimaterial mode, n is set to 1 and all materials use the same velocity field in each crack velocity field. All these fields are on the node. In these notes, we denote a quantity, q, in a material velocity field (which is where momentum, forces, *etc.* are stored) on the grid as

$$q^{(k)}_{i,c:m} \tag{1.1}$$

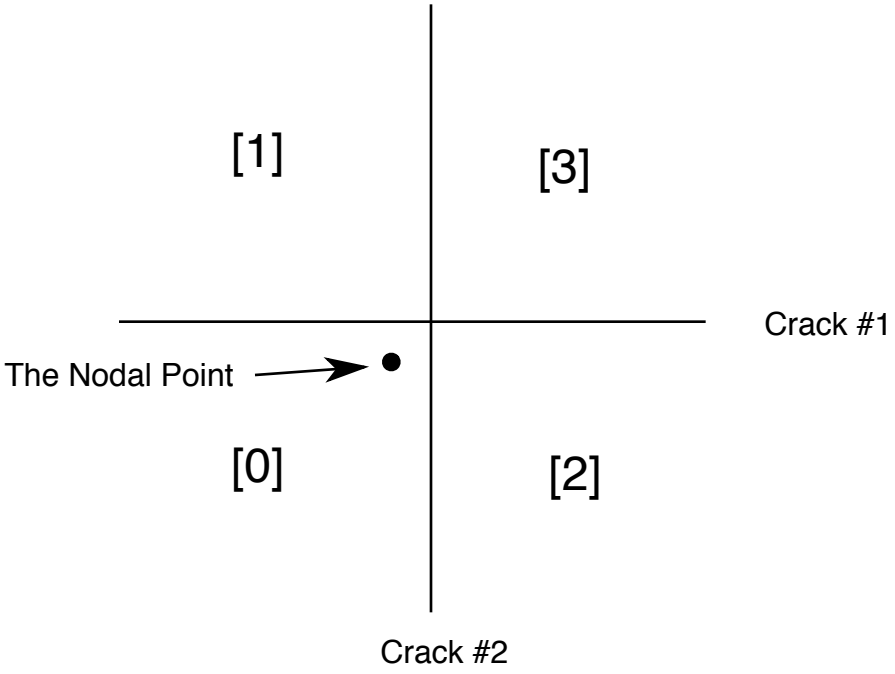

Figure 1.1: Schematic view of crack velocity fields around a single node. The possibility of two interacting cracks leads to at most four velocity fields.

where i is node number, c is crack velocity field, m is material velocity field, and k indicates time step number. When no cracks are present, c can be omitted; when in single material model, m can be omitted, when no cracks and single material mode both c and m can be omitted (*i.e.*, each node has only one material velocity field). When analyzing all values within a single time step, (k) can be omitted for simplicity.

Chapter 2

MPM Time Step Tasks

This section lists each task in NairnMPM outlining only those parts relevant to crack and material velocity fields, to contact conditions, to imperfect interfaces, to traction laws on cracks, and to nodal velocity boundary conditions.

2.1 Simulation Initialization

Each nodal point is permanently assigned crack field [0]. Other crack fields are dynamically allocated as needed. When in single material velocity field, field [0] is given a single material velocity field (field [0:0] for [crack field number:material field number]). When using multimaterial mode, crack velocity field [0] is assigned pointers to material velocity fields 0 to $m-1$ where m is the number of active materials in the simulations (each index maps to material number defined in the input file). The pointers are initialized to `NULL`. Any material velocity fields needed in a simulation are dynamically allocated as needed.

2.2 Time Step Initialization

All currently allocated crack and material velocity fields are zeroed. Then each material point is examined and compared to all nodes it sees with non-zero shape functions. When cracks a present, a line is traced from the particle to the node and compared to crack surfaces. The end result is determination of the particles crack velocity field number `vfld` from [0] to [3]. If the node does not have that crack velocity field, it is allocated. When no cracks are present `vfld` is set to 0. The crack field number, `vfld`, is stored on the particle. This crack field number determination is likely the most time-consuming part of explicit crack calculations.

Note that first crack found on a node by itself goes into field [1] and second goes into field [2], which means a node never have field [2], wtihout field [1]. Not counting a node with zero fields, there are 15 combinations of velocity fields possible on a node. Of these, 5 have no contact, 6 may have contact, and 4 never occur (because they have [2] without [1]):

- No Contact: [0], [1], [3], [0]&[3], [1]&[2]
- May Have Contact: [0]&[1], [1]&[3], [0]&[1]&[2], [0]&[1]&[3], [1]&[2]&[3], and [0]&[1]&[2]&[3].
- Never Occur: [2], [0]&[2], [2]&[3], [0]&[2]&[3]

Once crack field is determined, calculations have to go to the appropriate material velocity field. A table look up converts material number (from input file) to index `matfld` (or field ID) into the

`mvf[]` array of pointers on each crack velocity field. If that pointer `mvf[matfld]` is `NULL`, a new material velocity field is created and zeroed when needed.

2.3 Momentum and Mass Extrapolation to the Grid

For each material point, determine the appropriate material field (`matfld`) from the material assigned to that material point. Add the momentum and mass to material velocity field `matfld` of crack velocity field `vfld` (where `vfld` is stored on the particle). Mass and momentum are always extrapolated:

$$m_{i,c:j} = \sum_{p\in j} S_{ip} m_p \tag{2.1}$$

$$\boldsymbol{p}_{i,c:j} = m_{i,c:j}\boldsymbol{v}_{i,c:j} = \sum_{p\in j} S_{ip}\boldsymbol{p}_p \tag{2.2}$$

To support crack and material contact, each node might also need particle particle position, particle displacement, particle size, and particle gradient. These quantities for material velocity field $j =$`matfld` in crack velocity field $c =$`vfld` for particle p are:

$$m_{i,c:j}\boldsymbol{x}_{i,c:j} = \sum_{p\in j} S_{ip} m_p \boldsymbol{x}_p \tag{2.3}$$

$$m_{i,c:j}\Delta\boldsymbol{x}_{i,c:j} = \sum_{p\in j} S_{ip} m_p \Delta\boldsymbol{x}_p \tag{2.4}$$

$$\Omega_{i,c:j} = \begin{cases} \sum_{p\in j} S_{ip} V_p & \text{3D} \\ \sum_{p\in j} S_{ip} A_p & \text{2D and axisymmetric} \end{cases} \tag{2.5}$$

$$\boldsymbol{g}_{i,c:j} = \begin{cases} \sum_{p\in j} \boldsymbol{G}_{ip} V_p & \text{3D} \\ \sum_{p\in j} \boldsymbol{G}_{ip} A_p & \text{2D and axisymmetric} \end{cases} \tag{2.6}$$

The time step (k) has been omitted. The extrapolations are loaded into contact terms vector area in material velocity fields. They are not always needed:

- Contact volume ($\Omega_{i,c:j}$) is almost always used and currently always extrapolated when doing contact.
- Material displacement ($\Delta\boldsymbol{x}_{i,c:j}$) always needed for interfaces and also needed when normal and separation found using uncorrected displacements (see Chapter 3). It could be skipped (but currently not coded) if all these hold:
  - No interfaces
  - No cracks OR crack using a `contactPosition` option mode
  - `LogReg/LinReg` normals with no rigid contact particles (or with `rigidBias`< 10) OR single material mode.
- Material position ($\boldsymbol{x}_{i,c:j}$) is only extrapolated if material or crack contact are using a `contactPosition` option (see Chapter 3).
- Gradient ($\boldsymbol{g}_{i,c:j}$) is extrapolated when needed. It is not needed for `SN` normals or for `LogReg/LinReg` normals with no rigid contact particles (or with `rigidBias`< 10).

2.3.1 Post Mass and Momentum Extrapolations

After the extrapolations are done, each node is revisited for more calculations and when needed to impose multimaterial and crack contact and imperfect interfaces. The steps are

1. Mirror/copy materials that ignore cracks from [0] to all other active crack fields. `OSParticulas` mirrors any material type specified as ignoring cracks. `NairnMPM` does only rigid materials and they are copied instead of mirrored.

2. Calculate total mass and number of materials in each velocity field. For mirrored fields in `OSParticulas`, only count mass of materials that ignore cracks for field [0]. Store a copy of the initially extrapolated momenta $\boldsymbol{p}_{i,c:j}$ for all crack and material velocity fields on all nodes.

3. Code for step #2 returns `true` if any crack field has more than one material and therefore might have material contact. If it does, create a `MaterialContactNode` object. The end result will be a list of all nodes that might have contact in the vector `materialContactNodes` (which is static member of `MaterialContactNode` class).

4. When have cracks, check if each node might have crack contact (see possible fields listed above that might have contact). If it does, create a `CrackNode` object. The end result will be a list of all nodes that might have crack contact in the vector `crackContactNodes` (which is static member of `CrackNode` class).

5. Call `ContactAndMomentaBCs()` with type of `MASS_MOMENTUM_CALL` to handle material and crack contact and interfaces as well as velocity boundary conditions.

2.3.2 Material Contact, Crack Contact, and Velocity Conditions

The `ContactAndMomentaBCs()` call in previous list does contact calculations and it may be called two or three times per time step. This section outlines the tasks done in that method when some nodes have multimaterial contact.

1. The entry code for all material contact is `CrackVelocityFieldMulti::MaterialContactOnCVF()`. If cracks are present, this calculation is done in each crack field (the subscript for crack field c is dropped below). It starts with following calculations:

   (a) Find center of mass momentum $\boldsymbol{P}_c = \sum_j \boldsymbol{p}_{i,j}$ and total mass $M_c = \sum_j m_{i,j}$.

   (b) Look for rigid materials setting `rigidMat` to material ID and `multiRigid` true if more than one.

   (c) Call `MaterialContactOnCVFLumped()` or call `RigidMaterialContactOnCVF()` if one or more are rigid. To explicitly handle 3+ materials on a node, try calling `MaterialContactThree()` first. If not handled by that call, fall back to `MaterialContactOnCVFLumped()`. Explicit handling requires recompiling with `THREE_MAT_CONTACT` activated and simulation setting lumping option in multimaterial mode command to 1.

2. Algorithm in `CrackVelocityFieldMulti::MaterialContactOnCVFLumped()`:

   (a) Before looping of each material (or over one material pair), do these calculations:

i. Get center of mass displacement in **dispc**:

$$\textbf{dispc} = \frac{1}{m_i}\sum_j m_{i,j}\boldsymbol{x}_{i,j} \quad \text{or} \quad \textbf{dispc} = \frac{1}{m_i}\sum_j m_{i,j}\Delta\boldsymbol{x}_{i,j}$$

depending on whether material contact is using extrapolated positions (with a correction) or displacement (with no correct). More details on this options if givine in subsequent chapters. This term is only used later to get **delta** and **delta** is only used to get `deln` in gradient methods.

ii. If any interfaces are being used and **dispc** was found using $\boldsymbol{x}_{i,j}$, then calculate a new center of mass displacements:

$$\textbf{dispcForInterface} = \frac{1}{m_i}\sum_j m_{i,j}\Delta\boldsymbol{x}_{i,j}$$

This term is only used to find tangential displacement for interface modeling

iii. For or `LogReg/LinReg` normals and no interfaces, neither **dispc** nor **dispcForInterface** are used. Their calculations could be skipped.

(b) Loop of each active materials. If possible (only two materials), the loop will look at only one material and the other material changes will be found by conservation of momentum. For each material in this loop:

- Preliminary Calculations
  - Get `massi`, `voli`, `volj` = rest lumped, `mred`, and `massRatio`. Not that

$$m_{red} = \frac{\texttt{massi} * (M_c - \texttt{massi})}{M_c}$$

but here just find $m_{red} = (M_c - \texttt{massi})/M_c$; it is multiplied by `massi` later.
  - Loop over other materials. Find other material with maximum volume and if being used find, lump gradients of other materials into **gradj** (otherwise **gradj** is undetermined). When done find contact law for material i with other material with maximum volume.
  - Get $\Delta p_{i,a}$ for momentum change that causes material a to move in the center-of-mass velocity field. See Chapter 4 for details.
  - If contact law is to ignore contact or `comContact` set by some flag, finish contact for this material by proceeding the "Momentum Change" step using $\Delta p_{i,a}$ and then continue to the next material in the loop.
- Contact Law Calculations
  - If mass ratio too low or too high, continue to next material
  - Get normal vector (if not already set by a developer flag). See chapter 3 and not that `LinReg/LogReg` normal returns calculated `deln` and sets `hasDeln` to true.
  - If `hasDeln` is false, get separation in `deln` from either positions or displacements. Is using positions, correct that value using a `contactPosition` method (see Chapter 3 for details).
  - If needed by contact law or interface, get contact area. If contact law is an interface and separation used position, switch **delta** to be based on displacements (by recalculating with **dispcForInterface**.

- For contact heat flow, get contact area (if previous step did not get it), and trhen do contact heat calculations.

- Frictional Contact
  - Get ΔF_i when in `UPDATE_MOMENTUM_CALL`. It is used for second-order heating and for improvement separation calculation (in some interface laws, such as Coulomb friction with interfacial stiffness).
  - Adjust $\Delta\boldsymbol{p}_{i,i}$ to reflect contact law. The calculations need input of initial $\Delta\boldsymbol{p}_{i,i}$, **norm**= $\hat{\boldsymbol{n}}$, `dotn`= $\Delta\boldsymbol{p}_{i,i}\cdot\hat{\boldsymbol{n}}$ and separation `deln`. May need `contactArea` and ΔF_i.
- Imperfect Interface Contact
  - Get tangent vector ($\hat{\boldsymbol{t}}$) and find **delta**$\cdot\hat{\boldsymbol{t}}$.
  - Get ΔF_i when in `UPDATE_MOMENTUM_CALL`. It is used for second-order heating and for improvement separation calculation.
  - Get interface force $\boldsymbol{f}_{i,i}^{(INT)}$ on material i which will depend on calculated material separation. If normal or tangential force is found, set the contact $\Delta p_{i,i}$ in the respective direction to zero. Some interface laws can become unstable (if two stiff). If they get too stiff, set interface force to zero, but keep corresponding component $\Delta p_{i,i}$ to reflect "sticking" mechanics. The final $\Delta p_{i,i}$ is added to materials. In `UPDATE_MOMENTUM_CALL` the interface forces are added $\Delta\boldsymbol{p}_{i,i}$ and interfacial energy is incremented.
- Momentum Change
  - Add $\Delta\boldsymbol{p}_{i,i}$ to materials. Add $-\Delta\boldsymbol{p}_{i,i}$ to other material is a single pair in one pas through the loop.
  - Track frictional heating

(c) When loop over materials is done:

- Convert frictional heating to flux condition
- Track total frictional work

3. Algorithm in `CrackVelocityFieldMulti::RigidMaterialContactOnCVF()`:

4. Crack Contact Calculations: For cracks, check contact on surface of first crack (*e.g.*, [0]&[1] and [2]&[3]) and on second cracks (*e.g.*, [0]&[2] and [1]&[3]. Any resulting $\Delta p_{i,c:a}$ is added to two sides of the target crack. If needed, the added momentum is spread out over all materials (weighted by their mass fraction) if the crack velocity field has more than one material.

5. Impose Grid Velocity Conditions: After contact changes nodal momenta, change them again to match grid velocity boundary conditions. In other words, grid conditions take precedence of contact changes. These conditions are only needed when about to update strain (*i.e.*, when in `USF` or `USAVG`± modes) and the calculation only need to adjust momenta. For a velocity BC on node i in direction $\hat{\boldsymbol{n}}$, we want the final momentum to be:

$$\boldsymbol{p}_{i,c:m}^{(k+1)} = m_{i,c:m}v_i^{(BC)}\hat{\boldsymbol{n}} + \left(\boldsymbol{p}_{i,c:m}^{(k)}\cdot\hat{\boldsymbol{t}}\right)\hat{\boldsymbol{t}} = \boldsymbol{p}_{i,c:m}^{(k)} + \left(m_{i,c:m}v_i^{(BC)} - \boldsymbol{p}_{i,c:m}^{(k)}\cdot\hat{\boldsymbol{n}}\right)\hat{\boldsymbol{n}}$$

such that component of final momentum in the $\hat{\boldsymbol{n}}$ direction is

$$\boldsymbol{p}_{i,c:m}^{(k+1)}\cdot\hat{\boldsymbol{n}} = m_{i,c:m}v_i^{(BC)}$$

To allow for multiple BCs on the same node, this change is imposed by looping twice over all nodal boundary conditions:

(a) Zero the initial component in the $\hat{\boldsymbol{n}}$ direction by setting

$$\boldsymbol{p}_{i,c:m}^{(k+1)'} = \boldsymbol{p}_{i,c:m}^{(k)} - \left(\boldsymbol{p}_{i,c:m}^{(k)} \cdot \hat{\boldsymbol{n}}\right) \hat{\boldsymbol{n}}$$

(b) Add the BC value by setting

$$\boldsymbol{p}_{i,c:m}^{(k+1)} = \boldsymbol{p}_{i,c:m}^{(k+1)'} + \left(m_{i,c:m} v_i^{(BC)}\right) \hat{\boldsymbol{n}}$$

For multiple BCs on the same node, each repeat of step #1 will make no additional change because

$$\boldsymbol{p}_{i,c:m}^{(k+1)'} \cdot \hat{\boldsymbol{n}} = 0$$

while multiple calls of step # will then result in

$$\boldsymbol{p}_{i,c:m}^{(k+1)} \cdot \hat{\boldsymbol{n}} = \sum m_{i,c:m} v_i^{(BC)}$$

But, what if there are two BCs with different normals. The zeroing process will result in

$$\begin{aligned}
\boldsymbol{p}_{i,c:m}^{(k+1)'} &= \boldsymbol{p}_{i,c:m}^{(k)} - \left(\boldsymbol{p}_{i,c:m}^{(k)} \cdot \hat{\boldsymbol{n}}_1\right) \hat{\boldsymbol{n}}_1 \\
\boldsymbol{p}_{i,c:m}^{(k+1)''} &= \boldsymbol{p}_{i,c:m}^{(k+1)'} - \left(\boldsymbol{p}_{i,c:m}^{(k+1)'} \cdot \hat{\boldsymbol{n}}_2\right) \hat{\boldsymbol{n}}_2 \\
&= \boldsymbol{p}_{i,c:m}^{(k)} - \left(\boldsymbol{p}_{i,c:m}^{(k)} \cdot \hat{\boldsymbol{n}}_1\right) \hat{\boldsymbol{n}}_1 - \left(\boldsymbol{p}_{i,c:m}^{(k)} \cdot \hat{\boldsymbol{n}}_2\right) \hat{\boldsymbol{n}}_2 + \cos\theta \left(\boldsymbol{p}_{i,c:m}^{(k)} \cdot \hat{\boldsymbol{n}}_1\right) \hat{\boldsymbol{n}}_2
\end{aligned}$$

If repeated calls are in order (*i.e.*, all $\hat{\boldsymbol{n}}_1$ first followed by all $\hat{\boldsymbol{n}}_2$), subsequent calls are acceptable because

$$\boldsymbol{p}_{i,c:m}^{(k+1)'} \cdot \hat{\boldsymbol{n}}_1 = \boldsymbol{p}_{i,c:m}^{(k+1)''} \cdot \hat{\boldsymbol{n}}_2 = 0$$

but, if an $\hat{\boldsymbol{n}}_1$ call comes after a previous $\hat{\boldsymbol{n}}_1$ and $\hat{\boldsymbol{n}}_2$ call

$$\boldsymbol{p}_{i,c:m}^{(k+1)''} \cdot \hat{\boldsymbol{n}}_1 = -\left(\boldsymbol{p}_{i,c:m}^{(k+1)'} \cdot \hat{\boldsymbol{n}}_2\right) \cos\theta = -\left(\boldsymbol{p}_{i,c:m}^{(k)} \cdot \hat{\boldsymbol{n}}_2 - \left(\boldsymbol{p}_{i,c:m}^{(k)} \cdot \hat{\boldsymbol{n}}_1\right) \cos\theta\right) \cos\theta$$

Finally, looping to add velocity conditions will result in

$$\begin{aligned}
\boldsymbol{p}_{i,c:m}^{(k+1)} = \boldsymbol{p}_{i,c:m}^{(k)} &+ \left(\sum m_{i,c:m} v_{i,1}^{(BC)} - \boldsymbol{p}_{i,c:m}^{(k)} \cdot \hat{\boldsymbol{n}}_1\right) \hat{\boldsymbol{n}}_1 \\
&+ \left(\sum m_{i,c:m} v_{i,2}^{(BC)} - \boldsymbol{p}_{i,c:m}^{(k)} \cdot \hat{\boldsymbol{n}}_2\right) \hat{\boldsymbol{n}}_2 + \cos\theta \left(\boldsymbol{p}_{i,c:m}^{(k)} \cdot \hat{\boldsymbol{n}}_1\right) \hat{\boldsymbol{n}}_2
\end{aligned}$$

All is acceptable if $\hat{\boldsymbol{n}}_1$ and $\hat{\boldsymbol{n}}_2$ are orthogonal (or $\cos\theta = 0$). Otherwise, trying to apply velocity BC in non-orthogonal directions will results in errors (*i.e.*, will not set correct velocities).

An approach to apply velocity in two non-orthogonal directions is as follows:

(a) If a node has velocity BC in two non-orthogonal directions, zero the momentum in the plane of the two normals. For 2D analyses, this step would set the momentum to zero. For 3D, it is set to

$$\boldsymbol{p}_{i,c:m}^{(k+1)'} = \left(\boldsymbol{p}_{i,c:m}^{(k)} \cdot \hat{\boldsymbol{n}}_X\right) \hat{\boldsymbol{n}}_X \quad \text{where} \quad \hat{\boldsymbol{n}}_X = \hat{\boldsymbol{n}}_1 \times \hat{\boldsymbol{n}}_2$$

(b) Then set the in-plane momentum to

$$\boldsymbol{p}_{i,c:m}^{(k+1)} = \frac{\sum m_{i,c:m} v_{i,1}^{(BC)} - \cos\theta \sum m_{i,c:m} v_{i,2}^{(BC)}}{1-\cos^2\theta}\hat{\boldsymbol{n}}_1 + \frac{\sum m_{i,c:m} v_{i,2}^{(BC)} - \cos\theta \sum m_{i,c:m} v_{i,1}^{(BC)}}{1-\cos^2\theta}\hat{\boldsymbol{n}}_2$$

For 3D analysis, add $\boldsymbol{p}_{i,c:m}^{(k+1)'}$ to retain the out-of-plane momentum. This setting results in

$$\boldsymbol{p}_{i,c:m}^{(k+1)} \cdot \hat{\boldsymbol{n}}_1 = \sum m_{i,c:m} v_{i,1}^{(BC)} \quad \text{and} \quad \boldsymbol{p}_{i,c:m}^{(k+1)} \cdot \hat{\boldsymbol{n}}_2 = \sum m_{i,c:m} v_{i,2}^{(BC)}$$

as desired.

Although this is not implemented in code, it can be duplicated as follows:

(a) Use Gram-Schmidt normalization to calculate a third normal that is orthogonal to $\hat{\boldsymbol{n}}_1$;

$$\hat{\boldsymbol{n}}_3 = \hat{\boldsymbol{n}}_2 - \cos\theta\hat{\boldsymbol{n}}_1 \quad \Longrightarrow \quad \hat{\boldsymbol{n}}_3 \cdot \hat{\boldsymbol{n}}_1 = 0$$

(b) Change the nodal BCs to use $\hat{\boldsymbol{n}}_1$ and $\hat{\boldsymbol{n}}_3$ with the momentum condition in direction $\hat{\boldsymbol{n}}_3$ set to

$$p_3 = \frac{\sum m_{i,c:m} v_{i,2}^{(BC)} - \cos\theta \sum m_{i,c:m} v_{i,1}^{(BC)}}{1-\cos^2\theta}$$

(c) The momentum after applying these orthogonal BCs will be

$$\boldsymbol{p}_{i,c:m}^{(k+1)} = \sum m_{i,c:m} v_{i,1}^{(BC)} \hat{\boldsymbol{n}}_1 + p_3\hat{\boldsymbol{n}}_3 + \left(\boldsymbol{p}_{i,c:m}^{(k)} \cdot \hat{\boldsymbol{n}}_X\right)\hat{\boldsymbol{n}}_X$$

Substituting for p_3 and $\hat{\boldsymbol{n}}_3$ returns the proper momentum above to give

$$\boldsymbol{p}_{i,c:m}^{(k+1)} \cdot \hat{\boldsymbol{n}}_1 = \sum m_{i,c:m} v_{i,1}^{(BC)} \quad \text{and} \quad \boldsymbol{p}_{i,c:m}^{(k+1)} \cdot \hat{\boldsymbol{n}}_2 = \sum m_{i,c:m} v_{i,2}^{(BC)}$$

Note that if nodal velocities are zero, the two non-orthogonal directions can be changed to any two orthogonal ones with zero velocity and will give the correct results. Having three non-orthogonal directions in 2D is not possible. This analysis is for two directions in 3D and could be extended to three non-orthogonal directions in 3D if ever needed.

2.4 Update Stresses and Strains on Particle

When updating stress and strains before finding forces and updating momenta (only needed for `USAVG`± and for `USF` update methods), extrapolate velocity gradient ($\nabla\boldsymbol{v}$) to the particles and implement the chosen constitutive laws. Because of contact calculations done after mass and momentum extrapolation, the extrapolated velocity gradient will come from velocities that account for contact and grid velocity conditions (but not yet for imperfect interface forces).

2.5 Project Forces to the Grid

For each material point, extrapolate $f_{i,c:m}$ due to particle stresses and $f_{i,T,c:m}$ due to particle traction loads. The values from particle p are added to material velocity field `matfld` of crack velocity field `vfld`, where `vfld` was found during time step initialization and `matfld` is known by the material point's material type.

After the force extrapolations are done, the following calculations complete the force calculations:

1. Restore momentum original extrapolated to the grid in mass and momentum task.

2. Particle force boundary conditions are added for those particles (although these force are actually already in place and were set in the initialization task).

3. If any cracks have traction laws, forces currently stored on crack particles are interpolated to nodes. Note that this extrapolation uses ordinary grid shape functions because crack particles have no volume and GIMP shape function imply a particle domain. Also note that shape function are renormalized if needed to guarantee 100% of force gets to nodes, which occurs if crack particle sees nodes with zero mass.

4. Add any body forces (such as gravity).

Once all above force steps are done, the force on nodes with grid velocity boundary conditions are changed to be consistent with that condition. The updated momentum on a node with prescribed velocity in the $\hat{\boldsymbol{n}}$ direction needs to be:

$$\begin{aligned} \boldsymbol{p}_{i,c:m}^{(k+1)} &= m_{i,c:m} v_i^{(BC)} \hat{\boldsymbol{n}} + \left(\left(\boldsymbol{p}_{i,c:m}^{(k)} + \boldsymbol{f}_{i,c:m}^{(tot)} \Delta t \right) \cdot \hat{\boldsymbol{t}} \right) \hat{\boldsymbol{t}} && (2.7) \\ &= \boldsymbol{p}_{i,c:m}^{(k)} + \left(\boldsymbol{f}_{i,c:m}^{(tot)} + f_{i,c:m}^{(BC)} \hat{\boldsymbol{n}} \right) \Delta t && (2.8) \end{aligned}$$

where $f_{i,c:m}^{(BC)}$ is magnitude of reaction force applied in the $\hat{\boldsymbol{n}}$ direction to satisfy the boundary condition on that node. The normal component of the updated momentum must give the boundary condition or:

$$\boldsymbol{p}_{i,c:m}^{(k+1)} \cdot \hat{\boldsymbol{n}} = m_{i,c:m} v_i^{(BC)} = \boldsymbol{p}_{i,c:m}^{(k)} \cdot \hat{\boldsymbol{n}} + (\boldsymbol{f}_{i,c:m}^{(tot)} \cdot \hat{\boldsymbol{n}}) \Delta t + f_{i,c:m}^{(BC)} \Delta t \tag{2.9}$$

This equation can be solved to find magnitude of the reaction force:

$$f_{i,c:m}^{(BC)} = \frac{m_{i,c:m} v_i^{(BC)} - \boldsymbol{p}_{i,c:m}^{(k)} \cdot \hat{\boldsymbol{n}}}{\Delta t} - \boldsymbol{f}_{i,c:m}^{(tot)} \cdot \hat{\boldsymbol{n}} \tag{2.10}$$

To impose this condition after all other forces are found, revisit all nodes with boundary conditions and add $f_i^{(BC)} \hat{\boldsymbol{n}}$ to nodal force that results in

$$\boldsymbol{f}_{i,c:m}^{(final)} = \boldsymbol{f}_{i,c:m}^{(tot)} - (\boldsymbol{f}_{i,c:m}^{(tot)} \cdot \hat{\boldsymbol{n}}) \hat{\boldsymbol{n}} + \frac{m_{i,c:m} v_i^{(BC)} - \boldsymbol{p}_{i,c:m}^{(k)} \cdot \hat{\boldsymbol{n}}}{\Delta t} \hat{\boldsymbol{n}} \tag{2.11}$$

These calculations are done in two loops over velocity BCs:

1. Zero the initial update component in the $\hat{\boldsymbol{n}}$ direction by setting

$$\boldsymbol{f}_{i,c:m}^{(tot)'} = \boldsymbol{f}_{i,c:m}^{(tot)} - \left((\boldsymbol{f}_{i,c:m}^{(tot)} \cdot \hat{\boldsymbol{n}}) + \frac{\boldsymbol{p}_{i,c:m}^{(k)} \cdot \hat{\boldsymbol{n}}}{\Delta t} \right) \hat{\boldsymbol{n}}$$

   Note that if this called more than once for BC in the same direction, the second call results in

$$\boldsymbol{f}_{i,c:m}^{(tot)''} = \boldsymbol{f}_{i,c:m}^{(tot)'} - \left((\boldsymbol{f}_{i,c:m}^{(tot)'} \cdot \hat{\boldsymbol{n}}) + \frac{\boldsymbol{p}_{i,c:m}^{(k)} \cdot \hat{\boldsymbol{n}}}{\Delta t} \right) \hat{\boldsymbol{n}} = \boldsymbol{f}_{i,c:m}^{(tot)'}$$

   or no additional change. For tracking BC reaction forces, this loop also adds to the above force change to a new reaction forces (which is zeroed before starting this loop. Subsequent calls add zero to the reaction force.

2. Add the BC value by setting

$$\boldsymbol{f}_{i,c:m}^{(final)} = \boldsymbol{f}_{i,c:m}^{(tot)'} + \left(\frac{m_{i,c:m}v_i^{(BC)}}{\Delta t}\right)\hat{\boldsymbol{n}}$$

For tracking BC reaction forces, the same force change is added to the nodal reaction force.

To support superposition of velocity conditions on the same node (but in the same direction $\hat{\boldsymbol{n}}$), the boundary condition value is replaced by

$$m_i^{(n)}v_i^{(BC)} = \sum_k m_i^{(n)}v_i^{(BC,k)} \tag{2.12}$$

where $v_i^{(BC,k)}$ is velocity for boundary condition k. Note that this approach to velocity boundary conditions (rather then just imposing them in the subsequent momentum update) is required to have the accelerations determined from nodal forces be the correct accelerations needed to move the particle from its initial position in the time step to the proper position that reflects the boundary conditions.

A single node can superpose boundary conditions in different directions only if the directions are orthogonal. Two apply two, non-orthogonal conditions, use the method described setion 2.3.2 to convert to orthogonal boundary conditions using $\hat{\boldsymbol{n}}_1$ and $\hat{\boldsymbol{n}}_3$. The final reaction force (which is equal to change in force) will be

$$\begin{aligned}
\boldsymbol{f}_{i,c:m}^{(BC)} &= \frac{m_{i,c:m}v_{i,1}^{(BC)} - \boldsymbol{p}_{i,c:m}^{(k)}\cdot\hat{\boldsymbol{n}}_1}{\Delta t}\hat{\boldsymbol{n}}_1 + \frac{p_3 - \boldsymbol{p}_{i,c:m}^{(k)}\cdot\hat{\boldsymbol{n}}_3}{\Delta t}\hat{\boldsymbol{n}}_3 - \left(\boldsymbol{f}_{i,c:m}^{(tot)}\cdot\hat{\boldsymbol{n}}_1\right)\hat{\boldsymbol{n}}_1 - \left(\boldsymbol{f}_{i,c:m}^{(tot)}\cdot\hat{\boldsymbol{n}}_3\right)\hat{\boldsymbol{n}}_3 \\
&= \frac{m_{i,c:m}v_{i,1}^{(BC)}}{\Delta t}\hat{\boldsymbol{n}}_1 + \frac{p_3}{\Delta t}\hat{\boldsymbol{n}}_3 + \left(\left(\frac{\boldsymbol{p}_{i,c:m}^{(k)}}{\Delta t} + \boldsymbol{f}_{i,c:m}^{(tot)}\right)\cdot\hat{\boldsymbol{n}}_X\right)\hat{\boldsymbol{n}}_X - \left(\frac{\boldsymbol{p}_{i,c:m}^{(k)}}{\Delta t} + \boldsymbol{f}_{i,c:m}^{(tot)}\right) \\
&= \frac{m_{i,c:m}v_{i,1}^{(BC)} - \cos\theta m_{i,c:m}v_{i,2}^{(BC)}}{(1-\cos^2\theta)\Delta t}\hat{\boldsymbol{n}}_1 + \frac{m_{i,c:m}v_{i,2}^{(BC)} - \cos\theta m_{i,c:m}v_{i,1}^{(BC)}}{(1-\cos^2\theta)\Delta t}\hat{\boldsymbol{n}}_2 \\
&\quad + \left(\left(\frac{\boldsymbol{p}_{i,c:m}^{(k)}}{\Delta t} + \boldsymbol{f}_{i,c:m}^{(tot)}\right)\cdot\hat{\boldsymbol{n}}_X\right)\hat{\boldsymbol{n}}_X - \left(\frac{\boldsymbol{p}_{i,c:m}^{(k)}}{\Delta t} + \boldsymbol{f}_{i,c:m}^{(tot)}\right)
\end{aligned}$$

The reaction forces in the controlled directions are

$$\begin{aligned}
\boldsymbol{f}_{i,c:m}^{(BC)}\cdot\hat{\boldsymbol{n}}_1 &= \frac{m_{i,c:m}v_{i,1}^{(BC)}}{\Delta t} - \left(\frac{\boldsymbol{p}_{i,c:m}^{(k)}}{\Delta t} + \boldsymbol{f}_{i,c:m}^{(tot)}\right)\cdot\hat{\boldsymbol{n}}_1 \\
\boldsymbol{f}_{i,c:m}^{(BC)}\cdot\hat{\boldsymbol{n}}_2 &= \frac{m_{i,c:m}v_{i,2}^{(BC)}}{\Delta t} - \left(\frac{\boldsymbol{p}_{i,c:m}^{(k)}}{\Delta t} + \boldsymbol{f}_{i,c:m}^{(tot)}\right)\cdot\hat{\boldsymbol{n}}_2
\end{aligned}$$

2.6 Update Nodal Momenta

This task updates the nodal momenta using

$$\boldsymbol{p}_{i,c:m}^{(k+1)} = \boldsymbol{p}_{i,c:m}^{(k)} + f_{i,c:m}\Delta t \tag{2.13}$$

Now, depending on the problem, this update may cause more contact. Thus the update must be followed by adjusting momenta at all nodes with overlapping materials and at all cracks now in

contact. The logic again is that the final momenta in this step should be identical when doing multiple velocity fields with stick contact to the momenta when using a single velocity field.

The main difference is that interfacial force is added (to both field momenta and field force) when contact done with updated momenta. In addition, the code tracks contact forces (with rigid particles) and friction heating (for frictional contact). These calculations are only done in contact calculations associated with the momentum update.

If contact calculations are done, the velocity boundary conditions are imposed again. If not done, momenta change added in contact calculation might change those conditions. It is easier to reimpose them here than to try and track all times other calculations might change them (and imposing them might eliminate any round off error by finding force that updates to the correct velocity). The BCs, however, can be skipped if there is no contact and no cracks.

When done in this step, the calculation finds the change in momentum to match BCs (see section 2.3.2) but now adds that both to momemtum and adds the change divided by Δt to force. During the tasks, one needs to be sure that momenta and force are consistent such that particle updates will use the correct accelerations.

2.7 Update the Particles

Update particle position and velocity using nodal velocity (from $\boldsymbol{v}_{i,c:m} = \boldsymbol{p}^{(k+1)}_{i,c:m}/m_{i,c:m}$) and acceleration (from $\boldsymbol{a}_{i,c:m} = \boldsymbol{f}_{i,c:m}/m_{i,c:m}$) found in material velocity field `matfld` (for type of particle) of crack velocity field `vfld` (determined on current time step for the particle.

2.8 Optional Update Strain Last

Optionally the calculations can update stain after the momentum update. This calculation is done for methods `USAVG`$\pm$ and for `USL`$\pm$. The - (minus) version updates using the updated velocity and their extrapolated velocity gradient (exactly like the first stress and strain update). The + (plus) method does new extrapolation of momentum to the grid. Because this projection might induce new contact, adjust momenta again at nodes with multiple materials and at crack surfaces. If contact calculations are needed, velocity boundary condition are imposed to insure extrapolated velocity gradients will reflect contact physics. These final set of contact calculations will not include imperfect interface forces of cohesive law forces.

2.9 Move Cracks

The crack surfaces and crack plane are updated using the center-of-mass velocity field of each crack velocity field (for those that have multiple materials). In addition, the new crack opening displacements are used to calculation traction loads to be input to the next time step during its force calculations.

Chapter 3

Finding Normal Vector and Separation For Contact

3.1 Introduction

Getting the normal correct is vital to accurate contact in MPM. Most problems involving contact can be traced to inaccurate normals and normal are prerequisite to all other contact calculatinos. Although dynamic contact detection is advantage of MPM, finding the normal is cost of using that advantage.

3.2 Gradient Methods

Consider material a and b in one crack velocity field on node i. All except material ID in property subscripts will by omitted for clarity. The goal is to find normal $\hat{\boldsymbol{n}}$ directed from a to b. Ignoring b, a unit normal can be estimated from volume gradient:

$$\hat{\boldsymbol{n}} = \frac{\boldsymbol{g}_a}{||\boldsymbol{g}_a||} \tag{3.1}$$

where $\boldsymbol{g}_a$ is extrapolated volume gradient in Eq. (2.6). Note that code should use volume gradient instead of mass gradient to handle situations involving contact between materials with different densities. This normal may or may not be accurate and may or may not be same as corresponding normal found from material b. Some MPM codes found normal for each material and looped over materials. But when normals are not equal and opposite, the momentum would not be conserved. A better approach is to use a single normal for the material pair. No on method works best. The options in `OSParticulas` that use gradient are:

- `AVGGRAD`: Find normal from volume average volume gradients of the two materials:

$$\hat{\boldsymbol{n}}||\hat{\boldsymbol{n}}|| = \boldsymbol{g}_a + \boldsymbol{g}_b \tag{3.2}$$

  The justification is that averaging is often best and volume weighting counts the one with more information (*i.e.* closest to the contact point) the most. The terms are simply added because $\boldsymbol{g}_a$ and $\boldsymbol{g}_b$ are already volume-weighted sums (see Eq. (2.6)).

- `MAXGRAD`: Find normal from materials whose volume gradient has the largest magnitude:

$$\hat{\boldsymbol{n}}||\hat{\boldsymbol{n}}|| = \begin{cases} \boldsymbol{g}_a & \text{if } ||\boldsymbol{g}_\mathrm{a}|| > ||\boldsymbol{g}_\mathrm{b}|| \\ -\boldsymbol{g}_b & \text{otherwise} \end{cases} \tag{3.3}$$

The justification is that one with more information, represented by higher absolute magnitude, might have more accurate information.

- `MAXVOL`: Find normal from materials whose volume has the largest magnitude:

$$\hat{\boldsymbol{n}}||\hat{\boldsymbol{n}}|| = \begin{cases} \boldsymbol{g}_a & \text{if } \Omega_\text{a} > \Omega_\text{b} \\ -\boldsymbol{g}_b & \text{otherwise} \end{cases} \tag{3.4}$$

The justification is that one with more information, represented by more volume, might have more accurate information.

- `OWN`: Find normal from each material and consider contact effects separately for all materials:

$$\hat{\boldsymbol{n}} = \frac{\boldsymbol{g}_a}{||\boldsymbol{g}_a||} \tag{3.5}$$

This method is never recommended and only included in `OSParticulas` to allow comparison to old MPM methods for handling contact.

3.3 Nongradient Methods

Two methods in `OSParticulas` find normal without using the gradient (and therefore need not even extrapolate it for calculations):

- `SPECIFY`: Instead of calculating the normals, prescribe a normal that applies to all contact situations. The normal is directed from the lower number material (in the list of defined materials) to the higher one. This approach is only viable when all contact surfaces have the same normal and the normal does not change during the simulation. This situation is uncommon in real problems. Note that if the contact is with a rigid material (see details below), a specified normal mans from non-rigid into the rigid material, regardless of order of the material numbering.

- `LINREG` and `LOGREG`: These regression methods are based on machine learning methods and find normal by looking at geometry of particle that see each node. Details are given in section 3.4.

3.4 Regression Methods

The plan is to use regression methods to find normal and separation between two material velocity fields in MPM. We start with 2D problems and around some multimaterial node we find N material points with non-zero shape function located at positions $\boldsymbol{x}_i$. Some will be particles for material a and some for material b. We select m_i as indicator of material class for material point i. We seek to minimize the error

$$\Omega = \sum_i w_i (f(\boldsymbol{x}_i, \boldsymbol{\beta}) - m_i)^2 + \sum_j \lambda_j \beta_j^2 \tag{3.6}$$

where $f(\boldsymbol{x}_i, \boldsymbol{\beta})$ is the function chosen for regression analysis, $\boldsymbol{\beta}$ are "n" regression fitting parameters, and λ_j on penalties applied to limit magnitude of selected parameters. In usual non-linear least squares (NLLS) analysis, we start with an initial guess $\boldsymbol{\beta}_0$ and then define updated $\boldsymbol{\beta}^{(k+1)}$ as:

$$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} + \left(\mathbf{J}^T\mathbf{W}\mathbf{J} + \boldsymbol{\Lambda}\right)^{-1} \left(\mathbf{J}^T\mathbf{W}(\boldsymbol{m} - \boldsymbol{f}(\boldsymbol{\beta}_k)) - \boldsymbol{\Lambda}\boldsymbol{\beta}_k\right) \tag{3.7}$$

where $\mathbf{J}$ is $N \times n$ Jacobian of the regression function at current $\boldsymbol{\beta}^{(k)}$:

$$J_{ij} = \frac{\partial f(\boldsymbol{x}_i, \boldsymbol{\beta}^{(k)})}{\partial \beta_j}, \tag{3.8}$$

$\mathbf{W}$ is an $N \times N$ diagonal matrix with w_i on the i^{th} diagonal element, $\boldsymbol{\Lambda}$ is an $n \times n$ diagonal matrix with λ_j penalty for β_j on the j^{th} diagonal element, $\boldsymbol{m}$ is an N-vector of m_i material classes, and $\boldsymbol{f}(\boldsymbol{\beta}_k)$ is an N-vector of $f(\boldsymbol{x}_i, \boldsymbol{\beta})$ values.

3.4.1 Specific Functions

These notes consider three regression functions. For 2D problems, define $\boldsymbol{x}_i = (p_{i,x}, p_{i,y}, 1)$ where $(p_{i,x}, p_{i,y})$ is position of material point i and $\boldsymbol{\beta} = (\beta_1, \beta_2, \beta_3)$. The first function is linear regression:

$$f(\boldsymbol{x}, \boldsymbol{\beta}) = \boldsymbol{x} \cdot \boldsymbol{\beta} \tag{3.9}$$

that defines a line in 2D coordinates with normal direction (β_1, β_2) and β_3 is an offset. This function sets $m_a = -1$ and $m_b = 1$ for material classes. The second is a rescaled logistic function to use with the same material class values:

$$f(\boldsymbol{x}, \boldsymbol{\beta}) = \frac{2}{1 + e^{-\boldsymbol{x}\cdot\boldsymbol{\beta}}} - 1 \tag{3.10}$$

The second is unscaled logistic function

$$f(\boldsymbol{x}, \boldsymbol{\beta}) = \frac{1}{1 + e^{-\boldsymbol{x}\cdot\boldsymbol{\beta}}} \tag{3.11}$$

but this requires changing $m_a = 0$ instead of -1.

The Jacobians for all three function can be written as

$$J_{ij} = \phi(\boldsymbol{x}_i) x_{i,j} \tag{3.12}$$

where $x_{i,j}$ is j^{th} component of $\boldsymbol{x}_i$ and the needed functions are

$$\phi(\boldsymbol{x}) = \begin{cases} 1 & \text{linear} \\ \frac{2e^{-\boldsymbol{x}\cdot\boldsymbol{\beta}}}{\left(1+e^{-\boldsymbol{x}\cdot\boldsymbol{\beta}}\right)^2} & \text{scaled logistic} \\ \frac{e^{-\boldsymbol{x}\cdot\boldsymbol{\beta}}}{\left(1+e^{-\boldsymbol{x}\cdot\boldsymbol{\beta}}\right)^2} & \text{logistic} \end{cases} \tag{3.13}$$

3.4.2 Iteration Equation

We can form $\mathbf{J}^T\mathbf{W}\mathbf{J}$ as a $n \times n$ matrix with k-l element equal to

$$\left(J^T\mathbf{W}\mathbf{J}\right)_{kl} = \sum_i J^T_{ki} w_i J_{il} = \sum_i J_{ik} w_i J_{il} = \sum_i \phi(\boldsymbol{x}_i)^2 w_i x_{i,k} x_{i,l} \tag{3.14}$$

Explicitly in 2D:

$$\left(\mathbf{J}^T\mathbf{W}\mathbf{J} + \boldsymbol{\Lambda}\right) = \begin{bmatrix} \lambda_1 + \sum_i \phi(\boldsymbol{x}_i)^2 w_i x_{i,1}^2 & \sum_i \phi(\boldsymbol{x}_i)^2 w_i x_{i,1} x_{i,2} & \sum_i \phi(\boldsymbol{x}_i)^2 w_i x_{i,1} \\ \sum_i \phi(\boldsymbol{x}_i)^2 w_i x_{i,1} x_{i,2} & \lambda_2 + \sum_i \phi(\boldsymbol{x}_i)^2 w_i x_{i,2}^2 & \sum_i \phi(\boldsymbol{x}_i)^2 w_i x_{i,2} \\ \sum_i \phi(\boldsymbol{x}_i)^2 w_i x_{i,1} & \sum_i \phi(\boldsymbol{x}_i)^2 w_i x_{i,2} & \lambda_3 + \sum_i \phi(\boldsymbol{x}_i)^2 w_i \end{bmatrix} \tag{3.15}$$

which used that $x_{i,3} = 1$. Similarly, the vector term is:

$$\left(\mathbf{J}^T\mathbf{W}(\boldsymbol{m} - \boldsymbol{f}(\boldsymbol{\beta}_k)) - \boldsymbol{\Lambda}\boldsymbol{\beta}_k\right)_j = -\lambda_j \beta_j + \sum_i \phi(\boldsymbol{x}_i) w_i \left(m_i - f(\boldsymbol{x}_i, \boldsymbol{\beta}^{(k)})\right) x_{i,j} \tag{3.16}$$

Extension to 3D just adds entries for z component (by analogy).

Linear Regression

For linear regression, $\phi(\boldsymbol{x}) = 1$ (a constant). If we pick $\beta^{(0)} = 0$ (such that $\lambda_j \beta_j = 0$ and $f(\boldsymbol{x}_i, \boldsymbol{\beta}^{(0)}) = 0$), the solution is

$$\beta^{(1)} = \mathbf{L}^{-1} \boldsymbol{R} \tag{3.17}$$

where $\mathbf{L}$ is given by Eq. (3.15) with $\phi(\boldsymbol{x}_i) = 1$, and

$$\boldsymbol{R} = \left(\sum_i w_i m_i x_{i,1}, \sum_i w_i m_i x_{i,2}, \sum_i w_i m_i \right) \tag{3.18}$$

Substituting into update equation should give $\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)}$:

$$\boldsymbol{\beta}^{(2)} = \boldsymbol{\beta}^{(1)} + \mathbf{L}^{-1} \left(\boldsymbol{R} - \mathbf{J}^T \mathbf{W} \boldsymbol{f}(\boldsymbol{\beta}^{(1)}) - \boldsymbol{\Lambda} \boldsymbol{\beta}^{(1)} \right) \tag{3.19}$$

$$= \boldsymbol{\beta}^{(1)} + \mathbf{L}^{-1} \left(\boldsymbol{R} - \mathbf{L} \boldsymbol{\beta}^{(1)} \right) = \boldsymbol{\beta}^{(1)} \tag{3.20}$$

Logistic Regression

If we start logistic regression with $\beta^{(0)} = 0$, we start with function values of:

$$\phi(\boldsymbol{x}_i) = \begin{cases} \frac{1}{2} & \text{scaled logistic} \\ \frac{1}{4} & \text{logistic} \end{cases} \qquad \text{and} \qquad f(\boldsymbol{x}_i, 0) = \begin{cases} 0 & \text{scaled logistic} \\ \frac{1}{2} & \text{logistic} \end{cases} \tag{3.21}$$

Note that for logistic regression with $m_a = 0$ and $m_b = 1$ that $(m_i - f(\boldsymbol{x}_i, 0)) = -1/2$ for material a and $1/2$ for material b or half of m_i for rescaled logistic regression. As a result, both logisitic methods for the first update lead to

$$\boldsymbol{\beta}^{(1)} = 2 \mathbf{L}_*^{-1} \boldsymbol{R} \tag{3.22}$$

where $\mathbf{L}_*$ is linear regression equation except that λ_j is divided by $\phi(\boldsymbol{x}_i)^2$ (*i.e.*, times 4 for rescaled logistic and times 16 for logistic). Thus, starting with $\beta^{(0)} = 0$ is nearly identical to using linear regression to get the initial guess. If linear regression is used to get the initial guess, it might be better to scale λ in that analysis (*i.e.*, $4\times$ or $16\times$ logistic value) and multiply $\boldsymbol{\beta}$ by 2 to input to nonlinear steps.

3.4.3 Penalty Values

To get idea of how values of λ_j affects the result, consider problem with know solution in Fig. 3.1. This example has material b (black) above the node and material a below the node. The particles are assumed to be at their initial positions at quarter-points in the cells. By symmetry and starting with linear regression, $\mathbf{L}$ is diagonal and only $R_2 \neq 0$. Thus, the solution is $\beta_1 = \beta_3 = 0$ and

$$\beta_2 = \frac{\sum_i w_i m_i x_{i,2}}{\lambda_2 + \sum_i w_i x_{i,2}^2} = \frac{8 \Delta c \left(\frac{1}{4} + \frac{3}{4} \right)}{\lambda_2 + 8 (\Delta c)^2 \left(\frac{1}{16} + \frac{9}{16} \right)} \tag{3.23}$$

$$\frac{\beta_2 \Delta c}{4} = \frac{2}{5 + \frac{\lambda_2}{(\Delta c)^2}} \tag{3.24}$$

where explicit results assumed $w_i = 1$ and Δc is cell size (of square elements). Whatever value is picked for λ_2, it should be proportional to $(\Delta c)^2$ to insure that same relative geometric arrangement of particles will get the same normal independent of the cell size. For example, setting $\Delta c = 2.5$

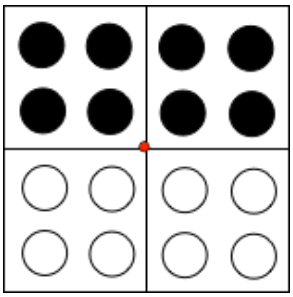

Figure 3.1: Material b (black dots) and material a (white dots) are one node (red dot). The particles are at their initial 1/4 and 3/4 points.

and $\lambda = 1$, the linear regression result is $\beta_2 = 0.620155$ and $\beta_2 \Delta c/4 = 0.3876$. It would change to $\beta_2 \Delta c/4 = 0.4$ if $\lambda_2 \to 0$.

For rescaled logistic and starting with only $\beta_2^{(1)} \neq 0$, $\phi(\boldsymbol{x}_i)$ will be independent of $x_{i,1}$ and an even function of $x_{i,2}$ ($\phi(\boldsymbol{x}_i)^2$ will also be even function of $x_{i,2}$). As a result $\left(\mathbf{J}^T\mathbf{W}\mathbf{J} + \mathbf{\Lambda}\right)$ will be diagonal and the vector $\left(\mathbf{J}^T\mathbf{W}(\boldsymbol{m} - \boldsymbol{f}(\boldsymbol{\beta}_k)) - \mathbf{\Lambda}\boldsymbol{\beta}_k\right)$ will have only the second component nonzero. The problem will converge when that second component is zero or when

$$\lambda_2 \beta_2^{(\infty)} = \sum_i \phi(\boldsymbol{x}_i) w_i \big(m_i - f(\boldsymbol{x}_i, \boldsymbol{\beta}^{(\infty)})\big) x_{i,2} \tag{3.25}$$

$$= 8\Bigg\{ \left(\phi\left(\frac{\Delta c}{4}\right)\left[1 - f\left(\frac{\Delta c}{4}, \beta_2^{(\infty)}\right)\right]\frac{\Delta c}{4} + \phi\left(\frac{3\Delta c}{4}\right)\left[1 - f\left(\frac{3\Delta c}{4}, \beta_2^{(\infty)}\right)\right]\frac{3\Delta c}{4}\Bigg\} \tag{3.26}$$

$$\frac{\lambda_2}{(\Delta c)^2}\frac{\beta_2^{(\infty)}\Delta c}{4} = \frac{2e^{-\beta_2 \Delta c/2}}{(1 + e^{-\beta_2 \Delta c/4})^3} + \frac{6e^{-3\beta_2 \Delta c/2}}{(1 + e^{-3\beta_2 \Delta c/4})^3} \tag{3.27}$$

which assumed $w_i = 1$ and made use of $\big(m_i - f(\boldsymbol{x}_i, \boldsymbol{\beta}^{(\infty)})\big)x_{i,2}$ being even in $x_{i,2}$. For example, setting $\Delta c = 2.5$ and $\lambda = 1$, a graphical solution gives $\beta_2 \Delta c/4 = 0.89305$ or $\beta_2 = 1.4289$. Note that if $\lambda_2 \to 0$, the solution that makes right side equal to zero is when $\beta_2 \to \infty$.

The above result suggest λ_2 should be proportional to $(\Delta c)^2$. Presumably λ_1 should have the same scaling, but what about λ_3 applied to the offset value? Looking at diagonal elements of the key matrix in units of cell size:

$$\lambda_1 + (\Delta c)^2 \sum_i \phi(\boldsymbol{x}_i)^2 w_i x_{i,1}^2, \qquad \lambda_2 + (\Delta c)^2 \sum_i \phi(\boldsymbol{x}_i)^2 w_i x_{i,2}^2, \quad \text{and} \quad \lambda_3 + \sum_i \phi(\boldsymbol{x}_i)^2 w_i \tag{3.28}$$

where $x_{i,j}$ are in units of cell size. These terms suggest λ_1 and λ_2 should be proportional to $(\Delta c)^2$ to give same effect on the diagonal, while λ_3 should be independent of Δc.

3.4.4 Proposed Algorithm

If $\lambda_j \to 0$ and the material points are linearly separable (as in Fig. 3.1), then the converged solution will be normal vector having an infinite magnitude. Even though the magnitude does not converge to a finite value, it seems the direction of the normal vector converges well. Furthermore, using the $\lambda_j \to 0$ also appears to give improved normals. If λ_j is too high, the result converges to an

inaccurate finite value. One is tempted to use $\boldsymbol{\Lambda} = 0$, but some edge results suggest inaccuracies. The first attempt will be to set $\boldsymbol{\Lambda} = (0, 0, \lambda)$ or to add penalty only to the offset (which is term not needed in MPM calculations).

The follow algorithm is proposed:

1. Do one linear regression calculation with $\boldsymbol{\Lambda} = (0, 0, \lambda)$ (a separate loop for efficiency and also build list of m_i to be accessed in logistic regression iterations).
2. Scale linear regression results β by 2 and λ by 0.25 to mimic first pass in logistic regression
3. Iterate logistic regression until $1 - \hat{\boldsymbol{n}}^{(k+1)} \cdot \hat{\boldsymbol{n}}^{(k)} < \epsilon$ where $\hat{\boldsymbol{n}}$ are unit normals calculated from $\boldsymbol{\beta}$ and ϵ is chosen tolerance.

Note that the metric

$$1 - \hat{\boldsymbol{n}}^{(k+1)} \cdot \hat{\boldsymbol{n}}^{(k)} = 1 - \cos(\Delta\theta) \approx \frac{1}{2}(\Delta\theta)^2 \tag{3.29}$$

where $\Delta\theta$ is change in angle of the normal vector from step k and $k+1$. Thus, for desired relative error in $\Delta\theta$ or ϵ_θ, the tolerance ϵ should be set to $2\epsilon_\theta^2$. In other words, tolerance ϵ translates to $\epsilon_\theta = \sqrt{\epsilon/2}$.

Once the normals are found, the next task is find separation between the material points. This method finds the distance of the two closest points of each material type to the line (or plane in 3D) defined by $\boldsymbol{x} \cdot \boldsymbol{\beta} = 0$. In other words, it is looking for separation that would be found by a support vector machine (SVM) assuming SVM and regression are getting the sample hyperplane separating the materials (*i.e.*, normal vector). Given unnormalized $\boldsymbol{\beta}$ from regression analysis, one can find perpendicular distance to the plane for any particle at $\boldsymbol{x}_i$ using

$$t = -\frac{\boldsymbol{x}_i \cdot \beta}{\sqrt{\beta_x^2 + \beta_y^2 + \beta_z^2}} \tag{3.30}$$

where these components of $\boldsymbol{\beta}$ are ones associated with normal vector (2D needs on x and y components). We define the normal and scaled offset from the regression solution as

$$\hat{\boldsymbol{n}} = \begin{cases} \frac{(\beta_1,\beta_2)}{\sqrt{\beta_1^2+\beta_2^2}} & \text{2D} \\ \frac{(\beta_1,\beta_2,\beta_3)}{\sqrt{\beta_1^2+\beta_2^2+\beta_3^2}} & \text{3D} \end{cases} \qquad \text{and} \qquad b = \begin{cases} \frac{\beta_3}{\sqrt{\beta_1^2+\beta_2^2}} & \text{2D} \\ \frac{\beta_4}{\sqrt{\beta_1^2+\beta_2^2+\beta_3^2}} & \text{3D} \end{cases} \tag{3.31}$$

The distance to the line (derived more fully here) is finding intersection of line from particle at $\boldsymbol{p}$ along $\hat{\boldsymbol{n}}$ with the hyperplane solution or solving

$$0 = b + (\boldsymbol{p} + t\hat{\boldsymbol{n}}) \cdot \hat{\boldsymbol{n}} \qquad \text{or} \qquad t = -b - \boldsymbol{p} \cdot \hat{\boldsymbol{n}} \tag{3.32}$$

If b is correct offset for the line between the two materials (it may not be) and the materials are separable, then $t > 0$ in material a (because $\hat{\boldsymbol{n}}$ points from material a to b) and $t < 0$ in material b. The algorithm is thus to find minimum of t in material a and minimum of $-t$ in material b. The separation is the sum of these minima minus twice the particle radius. Because minima will contain equal and opposite contributions of b, that term can be ignored in the distance calculation.

Some possible issues in separation calculation to consider are:

1. All material configurations that are not linearly separable will have negative separation and be assumed to be in contact.
2. Should the radius subtraction be adjusted for deformation of the particles?
3. Should calculations account for direction of the normal within the grid?

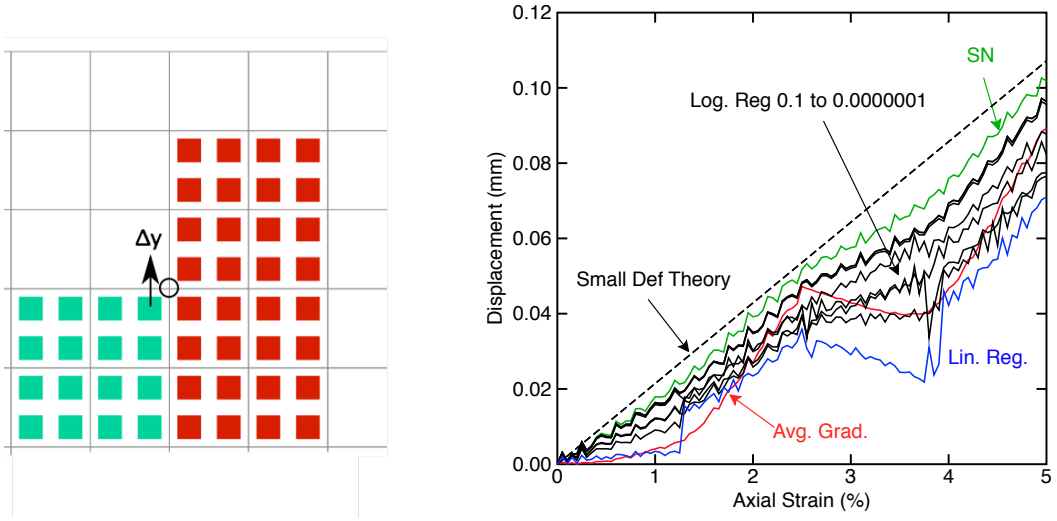

Figure 3.2: Left shows top right corner of block be compressed to the left by rigid particle that extend beyond the block and compress using contact. The circled node is only node that might get normal vector wrong in various normals methods. The right plots displacement, Δy of the indicated particle by linear regression, average gradient, specified normal, and logistic regress as a function of convergence tolerance ϵ.

3.4.5 Sample Calculations

What Tolerance Should be Used

Consider the simple problem of axial compression with rigid materials interacting by contact. Most normals methods will get the correct normal in the problem except for one (or two) node(s) near the edges (as shown in Fig. 3.2). The right half of Fig. 3.2, plots vertical displacement of the indicated particle as a function of axial compression strain. This plane strain calculation predicts that displacement to be:

$$\Delta y = \frac{\nu W}{2(1-\nu)}\varepsilon_{compress} \tag{3.33}$$

where ν is Poisson's ratio and W is total specimen width. These calculations used $\nu = 0.3$ and $W = 10$ mm which should result in plots of Δy *vs.* percent compression having slope of 0.0214286 (dashed line in Fig. 3.2). The "SN" result is method with perfect normals and thus best MPM can do for this corner particle. Note that although this corner particle does not reach the expected displacement, it is displaced the *least* of all particles on the edge. All others are closer to exact answer, but this particle is most affected by contact calculations and boundary condition issues and thus good choice to evaluate different methods of finding contact normal and opening displacement.

The regression calculations here used $\boldsymbol{\Lambda} = 0$ because they this problem did not need a penalty (effect of non-zero penalties are in the next section). The linear regression result is not very accurate. It gets a tilted normal at the corner point that pushes the particle in thereby limiting its displacement. Similarly, the average gradient method is not very good and it is cyclical. The variations are caused by the gradient from the material (which is tilted about 45 degrees) being averaged with the normal from the rigid particle (which will be correct). As the particles move through the mesh, the average gradient calculation will get variable fractions of correct and wrong normals.

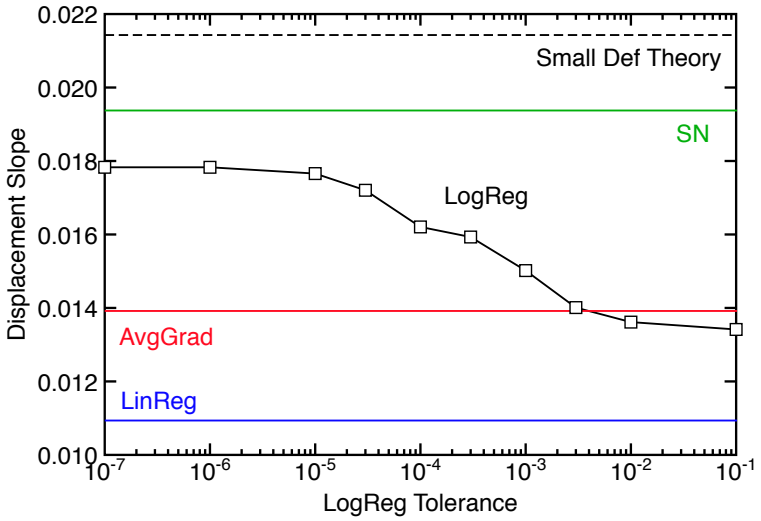

Figure 3.3: Slope of displacement for logistic regression as a function of convergence tolerance ϵ compared to other methods and to the exact solution.

All logistic regressions are better than linear regression (they all get at least one iteration beyond linear regression), are smoother (smaller cyclical effects), and approach perfect normals as the tolerance decreased from 0.1 to 10^{-7}. Figure 3.3 plots the slope of Δy *vs.* percent compression strains as function of ϵ and compares it to other normals methods. The logistic regression improves as ϵ decreases, but shows little improvement for reduction below 10^{-5}. It does not reach the perfect normals result. Note that that 10^{-5} tolerance in above algorithm corresponds to a 0.0022 tolerance in angle change (in radians).

What Offset Penalty Should be Used?

Returning to simple problem of axial compression of two materials in parallel with the interface in the middle showed that using $\boldsymbol{\Lambda} = 0$ degraded the solution compared to results with non-zero penalty. The problem can be traced to poor normals on nodes one cell away from the object that become activated due to Poisson's expansion in compression (see circled nodes in top of Fig. 3.4). Those nodes are low-mass nodes and "see" very few material points (only the edge ones). The logistic regression calculation is poor when $\lambda = 0$, but can be fixed with suitable λ. These low mass nodes will cause very little change in momenta, but apparently enough to affect the results.

The bottom of Fig. 3.4 plots displacement of the indicated particles at end of the calculation as a function of offset penalty λ. The two logistic regression curves are for two different cell sizes (2.5 mm and 0.5 mm). As expected from above, the choice of λ is independent of mesh size. The broad plateau also gives results very close to using specified normal (*i.e.*, perfect calculation of normals). A value in the middle is $\lambda = 5 \times 10^{-4}$. Repeating this calculation for linear regression (and 1 mm cells) gives similar results except the plateau region is shifted to higher λ (a good value in middle of the plateau is $\lambda = 0.1$). Both logisitic and linear regression get very close to perfect normals when λ is chosen correctly. Repeating this calculation with any other normals (such as average gradient) method gives poor results. All results are "off the charts" and therefore not very close to the perfect normals result. The single material mode should match these contact calculations. It is fairly close.

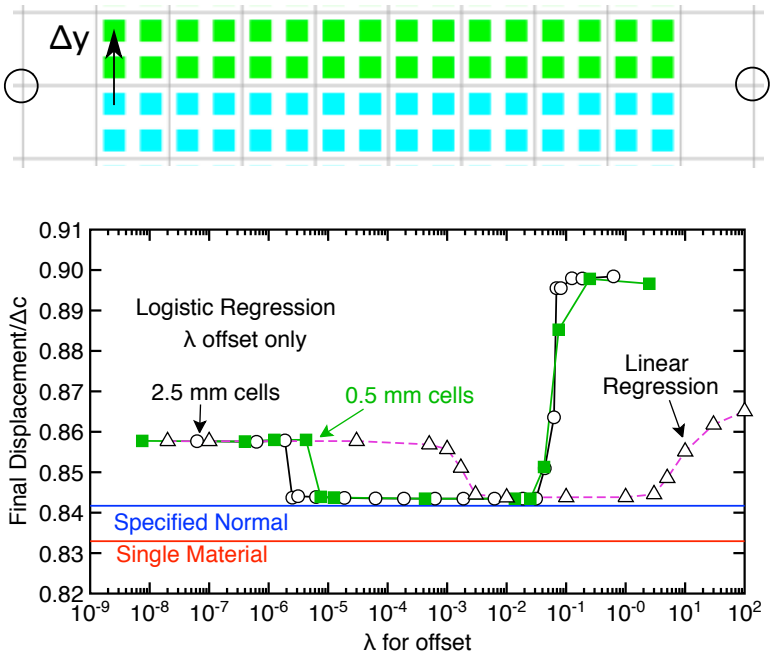

Figure 3.4: A single interface in axial compression (push up) with interface in middle of a large block. The two circled nodes on the top are low-mass modes that become activated due to Poisson's contractions. The plot on bottom is vertical displacement of the edge particles as function of λ applied to offset alone.

3.4.6 Revised Problem

Another approach to logistic regression is to use probability methods and iteratively reweighted least squares (IRLS) method. Perhaps this approach will have different convergence properties? We now seek to minimize the error

$$\Omega = -\sum_i \big(m_i \log f(\boldsymbol{x}_i, \boldsymbol{\beta}) + (1 - m_i) \log(1 - f(\boldsymbol{x}_i, \boldsymbol{\beta}))\big) + \sum_j \lambda_j \beta_j^2 \tag{3.34}$$

where $\boldsymbol{\beta}$ are "n" regression fitting parameters, and λ_j are penalties applied to limit magnitude of selected parameters. The function $f(\boldsymbol{x}_i, \boldsymbol{\beta})$ is the unscaled logistic function or:

$$f(\boldsymbol{x}, \boldsymbol{\beta}) = \frac{1}{1 + e^{-\boldsymbol{x}\cdot\boldsymbol{\beta}}} \tag{3.35}$$

This function requires defining $m_a = 0$ and $m_b = 1$. Using this specific function, Ω simplifies to:

$$\Omega = \sum_i \big(\log(1 + e^{-\boldsymbol{x}\cdot\boldsymbol{\beta}}) + (1 - m_i)\boldsymbol{x}\cdot\boldsymbol{\beta}\big) + \sum_j \lambda_j \beta_j^2 \tag{3.36}$$

The gradient of this function with respect to $\boldsymbol{\beta}$ parameters is

$$\frac{d\Omega}{d\beta_j} = 2\lambda_j\beta_j + \sum_i \big(f(\boldsymbol{x}_i, \boldsymbol{\beta}) - m_i\big)x_{i,j} \quad \text{or} \quad \nabla\Omega = 2\boldsymbol{\Lambda}\boldsymbol{\beta} + \boldsymbol{x}^T\big(f(\boldsymbol{\beta}) - \boldsymbol{m}\big) \tag{3.37}$$

The Hessian matrix ($\mathbf{H} = d\nabla\Omega/d\boldsymbol{\beta}$) in 2D becomes (after dropping 2's from arbitrary penalty terms):

$$\mathbf{H} = \begin{bmatrix} \lambda_1 + \sum_i \phi(\boldsymbol{x}_i) x_{i,1}^2 & \sum_i \phi(\boldsymbol{x}_i) x_{i,1} x_{i,2} & \sum_i \phi(\boldsymbol{x}_i) x_{i,1} \\ \sum_i \phi(\boldsymbol{x}_i) x_{i,1} x_{i,2} & \lambda_2 + \sum_i \phi(\boldsymbol{x}_i) x_{i,2}^2 & \sum_i \phi(\boldsymbol{x}_i) x_{i,2} \\ \sum_i \phi(\boldsymbol{x}_i) x_{i,1} & \sum_i \phi(\boldsymbol{x}_i) x_{i,2} & \lambda_3 + \sum_i \phi(\boldsymbol{x}_i) \end{bmatrix} \tag{3.38}$$

where

$$\phi(\boldsymbol{x}_i) = \frac{e^{-\boldsymbol{x}_i \cdot \boldsymbol{\beta}}}{\left(1 + e^{-\boldsymbol{x}_i \cdot \boldsymbol{\beta}}\right)^2} = f(\boldsymbol{x}_i, \boldsymbol{\beta})\big(1 - f(\boldsymbol{x}_i, \boldsymbol{\beta})\big) \tag{3.39}$$

An iteratively reweighted least squares solution (with 2's removed from penalties) uses:

$$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} - \mathbf{H}^{-1} \nabla\Omega \tag{3.40}$$

$$= \boldsymbol{\beta}^{(k)} + \mathbf{H}^{-1} \left(\boldsymbol{x}^T \big(\boldsymbol{m} - f(\boldsymbol{\beta}) \big) - \boldsymbol{\Lambda}\boldsymbol{\beta} \right) \tag{3.41}$$

Explicitly in 2D:

$$\nabla\Omega = \Big(-\lambda_1\beta_1 + \sum_i \big(m_i - f(\boldsymbol{x}_i, \boldsymbol{\beta})\big) x_{i,1}, -\lambda_2\beta_2 + \sum_i \big(m_i - f(\boldsymbol{x}_i, \boldsymbol{\beta})\big) x_{i,2}, \\ -\lambda_3\beta_3 + \sum_i \big(m_i - f(\boldsymbol{x}_i, \boldsymbol{\beta})\big) \Big) \tag{3.42}$$

The extension of above terms to 3D is obvious.

IRLS Logistic Regression

If we start IRLS logistic regression with $\beta^{(0)} = 0$, we start with function values of:

$$\phi(\boldsymbol{x}_i) = \frac{1}{4} \qquad \text{and} \qquad f(\boldsymbol{x}_i, 0) = \frac{1}{2} \tag{3.43}$$

Note that for for IRLS logistic regression with $m_a = 0$ and $m_b = 1$ that $(m_i - f(\boldsymbol{x}_i, 0)) = -1/2$ for material a and $1/2$ for material b. As a result, IRLS logistic regression for the first update leads to

$$\boldsymbol{\beta}^{(1)} = 2\mathbf{L}_*^{-1}\boldsymbol{R} \tag{3.44}$$

where $\mathbf{L}_*$ is linear regression equation except that λ_j is divided by $\phi(\boldsymbol{x}_i)$ (*i.e.*, times 4). Thus, starting with $\beta^{(0)} = 0$ is nearly identical to using linear regression to get the initial guess. If linear regression is used to get the initial guess, it might be better to scale λ in that analysis (*i.e.*, $4\times$) and multiply multiply $\boldsymbol{\beta}$ by 2 to input to nonlinear steps.

What Tolerance Should be Used

To check for tolerance and if any improvement over previous method, I reran the compression and looked at displacement of corner particle (see Fig. 3.2). The results using IRLS method are in Fig. 3.5. For comparable tolerance, the IRLS method is not as good as the previous NLLS method. Furthermore, the IRLS method fails to converge in the 12 iteration limit more often then NLLS. One curve extends the convergence limit to 30 iterations. It still failed to converge and had very little improvement over the 12 iteration limit. Both logistic regression methods had no penalty on the values, but that did not appear to be needed in this calculation.

In summary, the IRLS and NLLS give similar results, but NLLS converges a little better.

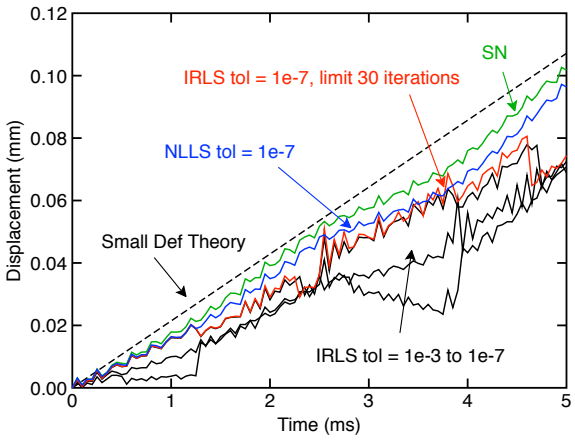

Figure 3.5: Displacement, Δy of the indicated particle in Fig. 3.2 by logistic regression using IRLS or NLLS as a function of convergence tolerance ϵ compared to theory and specified normal

3.5 Bias for Rigid Material Normals

When modeling contact with a rigid material, the particles in the rigid material are not deformed. As a result the volume gradient from the rigid material may be more reliable than the one from the moving non-rigid material. The rigid bias option is to use the normal from the rigid material in contact calculations between non-rigid and rigid materials as the normal vector for contact. To avoid numerical issues for moving rigid materials that might extrapolate very little information to a node, the rigid bias options uses the rigid normal unless its magnitude is some factor less than magnitude of the non-rigid material's volume gradient. This factor is set in the `rigidBias` parameter.

Note that when using `LINREG` and `LOGREG` and if `rigidBias` parameter is set to 10 or higher, the code will get normal from the rigid material gradient instead of by regression methods when contact is with a rigid material. Because of this possibility, the gradients will be extrapolated during calculations.

3.6 Determination of Separation

The use of approach velocity alone (as done in early MPM contact methods) to detect contact is not enough. It often implies contact even when edges are clearly separated. The detection of contact thus requires calculation of the separation between the materials in the normal direction. In physical terms, the momentum change in normal direction of $\Delta \boldsymbol{p}_a \cdot \hat{\boldsymbol{n}} < 0$ corresponds to contact compression if and only if the two materials are in contact. If they are separated, that term corresponds to deceleration need to slow the material fields done as if they were a single material. In other words, contact detection is to find nodes with compression stress between the two materials and that determination needs to know both $\Delta \boldsymbol{p}_a \cdot \hat{\boldsymbol{n}}$ and whether to not the surfaces are actually in contact.

3.6.1 Separation using Regression Methods

The regression methods (`LINREG` and `LOGREG`) find normal vector from particles around a node. Once normal to interface is found, the separation can be found from perpedicular distance of closest

particles to the nodes. This calculation can correct for particle radius and thus is not only the best methods, but only method that works well to very large deformation. The method is described in section 3.4. Note that this calculation does not need extrapolated position or displacement.

3.6.2 Separation using Grid Information

Two approaches are possible when using grid methods (*e.g.* gradient methods or specified normal vectors):

1. Extrapolate displacement to the grid. While this works, it requires that the two materials start out in contact such that zero displacement is equal to contact and that total deformation in the problem is small. When the materials come into contact only after motion (such as closing a gap), displacements are not enough.

2. Extrapolate particle position. While this method is more general, the results have to be correct for grid effects.

When using extrapolated displacement, the interfacial displacement discontinuity on node i (subscript i omitted for clarity) is:

$$\boldsymbol{\delta} = \Delta\boldsymbol{x}_b - \Delta\boldsymbol{x}_a \tag{3.45}$$

where a and b are two materials (for material contact) or above and below a crack. The separation is then given by

$$\delta_n = \boldsymbol{\delta} \cdot \hat{\boldsymbol{n}} \tag{3.46}$$

This $\delta_n \leq 0$ can be used to detect contact, but this method only works when interfaces start out in contact and the overall displacement in the problem remains small.

When extrapolated displacement are not appropriate, one starts by subtracting extrapolated positions:

$$\boldsymbol{\delta} = \Delta\boldsymbol{x}_b - \Delta\boldsymbol{x}_a \tag{3.47}$$

But this result must be corrected. To correct extrapolated positions for material separation calculations we consider 1D extrapolation near the edge of an object. Figure 3.6 shows group of undeformed particles (when using two particle per element in each direction) approaching node i at $x = 0$ with d as the actual distance from the edge of the material points to that node. The "Classic start" shows that position on node i will be zero until $d = -0.75$ or until the center of the first particle enters the element to the left of node i. Both `GIMP` and `CPDI` will start extrapolating to node i the instant the edge enters that element (see "`GIMP` start").

For each d_m, or actual signed distance from edge of material m (a or b) to node i, we can calculate $d_m^{(ext)} = (\boldsymbol{x}_m - \boldsymbol{x}_i) \cdot \hat{\boldsymbol{n}}$, or the distance from extrapolated position $\boldsymbol{x}_m$ to node i using contact normal and a given shape function method. Once these results are found, we invert and the results as shown in Fig. 3.7. The "Left Material" is for edge approaching from the left (if is material a if $\hat{\boldsymbol{n}}$ points to the right). When the particle edge first enters the element to left of node i (assuming `GIMP` or `CPDI`), the extrapolated position (from particle center) will be -1.25 from the node. As the edge approaches, the extrapolation will decrease, but remains negative. When particles of material m completely fill the two elements on opposite sides of the node i, the extrapolated position will be zero. Thus extrapolation values from -1.25 to 0 are mapping to actual edge distances from $d = -1$ to 1. The dotted red lines shows this mapping for `GIMP` (Classic would be piecewise linear approximation starting at (-1,-0.75) and `CPDI` should be similar to `uGIMP`, but neither are shown). The dotted red curve for "Right Material" is the same calculation for edge

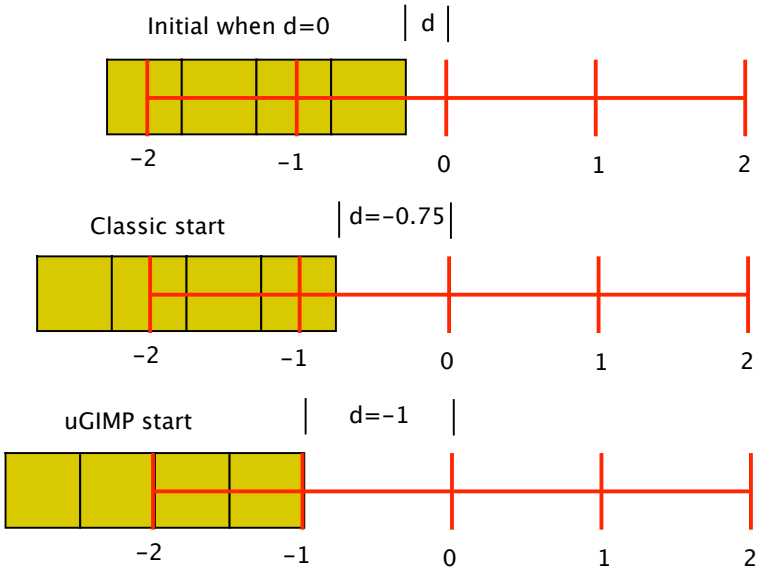

Figure 3.6: One dimensional extrapolation of particle position to grid node. Node i is at $x = 0$ (x in units of cells size). d is distance from edge of material points to node i. The "Classic" and "uGIMP" start are the position the extrapolation first becomes nonzero.

of particles departing to the right (or material b in $\hat{\boldsymbol{n}}$ points to the right). It is always positive and not 0 to 1.25 maps to $d = -1$ to 1. More details are given in supplemental material to our nanoindentation paper [1]

Now imagine two materials (or crack surfaces) near node i. The normal from material a to b is taken in the positive direction, such material a is the "left" material and b is the "Right" material. The separation is written as:

$$\delta_n = d_b(d_b^{(ext)}) - d_a(d_a^{(ext)}) \quad \text{where} \quad d_m^{(ext)} = (\boldsymbol{x}_m - \boldsymbol{x}_i) \cdot \hat{\boldsymbol{n}} \tag{3.48}$$

is the extrapolated (or apparant) distance from edge to node and $d_m(d_m^{(ext)})$ converts extrapolated distance to actual distance. The final δ_n is actual separation and $\delta_n \le 0$ is required for contact.

`OSParticulas` has two methods to find δ_n. The first method (as published in my papers) is to assume the $d_m()$ functions are linear and given by:

$$d_m(d_m^{(ext)}) = \begin{cases} d_a^{(ext)} + 0.4\Delta x & \text{left or } m = a \\ d_b^{(ext)} - 0.4\Delta x & \text{right or } m = b \end{cases} \tag{3.49}$$

where Δx is an effective cell size (and equal to cell size with square (2D) or cubic (3D) cells, but changed to $h_\perp$ for rectangular cells and changed for arbitrary neighbor sizes for a Tartan grid). With this equation, the material separation becomes

$$\delta_n = (\boldsymbol{x}_b - \boldsymbol{x}_i) \cdot \hat{\boldsymbol{n}} - 0.4\Delta x - \big((\boldsymbol{x}_a - \boldsymbol{x}_i) \cdot \hat{\boldsymbol{n}} + 0.4\Delta x\big) = \boldsymbol{\delta} \cdot \hat{\boldsymbol{n}} - 0.8\Delta x \tag{3.50}$$

My paper describes this approach as subtracting a contact constant, but the net effect of this approach is same as assuming the above linear conversion form $d_m^{(ext)}$ to d_m. This net subtraction of $0.8\Delta x$ was found by a weighted average of $d_b^{(ext)} - d_a^{(ext)}$ when edges are in contact and counted

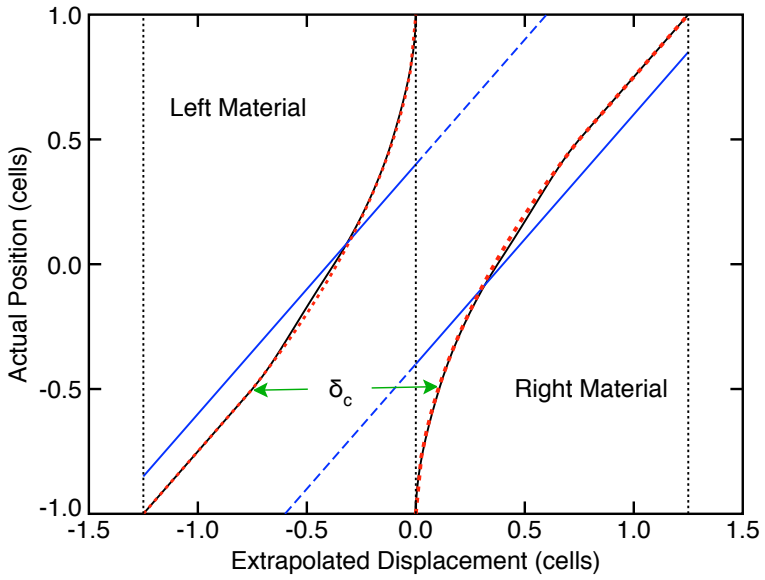

Figure 3.7: Plot of actual distance from edge to node i at $x = 0$ (d) as a function of extrapolated position (d_{ext} on node i when using one particle per cell. The "Left Material" if for edge approaching from the left. The "Right Material" is for an edge departing to the right.

the results near $d = 0$ more. The weighting logic was that momentum change is greatest when d is close to zero or contact is near a node. Although the subtraction value can be varied to any value, it appears results far from $0.8\Delta x$ cells give poor results or that $0.8\Delta x$ cells appears to be universal value when using this linear method and when have two particles per cell.

An alternate approach is to fit the inverted curves derived from shape function calculations. The solid black lines show the fits:

$$\frac{d_m(d_m^{(ext)})}{\Delta x} = \begin{cases} 1 - 2\left(\frac{-d_a^{(ext)}}{1.25\Delta x}\right)^{0.58} & \text{left or } m = a \\ 2\left(\frac{d_b^{(ext)}}{1.25\Delta x}\right)^{0.58} - 1 & \text{right or } m = b \end{cases} \tag{3.51}$$

Thus in contact calculations, one finds $d_m(d_m^{(ext)})$ for left (a) and right (b) materials using this equation, and then subtracts for the actual δ_n or surface separation. Contact occurs when $\delta_n \leq 0$. This option is selected in `OSParticulas` by setting the `ContactPosition` to a negative number and the absolute value of that number becomes the exponent in the above fits. Like 0.8 method, the exponent can be varied, but it appears that -0.58 (in the command, which uses exponent of 0.58) is universal value that should not be adjusted.

A slightly better fit can be found be recognizing that the shape function calculation is exactly linear over a small region. Using piecewise function with part linear and the rest a power law gives:

$$\frac{d_m(d_m^{(ext)})}{\Delta x} = \begin{cases} \begin{cases} \frac{d_a^{(ext)}}{\Delta x} + 0.25 & d_a^{(ext)} < -0.75\Delta x \\ 1 - \frac{3}{2}\left(\frac{-4d_a^{(ext)}}{3\Delta x}\right)^{0.5546} & d_a^{(ext)} > -0.75\Delta x \end{cases} & \text{left or } m = a \\ \begin{cases} \frac{3}{2}\left(\frac{4d_b^{(ext)}}{3\Delta x}\right)^{0.5546} - 1 & d_b^{(ext)} < 0.75\Delta x \\ \frac{d_b^{(ext)}}{\Delta x} - 0.25 & d_b^{(ext)} > 0.75\Delta x \end{cases} & \text{right or } m = b \end{cases} \tag{3.52}$$

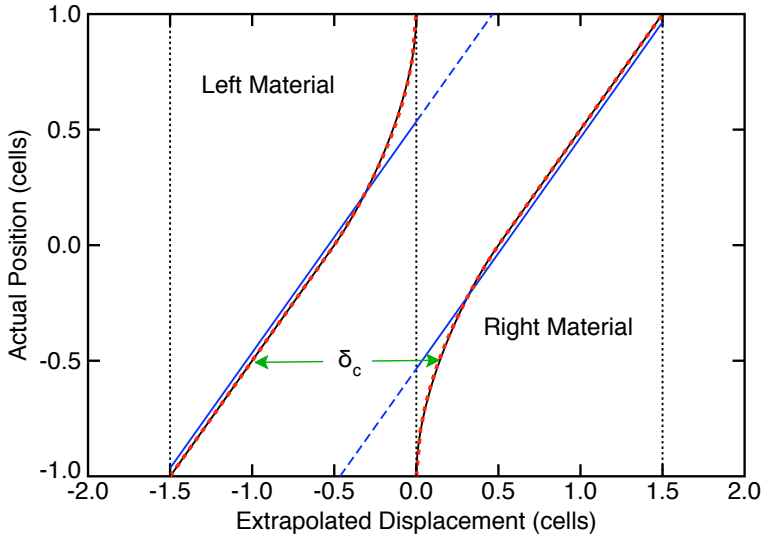

Figure 3.8: Plot of actual distance from edge to node i at $x = 0$ (d) as a function of extrapolated position (d_{ext} on node i when using one particle per cell. The "Left Material" if for edge approaching from the left. The "Right Material" is for an edge departing to the right.

The difference with first power law in Eq. (3.51) is small. This revised form is not yet implemented in code.

The above calculations assume two particles per cell. A displacement criterion to detect contact, however, depends on the number of particles. For one particle per cell, the linear method changes to:

$$d_m(d_m^{(ext)}) = \begin{cases} d_a^{(ext)} + 0.535\Delta x & \text{left or } m = a \\ d_b^{(ext)} - 0.535\Delta x & \text{right or } m = b \end{cases} \tag{3.53}$$

or

$$\delta_n = \boldsymbol{\delta} \cdot \hat{\boldsymbol{n}} - 1.07\Delta x \tag{3.54}$$

A power law fit can also be derived. For one particle, a significant portion of the curve is exactly linear. A piecewise fit with a linear portion and power law portion is given by:

$$\frac{d_m(d_m^{(ext)})}{\Delta x} = \begin{cases} \begin{cases} \frac{d_a^{(ext)}}{\Delta x} + 0.5 & d_a^{(ext)} < -0.5\Delta x \\ 1 - \left(\frac{-2d_a^{(ext)}}{\Delta x}\right)^{0.5387} & d_a^{(ext)} > -0.5\Delta x \end{cases} & \text{left or } m = a \\ \begin{cases} \left(\frac{2d_b^{(ext)}}{\Delta x}\right)^{0.5387} - 1 & d_b^{(ext)} < 0.5\Delta x \\ \frac{d_b^{(ext)}}{\Delta x} - 0.5 & d_b^{(ext)} > 0.5\Delta x \end{cases} & \text{right or } m = b \end{cases} \tag{3.55}$$

A plot the linear and power-law functions for one particle per cell is in Fig. 3.8

A final addition is needed when δ_n is found after the momentum update. Because δ_n is found from initially extrapolated positions, it should be updated for interfacial movement during the time step. The correction is

$$\delta_n' = \boldsymbol{\delta}_n + (\boldsymbol{v}_b - \boldsymbol{v}_a) \cdot \hat{\boldsymbol{n}} \Delta t = \boldsymbol{\delta}_n + \frac{\Delta \boldsymbol{p}_a \cdot \hat{\boldsymbol{n}} \Delta t}{m_{red}} \tag{3.56}$$

In summary, the three extrapolation options when using grid information in `OSParticulas` are:

1. No `ContactPosition` command — if contact position is never set, contact is determined by extrapolated displacements and no correction is applied to the extrapolated values. (Note: this is **not** the same as setting contact position to 0, which is to extrapolate position instead of displacement but not correct the values; these results would be very bad).

2. Positive `ContactPosition` — contact is determined by extrapolated positions which are adjusted by constant value provided (should probably always be 0.8 for two particles per cell or 1.07 for two.)

3. Negative `ContactPosition` — contact is determined by extrapolated positions which are adjusted by Eq. (3.51) where exponent is absolute value of value provided (should probably always be -0.58). This option is for two particle only. The piecewise curves for two or one particle per cell are not yet implemented.

Chapter 4

Contact Calculations

4.1 Detecting Contact

The next task is to decide if materials are in contact or if they are separated. When in contact, the materials are often left to move in the separate velocity fields. When they are in contact, however, the momenta have to change to prevent interpenetration and to implement contact mechanics. If the contact law includes adhesion, it may be necessary to examine separated material as well to decided if they have overcome some input adhesion value. For imperfect interface, forces generally need to be added for both contacted and separated interfaces. All cases need to know which has occurred. Below are the following steps used in `OSParticulas`.

First, a mass-prescreening option is used: The final change in momentum is related to reduced mass on the node, which means the change will be neglible if the mass ratio is high or small. This prescreen skips contact calculations if:

$$\frac{m_a}{m_{tot}} < \epsilon \qquad \text{or} \qquad \frac{m_a}{m_{tot}} > 1 - \epsilon \tag{4.1}$$

where ϵ is a small number. The current version of `OSParticulas` uses $\epsilon = 1.e^{-5}$ (might change in the future).

A discarded volume prescreening test used to ignore nodes where total volume on a node (sum of material volumes) is compared to total volume associated with that node (*i.e.*, cell volume — for grid with unequal element sizes, it is net volume of cells around node i). Let V_r be ratio of material volumes to node i cell volume. This prescreen will ignore contact on this node if $V_r < V_{min}$ where V_{min} is simulations parameter. The logic is not that contact at interfaces are not present, but rather that this node is likely on the edge of a material and the calculations will be inaccurate. It might better to skip them to to attempt them. This methods was never found to be effective and has now been removed from `OSParticulas`.

If the calculations are nor screened out, the normal is calculated (as explained above) and then contact is determined by one or both of the following checks — an approach velocity check and a interface separation check. Early MPM work used the first method only, but current thinking is that second is always needed. The option to skip the second check has been removed.

4.1.1 Approach Velocity

For two materials with velocities $\boldsymbol{v}_a$ and $\boldsymbol{v}_b$, the center of mass velocity is

$$\boldsymbol{v}_c = \frac{m_a \boldsymbol{v}_a + m_b \boldsymbol{v}_b}{m_a + m_b} \tag{4.2}$$

where m_a and m_b are the masses for the two materials. Next, define $\Delta \boldsymbol{p}_a$ as the momentum change required on material a for its velocity to change from $\boldsymbol{v}_a$ to $\boldsymbol{v}_c$ or

$$\boldsymbol{v}_a + \frac{\Delta \boldsymbol{p}_a}{m_a} = \boldsymbol{v}_c \quad \text{which means} \quad \Delta \boldsymbol{p}_a = -m_a(\boldsymbol{v}_a - \boldsymbol{v}_c) \tag{4.3}$$

Inserting the definition of $\boldsymbol{v}_c$ leads to

$$\Delta \boldsymbol{p}_a = -m_a(\boldsymbol{v}_a - \boldsymbol{v}_c) = m_b(\boldsymbol{v}_b - \boldsymbol{v}_c) = \frac{m_a \boldsymbol{p}_b - m_b \boldsymbol{p}_a}{m_a + m_b} \tag{4.4}$$

where $\boldsymbol{p}_a$ and $\boldsymbol{p}_b$ are the nodal momenta for the two materials. Notice that

$$\Delta \boldsymbol{v} = \boldsymbol{v}_b - \boldsymbol{v}_a = \frac{\Delta \boldsymbol{p}_a}{m_b} + \frac{\Delta \boldsymbol{p}_a}{m_a} = \frac{\Delta \boldsymbol{p}_a}{m_{red}} \tag{4.5}$$

where $m_{red} = m_a m_b/(m_a+m_b)$ is the reduced nodal mass. Thus the relative velocity of approach of the two surfaces is proportional to, and in the same direction as, $\Delta \boldsymbol{p}_a$. For detecting contact based on approach velocity, the two surfaces must be approaching each other. With a normal directed from a to b, this criteria implies $\Delta \boldsymbol{v} \cdot \hat{\boldsymbol{n}} < 0$ or identically that $\Delta \boldsymbol{p}_a \cdot \hat{\boldsymbol{n}} < 0$.

4.1.2 Displacement Check

Many contact laws (such as friction) are based on compression state. With the above velocity check, those contact laws can assume that $\Delta \boldsymbol{p}_a \cdot \hat{\boldsymbol{n}} < 0$ in their calculations. But, as discussed in section 3.6, this condition is not sufficient to say the interface is actually in contact and in compression. Instead, contact should mean both $\Delta \boldsymbol{p}_a \cdot \hat{\boldsymbol{n}} < 0$ and $\boldsymbol{\delta} \cdot \hat{\boldsymbol{n}} < 0$ where the later is found by methods in section 3.6.

If displacement calculations alone are accurate, one is tempted to use displacement or contact detected whenever $\boldsymbol{\delta} \cdot \hat{\boldsymbol{n}} < 0$. This option is available, but is not recommended. It has helped in simulations that are always, or almost always, in contact, but does not work well otherwise.

4.2 Changes in Momentum for Contact

The relatively easy part s(compared to finding normal and detecting contact) are changing momenta and finding proper force needed to enforce a contact law. The basic analysis is for material a and contact with material b, but to handle general problems where a node may see more that two materials (although it may not be handled rigorously), the following algorithm is used. At a node with any number of materials, the center of mass results are:

$$\boldsymbol{p}_c = \sum_i \boldsymbol{p}_i \qquad m_c = \sum_i m_i \qquad \boldsymbol{v}_c = \frac{\boldsymbol{p}_c}{m_c} \tag{4.6}$$

We define a *virtual* material b derived from all mass and momenta not associated with material a:

$$\boldsymbol{p}_b = \boldsymbol{p}_c - \boldsymbol{p}_a \qquad m_b = m_c - m_a \qquad \boldsymbol{v}_b = \frac{\boldsymbol{p}_b}{m_b} \tag{4.7}$$

The equations for $\Delta \boldsymbol{p}_a$ and $\Delta \boldsymbol{v}$ from above are unchanged, but in terms of just a and c, they become

$$\Delta \boldsymbol{p}_a = \frac{m_a}{m_c}\boldsymbol{p}_c - \boldsymbol{p}_a, \qquad \Delta \boldsymbol{v} = \frac{\Delta \boldsymbol{p}_a}{m_{red}}, \qquad \text{and} \qquad m_{red} = \frac{m_a(m_c - m_a)}{m_c} \tag{4.8}$$

All these results reduces exactly to two-material case for nodes with two materials.

To implement contact conditions, with inclusion of nodes with three or more materials, each material is considered relative to the center of mass velocity and $\Delta \boldsymbol{p}_a$ and $\Delta \boldsymbol{v}$ are found. Stick conditions are trival. The momentum change of $\Delta \boldsymbol{p}_a$ is applied to material a. These momenta changes correspond to normal and tangential forces of

$$\boldsymbol{f}_n = \frac{(\Delta \boldsymbol{p}_a \cdot \hat{\boldsymbol{n}})}{\Delta t}\hat{\boldsymbol{n}} \qquad \text{and} \qquad \boldsymbol{f}_t = \frac{(\Delta \boldsymbol{p}_a \cdot \hat{\boldsymbol{t}})}{\Delta t}\hat{\boldsymbol{t}} \tag{4.9}$$

where Δt is time step and $\hat{\boldsymbol{t}}$ is unit vector in the tangential direction of motion. We choose $\hat{\boldsymbol{t}}$ to be in the same direction as the relative motion in the tangential direction such that $(\boldsymbol{v}_b - \boldsymbol{v}_a) \cdot \hat{\boldsymbol{t}} > 0$ and identically such that $\Delta \boldsymbol{p}_a \cdot \hat{\boldsymbol{t}} > 0$ or that the angle between $\hat{\boldsymbol{t}}$ and $\boldsymbol{v}_b - \boldsymbol{v}_a$ is between -90 and 90 degrees. Give $\hat{\boldsymbol{n}}$, $\hat{\boldsymbol{t}}$ can be found from

$$\hat{\boldsymbol{t}} = \frac{\Delta \boldsymbol{p}_a - (\Delta \boldsymbol{p}_a \cdot \hat{\boldsymbol{n}})\hat{\boldsymbol{n}}}{||\Delta \boldsymbol{p}_a - (\Delta \boldsymbol{p}_a \cdot \hat{\boldsymbol{n}})\hat{\boldsymbol{n}}||} \tag{4.10}$$

The positive normal contact force is thus $f_n = -(\Delta \boldsymbol{p}_a \cdot \hat{\boldsymbol{n}})/\Delta t$ and the positive sliding force for interfaces to stick (move in same velocity field) is $f_t = (\Delta \boldsymbol{p}_a \cdot \hat{\boldsymbol{t}})/\Delta t$. We write the normal pressure (N) and sticking traction (S_{stick}) as

$$N = -\frac{\Delta \boldsymbol{p}_a \cdot \hat{\boldsymbol{n}}}{A_c \Delta t} \qquad \text{and} \qquad S_{stick} = \frac{\Delta \boldsymbol{p}_a \cdot \hat{\boldsymbol{t}}}{A_c \Delta t} \tag{4.11}$$

where A_c is contact area (calculated by methods in [2]). Implementation of general contact laws for sliding force S_{slide} as a function of normal pressure N is done as follows:

1. If $S_{stick} < S_{slide}$, then the driving forces for frictional sliding are too small to overcome sticking meaning the two materials stick rather than slide. Set final momentum change to $\Delta \boldsymbol{p}'_a = \Delta \boldsymbol{p}_a$.

2. But, if $S_{stick} > S_{slide}$, the surface would rather slide at the lower S_{slide} then stick. Set final momentum change to
$$\Delta \boldsymbol{p}'_a = d_n \hat{\boldsymbol{n}} + (S_{slide} A_c \Delta t)\hat{\boldsymbol{t}} \tag{4.12}$$
where $d_n = \Delta \boldsymbol{p}_a \cdot \hat{\boldsymbol{n}} = -f_n \Delta t$

3. Change the momenta for materials a and b to
$$\boldsymbol{p}'_a = \boldsymbol{p}_a + \Delta \boldsymbol{p}'_a \text{ and } \boldsymbol{p}'_b = \boldsymbol{p}_b - \Delta \boldsymbol{p}'_a \tag{4.13}$$
A subtle point, which is only found by requiring two identical materials in contact at a perfect interface to match single material mode (see below), is that when consistent forces are found for grid boundary conditions, they must be based on $\boldsymbol{p}'_a$ and not on the originally extrapolated $\boldsymbol{p}_a$. The forces are found from stresses in materials that are based on contact conditions being satisfied and therefore boundary conditions force must do the same.

4. When momentum change is found *after* finding forces and updating momenta, the updated foces need to be kept with changed momenta. The nodal forces should be changed to:
$$\boldsymbol{f}'_a = \boldsymbol{f}_a + \frac{\Delta \boldsymbol{p}_a^{+\prime}}{\Delta t} \tag{4.14}$$
where $\Delta \boldsymbol{p}_a^{+\prime}$ is momentum change in contact calculations using updated momenta.

5. If three or more materials are present, some modifications are needed (see below).

See our paper on contact laws for more details and handling various types of contact laws (such as those with velocity dependent laws or with adhesion). When frictional sliding occurs, the energy dissipated can be found from frictional work and input to heat conduction calculations. This option is also described in generalized contact paper [3].

4.2.1 Material Velocity Field Updates

When the calculations update momenta, $\boldsymbol{p}_a$ will have changed to $\boldsymbol{p}'_a$ (similar for material b) and $\boldsymbol{f}_a$ will have been extrapolated to the grid; the update will be:

$$\boldsymbol{p}_a^+ = \boldsymbol{p}'_a + \boldsymbol{f}_a \Delta t \tag{4.15}$$

Here $\boldsymbol{f}_a$ are the forces extrapolated from particle stresses that are found when interfaces satisfy boundary conditions. After updating momentum, new contact calculations will find $\Delta p_a^{+'}$ meaning that the final particle momentum and force are:

$$\boldsymbol{p}_a^{+'} = \boldsymbol{p}_a^+ + \Delta p_a^{+'} \quad \text{and} \quad \boldsymbol{f}'_a = \boldsymbol{f}_a + \frac{\Delta \boldsymbol{p}_a^{+'}}{\Delta t} \tag{4.16}$$

The total particle update becomes

$$\boldsymbol{p}_a^{+'} = \boldsymbol{p}'_a + \boldsymbol{f}'_a \Delta t = \boldsymbol{p}'_a + \left(\boldsymbol{f}_a + \frac{\Delta \boldsymbol{p}_a^{+'}}{\Delta t} \right) \Delta t \tag{4.17}$$

The final velocity and acceleration for material a are:

$$\boldsymbol{v}_a^+ = \frac{\boldsymbol{p}_a^{+'}}{m_a} \quad \text{and} \quad \boldsymbol{a}_a^+ = \frac{\boldsymbol{f}'_a}{m_a} \tag{4.18}$$

To validate this process, show that for two identical materials with a perfect interface translates into material velocity and acceleration equal to center of mass velocity and acceleration. When two identical materials are in contact and the interface is perfect, we know that

$$\boldsymbol{p}'_a = \frac{m_a}{m_i} \boldsymbol{p}_i \quad \text{and} \quad \boldsymbol{p}_a^{+'} = \frac{m_a}{m_i} (\boldsymbol{p}_i + \boldsymbol{f}_i \Delta t), \tag{4.19}$$

and

$$\Delta \boldsymbol{p}_a^{+'} = \boldsymbol{p}_a^{+'} - \boldsymbol{p}_a^+ = \frac{m_a}{m_i} (\boldsymbol{p}_i + \boldsymbol{f}_i \Delta t) - \left(\boldsymbol{p}'_a + \boldsymbol{f}_a \Delta t \right) = \frac{m_a}{m_i} \boldsymbol{f}_i \Delta t - \boldsymbol{f}_a \Delta t \tag{4.20}$$

Solving for forces gives:

$$\frac{\boldsymbol{f}_i}{m_i} = \frac{1}{m_a} \left(\boldsymbol{f}_a + \frac{\Delta \boldsymbol{p}_a^{+'}}{\Delta t} \right) \tag{4.21}$$

The end result is that

$$\boldsymbol{v}_a^+ = \frac{\boldsymbol{p}_a^{+'}}{m_a} = \frac{1}{m_i} (\boldsymbol{p}_i + \boldsymbol{f}_i \Delta t) \quad \text{and} \quad \boldsymbol{a}_a^+ = \frac{\boldsymbol{f}'_a}{m_a} = \frac{\boldsymbol{f}_i}{m_i} \tag{4.22}$$

In other words the material velocity and acceleration both match the single-material mode velocity and acceleration. Any other handing of momenta and force would cause a mismatch between velocity and/or acceleration.

4.2.2 Force-Based Contact Calculations

Some early MPM contact claims improved results by looking at contact force rather then a momentum change, but the approach can be shown identical to above momentum change as long as one is careful to synchronize changes in momentum with changes in force. The approach is to consider velocity updates of the two velocity fields as

$$\boldsymbol{v}_a = \frac{\boldsymbol{p}_a + (\boldsymbol{f}_a + \boldsymbol{f}_c)\Delta t}{m_a} \quad \text{and} \quad \boldsymbol{v}_b = \frac{\boldsymbol{p}_b + (\boldsymbol{f}_b - \boldsymbol{f}_c)\Delta t}{m_b} \tag{4.23}$$

where $\boldsymbol{f}_c$ is contact force needed to impose contact mechanics. Equating the final velocities in the normal direction (same equations apply if equating velocities in tangential direction or in any direction defined by a unit vector) is easily solved to be:

$$\boldsymbol{f}_c = \left(\frac{m_a(\boldsymbol{p}_b + \boldsymbol{f}_b\Delta t) - m_b(\boldsymbol{p}_a + \boldsymbol{f}_a\Delta t)}{(m_a + m_b)\Delta t} \cdot \hat{\boldsymbol{n}} \right) \hat{\boldsymbol{n}} \tag{4.24}$$

$$\boldsymbol{f}_c = \left(\frac{\Delta \boldsymbol{p}_a}{\Delta t} \cdot \hat{\boldsymbol{n}} + \Delta \boldsymbol{f}_a \cdot \hat{\boldsymbol{n}} \right) \hat{\boldsymbol{n}} \tag{4.25}$$

But realizing that updated momentum before this calculation (*e.g.*, $\boldsymbol{p}'_a = \boldsymbol{p}_a + \boldsymbol{f}_a\Delta t$), this result can be written as:

$$\boldsymbol{f}_c\Delta t = \left(\frac{m_a\boldsymbol{p}'_b - m_b\boldsymbol{p}'_a}{(m_a + m_b)} \cdot \hat{\boldsymbol{n}} \right) \hat{\boldsymbol{n}} = (\Delta \boldsymbol{p}_a \cdot \hat{\boldsymbol{n}})\, \hat{\boldsymbol{n}} \tag{4.26}$$

where last form is using the updated momenta. This result is identical to the momentum method in previous section for stick in normal direction if you correctly add to nodal force as well as nodal momentum.

4.2.3 Low Mass Situations

The above algorithm never divides by a possibly low mass, but that does not necessarily mean it is not affected by one of the masses being very low. In GIMP calculations it will be common for a node one element away from the contact region to have nearly all mass in one material and a very low mass in the other material. When that occurs:

$$\text{if } m_a << m_b, \text{ then } \Delta\boldsymbol{p}_a \approx m_a\boldsymbol{v}_b - \boldsymbol{p}_a \quad \text{or} \quad \text{if } m_b << m_a, \text{ then } \Delta\boldsymbol{p}_a \approx \boldsymbol{p}_b - m_b\boldsymbol{v}_a \tag{4.27}$$

In other words, $\Delta\boldsymbol{p}_a$ will be based on the unreliable nodal momentum (since it is likely to be from one particle on the edge of a material). Since $\Delta\boldsymbol{p}_a$ is used to determine contact and to adjust the momenta, the entire algorithm will be unreliable. Now, since $\Delta\boldsymbol{p}_a$ will also be very small, it might not make a difference, but in one calculation sliding down an incline plane, the calculation stopped when accepting all contact situations and was fixed by screening out those with $m_a/(m_a + m_n)$ or $m_b/(m_a + m_n)$ less than 10^{-5} (might change in the future). Thus in step one above, if the mass ratio meets these criteria, ignore contact for that node.

4.2.4 Rigid Material Contact

Specify normals changes and add rigid bias option.

If material b is rigid, contact can be handled by a special case of the two-material contact, A rigid material corresponds to $m_b \to \infty$. The basic contact equations change to

$$\begin{aligned} \boldsymbol{v}_c &= \boldsymbol{v}_b && (4.28)\\ \Delta\boldsymbol{p}_a &= -m_a(\boldsymbol{v}_a - \boldsymbol{v}_b) && (4.29)\\ \Delta\boldsymbol{v} &= \boldsymbol{v}_b - \boldsymbol{v}_a = \frac{\Delta\boldsymbol{p}_a}{m_a} && (4.30)\\ m_{red} &= m_a && (4.31) \end{aligned}$$

After these changes, the contact methods above can be applied the same way except the momentum of the rigid particle is not changed. In `NairnMPM`, the density of rigid materials is set to 1000 g/cm^3 such that material point mass found in `NairnMPM::PreliminaryParticleCalcs()` is equal to the volume for the rigid material point in mm^3. Thus the mass gradient needed in the contact algorithm is equal to the volume gradient. The nodal mass, however, for material velocity field of rigid particles is set to zero as a way to determine a rigid particle velocity field. The nodal `vk` is always the correct velocity and is calculated when the first rigid particle is extrapolated to that node.

4.2.5 Three of More Materials

Several options. Lumping seems best. Need more detail.

Chapter 5

Imperfect Interface Calculations

5.1 Interface Forces

The displacement at node i for materials a and b in crack field c after current time step can be written as:

$$\boldsymbol{u}_a(\Delta t) = \boldsymbol{u}_a(0) + \frac{\boldsymbol{p}_a \Delta t}{m_a} + \frac{(\boldsymbol{f}_a + \boldsymbol{f}_a^{(INT)})(\Delta t)^2}{2m_a} \tag{5.1}$$

$$\boldsymbol{u}_b(\Delta t) = \boldsymbol{u}_b(0) + \frac{\boldsymbol{p}_b \Delta t}{m_b} + \frac{(\boldsymbol{f}_b - \boldsymbol{f}_a^{(INT)})(\Delta t)^2}{2m_b} \tag{5.2}$$

where node i and crack field c have been omitted (for clarity) and $\boldsymbol{f}_a^{(INT)}$ is force that must be added to a and substracted from b to impose interface conditions. Here $\boldsymbol{f}_a$ is force found in the current time step. When interface calculations are done after the momentum update, the initial momentum needed in above expression is found from updated momentum ($\boldsymbol{p}'_a$) using:

$$\boldsymbol{p}_a = \boldsymbol{p}'_a - \boldsymbol{f}_a \Delta t \qquad \text{and} \qquad \boldsymbol{p}_b = \boldsymbol{p}'_b - \boldsymbol{f}_b \Delta t \tag{5.3}$$

The same holds for material b. Alternatively, displacement could be written in terms of fully updated momenta as:

$$\boldsymbol{u}_a(\Delta t) = \boldsymbol{u}_a(0) + \frac{\boldsymbol{p}''_a \Delta t}{m_a} - \frac{(\boldsymbol{f}_a + \boldsymbol{f}_a^{(INT)})(\Delta t)^2}{2m_a} \tag{5.4}$$

$$\boldsymbol{u}_b(\Delta t) = \boldsymbol{u}_b(0) + \frac{\boldsymbol{p}''_b \Delta t}{m_b} - \frac{(\boldsymbol{f}_b - \boldsymbol{f}_a^{(INT)})(\Delta t)^2}{2m_b} \tag{5.5}$$

where

$$\boldsymbol{p}''_a = \boldsymbol{p}'_a + \boldsymbol{f}_a^{(INT)} \Delta t = \boldsymbol{p}_a + \left(\boldsymbol{f}_a + \boldsymbol{f}_a^{(INT)}\right) \Delta t \tag{5.6}$$

$$\boldsymbol{p}''_b = \boldsymbol{p}'_b - \boldsymbol{f}_a^{(INT)} \Delta t = \boldsymbol{p}_b + \left(\boldsymbol{f}_b - \boldsymbol{f}_a^{(INT)}\right) \Delta t \tag{5.7}$$

or partially updated momenta as

$$\boldsymbol{u}_a(\Delta t) = \boldsymbol{u}_a(0) + \frac{\boldsymbol{p}'_a \Delta t}{m_a} - \frac{(\boldsymbol{f}_a - \boldsymbol{f}_a^{(INT)})(\Delta t)^2}{2m_a} \tag{5.8}$$

$$\boldsymbol{u}_b(\Delta t) = \boldsymbol{u}_b(0) + \frac{\boldsymbol{p}'_b \Delta t}{m_b} - \frac{(\boldsymbol{f}_b + \boldsymbol{f}_a^{(INT)})(\Delta t)^2}{2m_b} \tag{5.9}$$

Any above approach is correct; these notes use the first.

For interface modeling, we need to follow the displacement discontinuity or $\boldsymbol{\delta}(\Delta t) = \boldsymbol{u}_b(\Delta t) - \boldsymbol{u}_a(\Delta t)$:

$$\boldsymbol{\delta}(\Delta t) = \boldsymbol{\delta}(0) + \frac{\Delta \boldsymbol{p}_a \Delta t}{m_{red}} + \frac{\Delta \boldsymbol{f}_a (\Delta t)^2}{2m_{red}} - \frac{\boldsymbol{f}_a^{(INT)} (\Delta t)^2}{2m_{red}} \tag{5.10}$$

where

$$\Delta \boldsymbol{p}_a = \frac{m_a \boldsymbol{p}_b - m_b \boldsymbol{p}_a}{m_a + m_b}, \qquad \Delta \boldsymbol{f}_a = \frac{m_a \boldsymbol{f}_b - m_b \boldsymbol{f}_a}{m_a + m_b}, \quad \text{and} \quad m_{red} = \frac{m_a m_b}{m_a + m_b} \tag{5.11}$$

But the displacement discontinuity condition must also follow an equation of motion determined by interfacial traction law that superposes the forces found in MPM with interfacial forces, or:

$$\boldsymbol{\delta}(t) = \boldsymbol{\delta}(0) + \frac{\Delta \boldsymbol{p}_a t}{m_{red}} + \frac{\Delta \boldsymbol{f}_a t^2}{2m_{red}} - \frac{1}{m_{red}} \int_0^t dt_2 \int_0^{t_2} dt_1 \, \boldsymbol{F}_{int}(\boldsymbol{\delta}(t_1)) \tag{5.12}$$

where the interfacial force is

$$\boldsymbol{F}_{int}(\boldsymbol{\delta}(t)) = F_n(\boldsymbol{\delta}(t))\hat{\boldsymbol{n}} + F_t(\boldsymbol{\delta}(t))\hat{\boldsymbol{t}} \tag{5.13}$$

where $F_n(\boldsymbol{\delta}(t))$ and $F_t(\boldsymbol{\delta}(t))$ are normal and tangential imperfect interface laws. Equating the two interfacial discontinuity equations at $t = \Delta t$, the interfacial force we must add (and subtract) becomes

$$\boldsymbol{f}_a^{(INT)} = \frac{2}{(\Delta t)^2} \int_0^{\Delta t} dt_2 \int_0^{t_2} dt_1 \, \boldsymbol{F}_{int}(\boldsymbol{\delta}(t_1)) \tag{5.14}$$

The integral term is found by solving the interfacial equations of motion. The equations for normal and tangential motion are:

$$\delta_n(t) = \delta_n(0) + \frac{d_n t}{m_{red}} + \frac{f_n t^2}{2m_{red}} - \frac{1}{m_{red}} \int_0^t dt_2 \int_0^{t_2} dt_1 \, F_n(\boldsymbol{\delta}(t_1)) \tag{5.15}$$

$$\delta_t(t) = \delta_t(0) + \frac{d_t t}{m_{red}} + \frac{f_t t^2}{2m_{red}} - \frac{1}{m_{red}} \int_0^t dt_2 \int_0^{t_2} dt_1 \, F_t(\boldsymbol{\delta}(t_1)) \tag{5.16}$$

where $d_n = \Delta \boldsymbol{p}_a \cdot \hat{\boldsymbol{n}}$, $d_t = \Delta \boldsymbol{p}_a \cdot \hat{\boldsymbol{t}}$, $f_n = \Delta \boldsymbol{f}_a \cdot \hat{\boldsymbol{n}}$, and $f_t = \Delta \boldsymbol{f}_a \cdot \hat{\boldsymbol{t}}$. These equations provide integral equations to be solved (for given interface law) to find $\delta_n(t)$ and $\delta_t(t)$. The analysis can be converted to differential equations:

$$\frac{d^2 \delta_n(t)}{dt^2} + \frac{F_n(\boldsymbol{\delta}(t))}{m_{red}} = \frac{f_n}{m_{red}} \qquad \text{and} \qquad \frac{d^2 \delta_t(t)}{dt^2} + \frac{F_t(\boldsymbol{\delta}(t))}{m_{red}} = \frac{f_t}{m_{red}} \tag{5.17}$$

Once these are solved, the solutions are used to find an *effective* interface force that must be added such that:

$$\delta_n(\Delta t) = \delta_n(0) + \frac{d_n \Delta t}{m_{red}} + \frac{f_n (\Delta t)^2}{2m_{red}} - \frac{f_{n,a}^{(INT)} (\Delta t)^2}{2m_{red}} \tag{5.18}$$

where the effective force is found from

$$f_{n,a}^{(INT)} = f_n + \frac{2}{\Delta t} \left(\frac{m_{red}\big(\delta_n(0) - \delta_n(\Delta t)\big)}{\Delta t} + d_n \right) \tag{5.19}$$

A similar equation is used for tangential force as well. The total force added to material a is then

$$\boldsymbol{f}_{i,c:a}^{(INT)} = f_{n,a}^{(INT)} \hat{\boldsymbol{n}} + f_{t,a}^{(INT)} \hat{\boldsymbol{t}} \tag{5.20}$$

And equal an opposite force is applied to material b. The equations above are coupled, but in some laws they would decouple if laws can be expressed as $F_n(\delta_n(t))$ and $F_t(\delta_t(t))$ where $\delta_n = \boldsymbol{\delta} \cdot \hat{\boldsymbol{n}}$ and $\delta_t = \boldsymbol{\delta} \cdot \hat{\boldsymbol{t}}$ are normal and tangential components of the interface separation. In other words, they decouple when normal force depends only on normal displacement and tangential force depends only on tangential displacement. Most of the following notes assume this decoupling and therefore might need revision if they do not decouple.

5.2 Interfacial Energy

For calculations of composite properties, it is useful to track interfacial energy. Two approaches are to calculate total interfacial energy on each time step. This approach only works for elastic interfaces (need not be linear). The second approach is to find incremental interfacial energy.

The total interfacial energy is:

$$E_{int} = \int_0^{\delta_n} F_n(\delta)\, d\delta + \int_0^{\delta_t} F_t(\delta)\, d\delta \tag{5.21}$$

The incremental interfacial energy for one component

$$\Delta E_{int} = \int_0^{\Delta t} F_n(\delta_n(t)) \frac{d\delta_n(t)}{dt}\, dt + \int_0^{\Delta t} F_t(\delta_t(t)) \frac{d\delta_t(t)}{dt}\, dt \tag{5.22}$$

The derivative is found after solving for $\delta_n(t)$

5.3 Linear Imperfect Interface

For a linear imperfect interface, such as tangential component of linear interface material is `OSPArticulas`, the differential equations can be solved in closed form. The tangential interface law is $F_t(\delta_t(t)) = D_t A \delta_t(t)$, where D_t is imperfect interface parameter and A is contact area. The differential equation and its solution are:

$$\frac{d^2\delta_t(t)}{dt^2} + \frac{D_t A \delta_t(t)}{m_{red}} = \frac{f_t}{m_{red}} \tag{5.23}$$

$$\delta_t(t) = \frac{f_t}{D_t A} + C_1 \sin \kappa_t t + C_2 \cos \kappa_t t \tag{5.24}$$

$$\text{where} \qquad \kappa_t = \sqrt{\frac{D_t A}{m_{red}}} \tag{5.25}$$

is frequency of the resulting harmonic oscillator. The initial conditions are

$$\delta_t(0) = C_2 + \frac{f_t}{D_t A} \qquad \text{and} \qquad \frac{d\delta_t(0)}{dt} = C_1 \kappa_t = \frac{d_t}{m_{red}} \tag{5.26}$$

The final solution becomes

$$\delta_t(t) = \frac{d_t}{m_{red}\kappa_t} \sin \kappa_t t + \left(\delta_t(0) - \frac{f_t}{D_t A}\right) \cos \kappa_t t + \frac{f_t}{D_t A} \tag{5.27}$$

$$= \frac{d_t}{m_{red}\kappa_t} \sin \kappa_t t + \left(\delta_t(0) - \frac{f_t}{m_{red}\kappa_t^2}\right) \cos \kappa_t t + \frac{f_t}{m_{red}\kappa_t^2} \tag{5.28}$$

The effective force is found using:

$$\delta_t(\Delta t) = \frac{d_t}{m_{red}\kappa_t}\sin\kappa_t\Delta t + \delta_t(0)\cos\kappa_t\Delta t + \frac{f_t}{m_{red}\kappa_t^2}(1-\cos\kappa_t\Delta t) \tag{5.29}$$

which leads to

$$\begin{aligned} f_{t,a}^{(INT)} &= f_t + \frac{2}{\Delta t}\left(\frac{m_{red}}{\Delta t}\left(\delta_t(0)-\delta_t(\Delta t)\right)+d_t\right) && (5.30)\\ &= f_t + \frac{2}{\Delta t}\left(d_t\left(1-\frac{\sin\kappa_t\Delta t}{\kappa_t\Delta t}\right)+\left(\frac{m_{red}\delta_t(0)}{\Delta t}-\frac{f_t}{\kappa_t^2\Delta t}\right)(1-\cos\kappa_t\Delta t)\right) && (5.31)\\ &= \frac{2d_t}{\Delta t}\left(1-\frac{\sin\kappa_t\Delta t}{\kappa_t\Delta t}\right)+\frac{2m_{red}\delta_t(0)\,(1-\cos\kappa_t\Delta t)}{(\Delta t)^2}+f_t\left(1-\frac{2\,(1-\cos\kappa_t\Delta t)}{\kappa_t^2(\Delta t)^2}\right) && (5.32)\end{aligned}$$

As $D_t \to 0$ or $\kappa_t \to 0$ and using L'Hopital's rule:

$$\begin{aligned} \lim_{\kappa_t\to 0}\frac{\sin\kappa_t\Delta t}{\kappa_t\Delta t} &= \lim_{\kappa_t\to 0}\frac{\Delta t\cos\kappa_t\Delta t}{\Delta t} = 1 && (5.33)\\ \lim_{\kappa_t\to 0}\frac{2(1-\cos\kappa_t\Delta t)}{\kappa_t^2(\Delta t)^2} &= \lim_{\kappa_t\to 0}\frac{2\Delta t\sin\kappa_t\Delta t}{2\kappa_t(\Delta t)^2} = 1 && (5.34)\end{aligned}$$

which leads to $f_{t,a}^{(INT)} \to 0$ as expected for a debonded interface. But as $D_t \to \infty$, $f_{t,a}^{(INT)}$ oscillates at increasing frequency. The issue is that MPM analysis cannot handle frequencies beyond its current resolution. Some calculations suggested stability requires $\kappa_t\Delta t < 1$ (or something on that order and certainly must have $\kappa_t\Delta t < \pi/2$).

This observed stability condition is

$$\Delta t < \frac{1}{\kappa_t} = \sqrt{\frac{m_{red}}{D_t A}} \tag{5.35}$$

These values vary in a simulation, but maximum $m_{red} \sim m_i/2 = \rho V_i/2 = \rho\Delta x^3/2$ and $A \sim \Delta x^2$. Thus stability requires

$$\Delta t < \sqrt{\frac{\rho\Delta x^3}{2D_t\Delta x^2}} = \Delta x\sqrt{\frac{\rho}{2D_t\Delta x}} = C_{int}\frac{\Delta x}{v_{int}} \tag{5.36}$$

where C_{int} is a CFL factor for interfaces and v_{int} is an effective wave speed for an interface and given by:

$$v_{int} = \sqrt{\frac{D_t\Delta x}{\rho}} \tag{5.37}$$

Here ρ would be minimum density of the materials in contact. This velocity depends on grid spacing. Imagine simulation with time step $\Delta t = C\Delta x/v$ (where C is CFL factor and v is maximum material wave speed), the maximum D_t that can be simulated would be

$$v_{int} < \frac{C_{int}v}{C}, \quad \sqrt{\frac{D_t\Delta x}{\rho}} < \frac{C_{int}v}{C}, \quad \text{or} \quad D_t < \left(\frac{C_{int}}{C}\right)^2\frac{\rho v^2}{\Delta x} \tag{5.38}$$

Thus, to simulate high D_t for given materials (ρ and v) and an acceptable interface CFL (C_{int}), the simulation needs to have sufficiently low Δx and/or C. To allow any D_t in a simulation, each calculation can check $\kappa_t\Delta t$. If it is larger than 1 (or maybe up to $\pi/2$), the calculation can revert to stick conditions. In other words, the calculation is converted to stick contact (which is like infinite

D_t). If the conversion to stick occurs when D_t has not reached the stick limit, a better simulation would need to increase resolution or decrease time step. A simulation could pre-screen interface laws before starting a calculation, but if that screening was based on $m_{red} \sim m_i/2$, the simulation should still become unstable at node with smaller m_{red}. It is better to check stability on each interface calculation.

5.3.1 Linear Interfacial Energy

The total interfacial energy is

$$E_{int} = \int_0^{\delta_t(\Delta t)} D_t A x \, dx = \frac{1}{2} D_t A \big(\delta_t(\Delta t)\big)^2 \tag{5.39}$$

For incremental energy, the results are:

$$\begin{aligned} \Delta E_{int} &= D_t A \int_0^{\Delta t} \delta_t(t) \frac{d\delta_t(t)}{dt} \, dt = \frac{1}{2} D_t A \int_0^{\Delta t} \frac{d(\delta_t(t))^2}{dt} \, dt & (5.40) \\ &= \frac{1}{2} D_t A \, (\delta_t(t))^2 \big|_0^{\Delta t} = \frac{1}{2} D_t A \left((\delta_t(\Delta t))^2 - (\delta_t(0))^2 \right) & (5.41) \end{aligned}$$

5.3.2 Linear interface Summary and Implentation

Summarizing linear interfaces, set $\phi = \kappa_t \Delta t$ (dimensionless) and $m = m_{red}/\Delta t$ (mass/time), then:

$$\begin{aligned} f_{t,a}^{(INT)} &= \frac{2m\delta_t(0)\,(1-\cos\phi)}{\Delta t} + \frac{2d_t}{\Delta t}\left(1 - \frac{\sin\phi}{\phi}\right) + f_t\left(1 - \frac{2(1-\cos\phi)}{\phi^2}\right) & (5.42) \\ \delta_t(\Delta t) &= \delta_t(0)\cos\phi + \frac{d_t}{m}\frac{\sin\phi}{\phi} + \frac{f_t \Delta t\,(1-\cos\phi)}{m\phi^2} & (5.43) \end{aligned}$$

The final discontinuity can be used in either total or incremental energies above. When calculations are done after updating the momenta:

$$d_t = d_t' - f_t \Delta t \qquad \text{where} \qquad d_t' = \Delta \boldsymbol{p}_a' \cdot \hat{\boldsymbol{t}} \tag{5.44}$$

and $\Delta \boldsymbol{p}_a'$ is the updated momentum term. The force and displacement become:

$$\begin{aligned} f_{t,a}^{(INT)} &= \frac{2m\delta_t(0)\,(1-\cos\phi)}{\Delta t} + \frac{2d_t'}{\Delta t}\left(1 - \frac{\sin\phi}{\phi}\right) + f_t\left(\frac{2\sin\phi}{\phi} - \frac{2(1-\cos\phi)}{\phi^2} - 1\right) & (5.45) \\ \delta_t(\Delta t) &= \delta_t(0)\cos\phi + \frac{d_t'}{m}\frac{\sin\phi}{\phi} + \frac{f_t \Delta t}{m}\left(\frac{1-\cos\phi}{\phi^2} - \frac{\sin\phi}{\phi}\right) & (5.46) \end{aligned}$$

The options for implementation are:

1. The interface calculations can be done after the mass and momentum calculation, which is before forces are known. This calculation can use the above analysis with $f_t = 0$. This approach is correct to first order in time step. It has been used in `OSParticulas` with good results.

2. Alternatively, the calculations can be done after the momentum update and then add the second order terms involving f_t. New calculations suggest the post-update method is better by a small amount, which corresponds to it being a second order calculation of interfacial force. It is the current `OSParticulas` method.

3. Attempts to combine the first two (*i.e.*, update momenta by interfacial force in mass and momentum update for input to constitutive laws, and then again after momentum and add energy calculations) gives very poor results. It is not clear why (because first phase momenta are restored prior to update), but it is a very large effect (and poor results). In other words, use method 1 or 2, but not some combination of the two methods.

4. Note that contact calculations, which include crack and multimaterial nodes modeling imperfect interfaces, are done two or three times per time step. For interfaces, each of these calculations takes momentum change to stick as input. If the interface can find stable normal or tangential force, the component of momentum change in those direction are set to zero. If stability requires conversion to stick or debond, the momentum change is set to model that condition and interface forces are set to zero. Interfacial forces, when determined, are applied to the node only when contact is done after updating momenta. In other words, interface calculations are superposition of interfaces with forces and interfaces acting like stick or debonded contact. They are always one or the other and never both.

5. The toggling between stick (or debonded) and interface forces is essential for stiff interfaces or for when $\phi = \kappa_t \Delta t$ is too large. This condition arises often is interfacial stiffness is too high and at nodes with very low m_{red} (even when interfacial stiffness is not too high). Interface calculations must therefore screen out all high frequency interfaces by switching to stick conditions whenever $\phi > \phi_{max}$. The results do not appear very sensitive to cutoff for values of $\phi_{max} < \pi/2$, although $\phi_{max} = \pi$ shows instabilities. Current code uses $\phi < \pi/2$. When $\phi > \phi_{max}$, the interface uses stick conditions (by leaving stick momentum in the input $\Delta \boldsymbol{p}_a$ and by setting interface force and energy to zero). For non-linear interfaces, unstable conditions could be converted to a debonded interfaces if it is softening, but that is not relevant to linear interfaces.

6. Incremental energy calculations are very bad. Again, the reasons are not certain but perhaps are caused by uncertainty in finding initial separation. In other words, subtracting $\delta_t(0)^2$ does not compensate enough for actual initial energy and accumulation of errors grow with time step. The calculations with the two layer example are probably a worst-case problem because interfacial energy is small overall.

5.3.3 Compare to Old Method

The interface force in my paper was calculated using only the initial discontinuity or:

$$f_{t,a}^{(INT,0)} = D_t A \delta_t(0) = m_{red} \kappa_t^2 \delta_t(0) = \frac{m \delta_t(0) \phi^2}{\Delta t} \tag{5.47}$$

This is identical to the $\delta(0)$ term in harmonic oscillator analysis by expanding $(1 - \cos\phi)$ to two terms. The old method limited forces to value equal to mean of high-frequency forces, but it may not be best way to limit forces. The limit based on ϕ appears better.

If above is considered zeroth order, maybe improved result, and one that would work for non-linear laws, could be to use the zeroth order force to estimate change in discontinuity during the time step. Substituting the force:

$$\delta_t(t) = \delta_t(0) \left(1 - \frac{\phi^2 t^2}{(\Delta t)^2} \right) + \frac{d_t t}{m_{red}} + \frac{f_t t^2}{2 m_{red}} \tag{5.48}$$

The mean displacement during the time step is:

$$\frac{1}{\Delta t}\int_0^{\Delta t}\delta_t(t)dt = \delta_t(0)\left(1-\frac{\phi^2}{6}\right)+\frac{d_t}{2m}+\frac{f_t\Delta t}{6m} \tag{5.49}$$

A force based on mean displacement is:

$$f_{t,a}^{(INT,1)} = \frac{2m\delta_t(0)}{\Delta t}\left(\frac{\phi^2}{2}-\frac{\phi^4}{12}\right)+\frac{2d_t}{\Delta t}\left(\frac{\phi^2}{4}\right)+f_t\left(\frac{\phi^2}{6}\right) \tag{5.50}$$

A result that plots better is

$$f_{t,a}^{(INT,1')} = \frac{f_{t,a}^{(INT,0)}+f_{t,a}^{(INT,1)}}{2} = \frac{2m\delta_t(0)}{\Delta t}\left(\frac{\phi^2}{2}-\frac{\phi^4}{24}\right)+\frac{2d_t}{\Delta t}\left(\frac{\phi^2}{8}\right)+f_t\left(\frac{\phi^2}{12}\right) \tag{5.51}$$

Using updated force, the displacement is

$$\delta_t(t) = \delta_t(0)\left(1-\frac{t^2}{(\Delta t)^2}\left(\frac{\phi^2}{2}-\frac{\phi^4}{24}\right)\right)+\frac{d_t t}{m_{red}}\left(1-\frac{\phi^2 t}{8\Delta t}\right)+\frac{f_t t^2}{2m_{red}}\left(1-\frac{\phi^2}{12}\right) \tag{5.52}$$

I tried another round, but it did not improve the results. The above first order looks like a good option to full solution.

Figure 5.1 compares first order force, $f_{t,a}^{(INT,1')}$, to full method found by solving differential equation over the full range of ϕ allowed for stability.. The curves give interface force for several relative values of $\delta_t(0)$, d_t, and f_t. For a wide range of values, the approximate method is always close to the full harmonic oscillator solution. The two solutions are virtually identical for $d_t,\ f_t << \delta_t(0)$ (*i.e.*, the (0,0) curves). The solutions are also always close when $\phi \sim 0$ as well (*i.e.*, compliant interface with low interface forces). Note that old zeroth method is the dotted line. It falls within all data, is good for compliant interface and for $d_t,\ f_t << \delta_t(0)$ provided $\phi < 1$.

The final results for implementation (based on updated momenta and using position at end from final force) are:

$$f_{t,a}^{(INT,1')} = \frac{m\delta_n(0)\phi^2}{\Delta t}\left(1-\frac{\phi^2}{12}\right)+\frac{d_t'}{\Delta t}\left(\frac{\phi^2}{4}\right)-f_t\left(\frac{\phi^2}{6}\right) \tag{5.53}$$

$$\delta_t(\Delta t) = \delta_t(0)\left(1-\frac{\phi^2}{2}\left(1-\frac{\phi^2}{12}\right)\right)+\frac{d_t'}{m}\left(1-\frac{\phi^2}{8}\right)-\frac{f_t\Delta t}{m}\left(\frac{1}{2}-\frac{\phi^2}{12}\right) \tag{5.54}$$

5.4 Extension to Non-Linear Laws

The revised old method in previous section is OK, but could just use full harmonic oscillator analysis instead. For non-linear laws, however, the differential equation probably cannot be solved and the method of the previous section could guide an approximate method. Repeating the steps for arbitrary $F_t(\delta(t))$ lead to:

1. Define incremental frequency as:

$$\kappa_t = \sqrt{\frac{|F_t'(\delta_t(0))|}{m_{red}}} \qquad \text{and} \qquad \phi = \kappa_t\Delta t \tag{5.55}$$

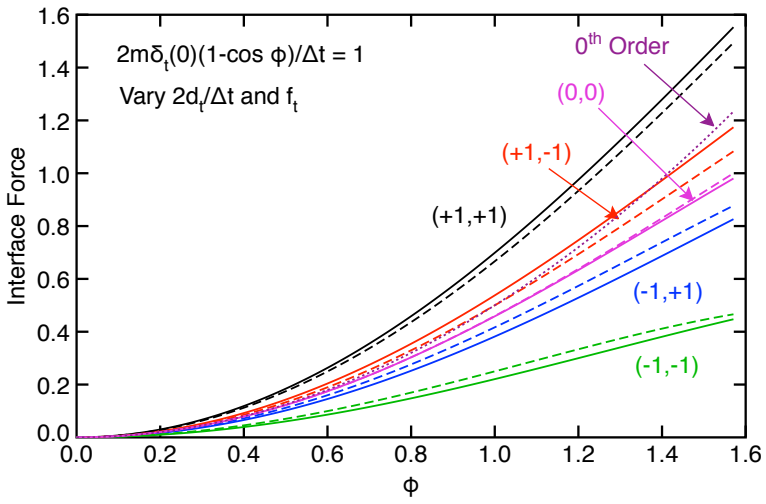

Figure 5.1: Plot of sample force curves for $2m\delta_n(0)\phi^2/\Delta t = 1$ and various values of $(2d_t/\Delta t,\ f_t)$ as indicated. The zeroth order result does not depend on d_t or f_t and is therefore a single curve.

If $\phi > \pi/2$ (or maybe a lower cutoff) proceed with stick conditions. But, if $F_t'(\delta_t(0)) < 0$ (*i.e.*, law with softening region), the linearized differential equation would involve exponential terms that would blow up for large ϕ. In this case, the physical interpretation is that high frequency corresponds to failure and the interface should be treated as debonded (both force and momentum change set to zero). For low ϕ, should be possible to stably model softening.

2. Find the zeroth order force, $f_{t,a}^{(INT,0)} = F_t(\delta(0))$, and calculate displacement discontinuity and its mean value:

$$\delta_t(t) = \delta_t(0) - \frac{F_t(\delta_t(0))t^2}{2m_{red}} + \frac{d_t t}{m_{red}} + \frac{f_t t^2}{2m_{red}} \tag{5.56}$$

$$\left\langle \delta_t^{(1)} \right\rangle = \delta_t(0) - \frac{F_t(\delta_t(0))\Delta t}{6m} + \frac{d_t}{2m} + \frac{f_t \Delta t}{6m} \tag{5.57}$$

3. Find force based on updated mean displacement

$$f_{t,a}^{(INT,1)} = F_t\left(\left\langle \delta_t^{(1)} \right\rangle\right) \tag{5.58}$$

Also check stability by finding $F_t'\left(\left\langle \delta_t^{(1)} \right\rangle\right)$ and switch to stick (or debonding if softening) if needed.

4. Find adjusted force (this combination was based on best approach for linear problem and one that gives exact first two Taylor series terms for the $\cos\phi$ term):

$$f_{t,a}^{(INT,1')} = \frac{F_t(\delta(0)) + F_t\left(\left\langle \delta_t^{(1)} \right\rangle\right)}{2} \tag{5.59}$$

5. Find displacement at end of time step based on adjusted force:

$$\delta_t(\Delta t) = \delta_t(0) - \frac{f_{t,a}^{(INT,1')}\Delta t}{2m} + \frac{d_t}{m} + \frac{f_t \Delta t}{2m} \tag{5.60}$$

Also check stability by finding $F_t'(\delta_t(\Delta t))$ and switch to stick (or debonding if softening) if needed.

6. Calculate total energy based on final discontinuity (note this approach only allows elastic laws and here would find area under curve up to final displacement discontinuity). Energy is only needed for other calculations. It works for composite analysis, but not anything else.

7. Summarizing and using post-update terms:

$$\left\langle \delta_t^{(1)} \right\rangle = \delta_t(0) - \frac{F_t(\delta_t(0))\Delta t}{6m} + \frac{d_t'}{2m} - \frac{f_t \Delta t}{3m} \tag{5.61}$$

$$f_{t,a}^{(INT,1')} = \frac{F_t(\delta(0)) + F_t\left(\left\langle \delta_t^{(1)} \right\rangle\right)}{2} \tag{5.62}$$

$$\delta_t(\Delta t) = \delta_t(0) - \frac{f_{t,a}^{(INT,1')}\Delta t}{2m} + \frac{d_t'}{m} - \frac{f_t \Delta t}{2m} \tag{5.63}$$

with stability checks done for $\delta(0)$, $\left\langle \delta_t^{(1)} \right\rangle$, and $\delta_t(\Delta t)$, although maybe mean one can be skipped. Implementation could use these equations, or just stope with zeroth order force and displacements (or an option to pick). Class to implement any non-linear law only needs to provide $F_t(\delta)$, $F_t'(\delta)$, and methods to check stability (for stick, stable, or debonding) and to find energy.

5.5 Bilinear Imperfect Interface

The linear interface contact law in `OSParticulas` is bilinear in the normal direction (and therefore nonlinear) with different interface parameters in tension (D_{nt}) and compression (D_{nc}). We start with the interfaces initially separated or $\delta_n(0) > 0$. The situation evolves as a linear interface and remains that way as long as $\delta_n(t)$ remains positive. Assuming all linear, final separation is

$$\delta_t(\Delta t) = \frac{d_n}{m_{red}\kappa_{nt}} \sin \kappa_{nt}\Delta t + \left(\delta_n(0) - \frac{f_n}{m_{red}\kappa_{nt}^2}\right) \cos \kappa_{nt}\Delta t + \frac{f_n}{m_{red}\kappa_{nt}^2} \tag{5.64}$$

$$\kappa_{nt} = \sqrt{\frac{D_{nt}A}{m_{red}}} \tag{5.65}$$

If $\delta_t(\Delta t) > 0$, use fully linear analysis in previous section (using normal direction parameters). Similarly, if start with $\delta_n(0) < 0$, the interface evolves as a linear interface as long as $\delta_n(t)$ remains negative. The equations are same as above, but use compression stiffness.

$$\kappa_{nc} = \sqrt{\frac{D_{nc}A}{m_{red}}} \tag{5.66}$$

and check if $\delta_t(\Delta t) < 0$ before using linear analysis.

The above calculations ignores possible vibrations of the interface opening and closing during the time step but ending up the same way it started.

5.5.1 Intrastep Contact

If $\delta_t(\Delta t)$ goes from positive to negative during the time step, the separation becomes zero during the time step at $0 < t_c < \Delta t$ found by solving

$$0 = \frac{d_n}{m_{red}\kappa_{nt}} \sin \kappa_{nt} t_c + \left(\delta_n(0) - \frac{f_n}{m_{red}\kappa_{nt}^2}\right) \cos \kappa_{nt} t_c + \frac{f_n}{m_{red}\kappa_{nt}^2} \tag{5.67}$$

If $\delta_t(\Delta t)$ goes from negative to positive during the time step, we solve the same equation, but use κ_{nc}. Two forms useful below are:

$$\delta_n(0) - \frac{f_n}{m_{red}\kappa_{nt}^2} = -\frac{d_n}{m_{red}\kappa_{nt}}\tan\kappa_{nt}t_c - \frac{f_n}{m_{red}\kappa_{nt}^2}\sec\kappa_{nt}t_c \tag{5.68}$$

$$\frac{d_n}{m_{red}\kappa_{nt}} = -\left(\delta_n(0) - \frac{f_n}{m_{red}\kappa_{nt}^2}\right)\cot\kappa_{nt}t_c - \frac{f_n}{m_{red}\kappa_{nt}^2}\csc\kappa_{nt}t_c \tag{5.69}$$

Based on stability arguments, we know that $0 < \kappa_{nt}t_c < \pi/2$ and therefore both $\sin\kappa_{nt}t_c$ and $\cos\kappa_{nt}t_c$ are positive. We can solve (with $d_n = d_n' - f_n\Delta t$) as follows:

$$-\left(\delta_n(0) - \frac{f_n}{m_{red}\kappa_{nt}^2}\right)\cos\kappa_{nt}t_c = \frac{d_n}{m_{red}\kappa_{nt}}\sin\kappa_{nt}t_c + \frac{f_n}{m_{red}\kappa_{nt}^2} \tag{5.70}$$

$$-\left(m_{red}\kappa_{nt}^2\delta_n(0) - f_n\right)\cos\kappa_{nt}t_c = \kappa_{nt}d_n\sin\kappa_{nt}t_c + f_n \tag{5.71}$$

$$\left(m_{red}\kappa_{nt}^2\delta_n(0) - f_n\right)^2\left(1 - \sin^2\kappa_{nt}t_c\right) = \kappa_{nt}^2d_n^2\sin^2\kappa_{nt}t_c + 2\kappa_{nt}f_nd_n\sin\kappa_{nt}t_c + f_n^2 \tag{5.72}$$

or a quadratic equation

$$0 = a\sin^2\kappa_{nt}t_c + b\sin\kappa_{nt}t_c + c \tag{5.73}$$

$$a = \left(m_{red}\kappa_{nt}^2\delta_n(0) - f_n\right)^2 + \kappa_{nt}^2d_n^2 \tag{5.74}$$

$$b = 2\kappa_{nt}f_nd_n \tag{5.75}$$

$$c = f_n^2 - \left(m_{red}\kappa_{nt}^2\delta_n(0) - f_n\right)^2 \tag{5.76}$$

The solution is positive term between 0 and Δt from:

$$\sin\kappa_{nt}t_c = \frac{-\kappa_{nt}f_nd_n \pm \left(m_{red}\kappa_{nt}^2\delta_n(0) - f_n\right)\sqrt{\left(m_{red}\kappa_{nt}^2\delta_n(0) - f_n\right)^2 + \kappa_{nt}^2d_n^2 - f_n^2}}{\left(m_{red}\kappa_{nt}^2\delta_n(0) - f_n\right)^2 + \kappa_{nt}^2d_n^2} \tag{5.77}$$

$$\cos\kappa_{nt}t_c = \frac{-f_n\left(m_{red}\kappa_{nt}^2\delta_n(0) - f_n\right) \mp \kappa_{nt}d_n\sqrt{\left(m_{red}\kappa_{nt}^2\delta_n(0) - f_n\right)^2 + \kappa_{nt}^2d_n^2 - f_n^2}}{\left(m_{red}\kappa_{nt}^2\delta_n(0) - f_n\right)^2 + \kappa_{nt}^2d_n^2} \tag{5.78}$$

In pre-update calculations (when $f_n = 0$ and therefore $d_n < 0$ for all contact cases), a simpler solution is

$$\tan\kappa_{nt}t_c = \frac{-m_{red}\kappa_{nt}\delta_n(0)}{d_n} \quad \text{or} \quad t_c = \frac{1}{\kappa_{nt}}\tan^{-1}\left(\frac{-m_{red}\kappa_{nt}\delta_n(0)}{d_n}\right) \tag{5.79}$$

Or from general solution with $f_n = 0$

$$\sin\kappa_{nt}t_c = \frac{m_{red}\kappa_{nt}\delta_n(0)}{\sqrt{\left(m_{red}\kappa_{nt}\delta_n(0)\right)^2 + d_n^2}} \quad \text{and} \quad \cos\kappa_{nt}t_c = \frac{-d_n}{\sqrt{\left(m_{red}\kappa_{nt}\delta_n(0)\right)^2 + d_n^2}} \tag{5.80}$$

5.5.2 Intrastep Contact of Nearly Debonded Interface

The fully debonded interface is considered in section 5.6.1. Here consider contact of nearly debonded interface or Eq. (5.73) when κ_{nt} (or κ_{nc}) is small. If $f_n \neq 0$ and $\sin\kappa_{nt}t_c \approx \kappa_{nt}t_c$ and resulting

constants divided by $2\kappa_{nt} m_{red} f_n$, the results reduce to:

$$0 = a t_c^2 + b t_c + c \tag{5.81}$$

$$a = \frac{f_n}{2m_{red}} + \left(\frac{d_n^2}{2m_{red} f_n} - \delta_n(0) \left(1 - \frac{m_{red} \delta_n(0)}{2 f_n} \kappa_{nt}^2 \right) \right) \kappa_{nt}^2 \tag{5.82}$$

$$b = \frac{d_n}{m_{red}} \tag{5.83}$$

$$c = \delta_n(0) \left(1 - \frac{m_{red} \delta_n(0)}{2 f_n} \kappa_{nt}^2 \right) \tag{5.84}$$

Neglecting terms κ_{nt}^2 and higher and multiplying back by $2m_{red}$ gives:

$$0 = f_n t_c^2 + 2 d_n t_c + 2 m_{red} \delta_n(0) \tag{5.85}$$

$$t_c = \frac{-d_n \pm \sqrt{d_n^2 - 2 f_n m_{red} \delta_n(0)}}{f_n} \tag{5.86}$$

If almost debonded ($\kappa_{nt} t_c$ small) and $f_n = 0$ and constants divided by $m_{red}^2 \kappa_{nt}^4$, the results reduce to:

$$0 = a t_c^2 + c, \qquad a = \frac{d_n^2}{m_{red}^2} + \kappa_{nt}^2 \delta_n(0)^2, \qquad c = \delta_n(0)^2 \tag{5.87}$$

$$t_c = \frac{|\delta_n(0)|}{\sqrt{\frac{d_n^2}{m_{red}^2} + \kappa_{nt}^2 \delta_n(0)^2}} \tag{5.88}$$

where absolute value are to use this starting opened or closed. Expanding in series and neglecting terms κ_{nt}^2 and higher gives:

$$t_c = \frac{m_{red} |\delta_n(0)|}{d_n} \tag{5.89}$$

which is solution to quadratic equation above when $f_n = 0$. A stable quadratic equation solver gets both results. These forms are used in code when ϕ is very small. Otherwise, t_c is found from the full Eq. (5.73).

5.5.3 Post Contact with new Interface Parameter

The displacement discontinuity allow for contact change at t_c within the time step is and relation for zero contact at t_c are:

$$\begin{aligned} \delta_n(t) = {} & \delta(0) + \frac{d_n t}{m_{red}} + \frac{f_n t^2}{2 m_{red}} - \frac{1}{m_{red}} \int_0^{t_c} dt_2 \int_0^{t_1} dt_1 D_{nt} A \delta_n(t_1) \\ & - \frac{1}{m_{red}} \int_{t_c}^{t} dt_2 \left(\int_0^{t_c} dt_1 D_{nt} A \delta_n(t_1) + \int_{t_c}^{t_2} dt_1 D_{nc} A \delta_n(t_1) \right) \end{aligned} \tag{5.90}$$

$$\delta_n(t_c) = 0 = \delta(0) + \frac{d_n t_c}{m_{red}} + \frac{f_n t_c^2}{2 m_{red}} - \frac{1}{m_{red}} \int_0^{t_c} dt_2 \int_0^{t_1} dt_1 D_{nt} A \delta_n(t_1) \tag{5.91}$$

Note that this post-contact analysis assumes that the interface remains in contact and does not oscillate between opened and closed. That later might be possible due the harmonic oscillations

inherent in final solution. Subtracting the second equation (which is zero) from the first gives:

$$\begin{aligned}\delta_n(t) &= \frac{d_n(t-t_c)}{m_{red}} + \frac{f_n(t^2-t_c^2)}{2m_{red}} - \frac{1}{m_{red}}\int_{t_c}^{t} dt_2 \left(\int_0^{t_c} dt_1 D_{nt} A\delta_n(t_1) + \int_{t_c}^{t_2} dt_1 D_{nc} A\delta_n(t_1)\right) \\ &= \frac{d_n(t-t_c)}{m_{red}} + \frac{f_n(t^2-t_c^2)}{2m_{red}} - \frac{t-t_c}{m_{red}}\int_0^{t_c} dt_1 D_{nt} A\delta_n(t_1) - \frac{1}{m_{red}}\int_{t_c}^{t} dt_2 \int_{t_c}^{t_2} dt_1 D_{nc} A\delta_n(t_1) \\ &= \frac{d_n^{(c)}(t-t_c)}{m_{red}} + \frac{f_n(t-t_c)^2}{2m_{red}} - \frac{1}{m_{red}}\int_{t_c}^{t} dt_2 \int_{t_c}^{t_2} dt_1 D_{nc} A\delta_n(t_1) \qquad (5.92)\end{aligned}$$

where $d_n^{(c)}$ is found by equating the last two equations above

$$\begin{aligned}\frac{d_n^{(c)}}{m_{red}} &= \frac{d_n}{m_{red}} + \frac{f_n t_c}{m_{red}} - \frac{1}{m_{red}}\int_0^{t_c} dt_1 D_{nt} A\delta_n(t_1) \qquad (5.93) \\ &= \frac{d_n}{m_{red}} + \frac{f_n t_c}{m_{red}} - \frac{D_{nt}A}{m_{red}\kappa_{nt}^2}\left(-\frac{d_n}{m_{red}}\cos\kappa_{nt}t_1\right. \\ &\qquad \left. + \left(\delta_n(0) - \frac{f_n}{m_{red}\kappa_{nt}^2}\right)\kappa_{nt}\sin\kappa_{nt}t_1 + \frac{f_n t}{m_{red}}\right)\Bigg|_0^{t_c} \qquad (5.94) \\ &= \frac{d_n}{m_{red}}\cos\kappa_{nt}t_c - \left(\delta_n(0) - \frac{f_n}{m_{red}\kappa_{nt}^2}\right)\kappa_{nt}\sin\kappa_{nt}t_c \qquad (5.95)\end{aligned}$$

Note that $d_n^{(c)}$ is also the momentum associated with separation velocity and the time of contact found by differentiating Eq. (5.64) using $\Delta t = t_c$:

$$\frac{d_n^{(c)}}{m_{red}} = \frac{d\delta_n(t_c)}{dt} = \frac{d_n}{m_{red}}\cos\kappa_{nt}t_c - \left(\delta_n(0) - \frac{f_n}{m_{red}\kappa_{nt}^2}\right)\kappa_{nt}\sin\kappa_{nt}t_c \qquad (5.96)$$

which leads to an identical result. If using post-update terms then

$$\begin{aligned}\frac{d_n^{(c)}}{m_{red}} &= \frac{d_n' - f_n\Delta t}{m_{red}}\cos\kappa_{nt}t_c - \delta_n(0)\kappa_{nt}\sin\kappa_{nt}t_c + \frac{f_n}{m_{red}\kappa_{nt}}\sin\kappa_{nt}t_c \qquad (5.97) \\ &= \frac{d_n'}{m_{red}}\cos\kappa_{nt}t_c - \delta_n(0)\kappa_{nt}\sin\kappa_{nt}t_c + \frac{f_n t_c}{m_{red}}\left(\frac{\sin\kappa_{nt}t_c}{\kappa_{nt}t_c} - \frac{\Delta t}{t_c}\cos\kappa_{nt}t_c\right) \qquad (5.98)\end{aligned}$$

We next shift the time axis to $t' = t - t_c$, to get

$$\delta_n(t') = \frac{d_n^{(c)}t'}{m_{red}} + \frac{f_n t'^2}{2m_{red}} - \frac{1}{m_{red}}\int_0^{t'} dt_2' \int_0^{t_2} dt_1' D_{nc} A\delta_n(t_1') \qquad (5.99)$$

The general solution to the differential equation is

$$\delta_n(t') = \frac{f_n}{m_{red}\kappa_{nc}^2} + C_1\sin\kappa_{nc}t' + C_2\cos\kappa_{nc}t' \qquad \text{where} \qquad \kappa_{nc} = \sqrt{\frac{D_{nc}A}{m_{red}}} \qquad (5.100)$$

The initial conditions are

$$\delta_n(0) = 0 = \frac{f_n}{m_{red}\kappa_{nc}^2} + C_2 \qquad \text{and} \qquad \frac{d\delta_n(0)}{dt} = C_1\kappa_{nc} = \frac{d_n^{(c)}}{m_{red}} \qquad (5.101)$$

The final solution becomes

$$\delta_n(t') = \frac{d_n^{(c)}}{m_{red}\kappa_{nc}}\sin\kappa_{nc}t' - \frac{f_n}{m_{red}\kappa_{nc}^2}\cos\kappa_{nc}t' + \frac{f_n}{m_{red}\kappa_{nc}^2} \tag{5.102}$$

$$\delta_n(t) = \frac{d_n^{(c)}}{m_{red}\kappa_{nc}}\sin\kappa_{nc}(t-t_c) - \frac{f_n}{m_{red}\kappa_{nc}^2}\cos\kappa_{nc}(t-t_c) + \frac{f_n}{m_{red}\kappa_{nc}^2} \tag{5.103}$$

Then using Eq. (5.19), the effective interface force for entire step along with the final discontinuity become:

$$f_{n,a}^{(INT)} = \frac{2}{\Delta t}\left(m\delta_n(0) + d_n - d_n^{(c)}\frac{\sin\kappa_{nc}(\Delta t - t_c)}{\kappa_{nc}\Delta t}\right) + f_n\left(1 - \frac{2\left(1-\cos\kappa_{nc}(\Delta t - t_c)\right)}{\kappa_{nc}^2(\Delta t)^2}\right) \tag{5.104}$$

$$\delta_t(\Delta t) = \frac{d_n^{(c)}}{m_{red}}\frac{\sin\kappa_{nc}(\Delta t - t_c)}{\kappa_{nc}} + \frac{f_n}{m_{red}}\left(\frac{1-\cos\kappa_{nc}(\Delta t - t_c)}{\kappa_{nc}^2}\right) \tag{5.105}$$

These are the equations implements in the linear interface contact low. The next section expands the trigonometric terms for perhaps a more basic result, but the result equations are more complex and less convenient for programming.

In programming, a special case for κ_{nc} is small (needed most when it is zero) neglecting terms κ_{nc}^2 and higher is:

$$f_{n,a}^{(INT)} = \frac{2}{\Delta t}\left(m\delta_n(0) + d_n - d_n^{(c)}\left(1 - \frac{t_c}{\Delta t}\right)\right) + f_n\left(1 - \left(1 - \frac{t_c}{\Delta t}\right)^2\right) \tag{5.106}$$

$$\delta_t(\Delta t) = \frac{d_n^{(c)}}{m_{red}}(\Delta t - t_c) + \frac{f_n}{2m_{red}}(\Delta t - t_c)^2 \tag{5.107}$$

5.5.4 Expand to Initial Time Step Values

Expanding the trigonometric terms:

$$\begin{aligned}\delta_n(t') = {} & \frac{d_n^{(c)}}{m_{red}\kappa_{nc}}\left(\sin\kappa_{nc}t\cos\kappa_{nc}t_c - \sin\kappa_{nc}t_c\cos\kappa_{nc}t\right) \\ & - \frac{f_n}{m_{red}\kappa_{nc}^2}\left(\cos\kappa_{nc}t\cos\kappa_{nc}t_c + \sin\kappa_{nc}t\sin\kappa_{nc}t_c\right) + \frac{f_n}{m_{red}\kappa_{nc}^2}\end{aligned} \tag{5.108}$$

which simplifies to:

$$\delta_n(t) = \frac{d_n^{(c)}\cos\kappa_{nc}t_c - \frac{f_n\sin\kappa_{nc}t_c}{\kappa_{nc}}}{m_{red}\kappa_{nc}}\sin\kappa_{nc}t - \frac{d_n^{(c)}\sin\kappa_{nc}t_c + \frac{f_n\cos\kappa_{nc}t_c}{\kappa_{nc}}}{m_{red}\kappa_{nc}}\cos\kappa_{nc}t + \frac{f_n}{m_{red}\kappa_{nc}^2} \tag{5.109}$$

The coefficients simplify with the contact conditions as follows:

$$\begin{aligned}\frac{d_n^{(c)}\cos\kappa_{nc}t_c - \frac{f_n\sin\kappa_{nc}t_c}{\kappa_{nc}}}{m_{red}\kappa_{nc}} &= \left[\frac{d_n}{m_{red}}\cos\kappa_{nt}t_c - \left(\delta_n(0) - \frac{f_n}{m_{red}\kappa_{nt}^2}\right)\kappa_{nt}\sin\kappa_{nt}t_c\right]\frac{\cos\kappa_{nc}t_c}{\kappa_{nc}} \\ &\quad - \frac{f_n\sin\kappa_{nc}t_c}{m_{red}\kappa_{nc}^2}\end{aligned} \tag{5.110}$$

$$= \left[\frac{d_n}{m_{red}}\cos\kappa_{nt}t_c + \left(\frac{d_n}{m_{red}}\tan\kappa_{nt}t_c + \frac{f_n}{m_{red}\kappa_{nt}}\sec\kappa_{nt}t_c\right)\sin\kappa_{nt}t_c\right]\frac{\cos\kappa_{nc}t_c}{\kappa_{nc}} - \frac{f_n\sin\kappa_{nc}t_c}{m_{red}\kappa_{nc}^2}$$

$$= \frac{1}{m_{red}\kappa_{nc}}\left[\left(d_n + \frac{f_n\sin\kappa_{nt}t_c}{\kappa_{nt}}\right)\frac{\cos\kappa_{nc}t_c}{\cos\kappa_{nt}t_c} - \frac{f_n\sin\kappa_{nc}t_c}{\kappa_{nc}}\right] \tag{5.111}$$

and

$$\frac{d_n^{(c)}\sin\kappa_{nc}t_c+\frac{f_n\cos\kappa_{nc}t_c}{\kappa_{nc}}}{m_{red}\kappa_{nc}}=\left[\frac{d_n}{m_{red}}\cos\kappa_{nt}t_c-\left(\delta_n(0)-\frac{f_n}{m_{red}\kappa_{nt}^2}\right)\kappa_{nt}\sin\kappa_{nt}t_c\right]\frac{\sin\kappa_{nc}t_c}{\kappa_{nc}}$$
$$+\frac{f_n\cos\kappa_{nc}t_c}{m_{red}\kappa_{nc}^2} \tag{5.112}$$
$$=-\left(\delta_n(0)-\frac{f_n}{m_{red}\kappa_{nt}^2}+\frac{f_n\cos\kappa_{nt}t_c}{m_{red}\kappa_{nt}^2}\right)\frac{\kappa_{nt}\sin\kappa_{nc}t_c}{\kappa_{nc}\sin\kappa_{nt}t_c}+\frac{f_n\cos\kappa_{nc}t_c}{m_{red}\kappa_{nc}^2} \tag{5.113}$$

If the problem was linear ($\kappa_{nt}=\kappa_{nc}=\kappa_n$) then

$$\frac{d_n^{(c)}\cos\kappa_{nc}t_c-\frac{f_n\sin\kappa_{nc}t_c}{\kappa_{nc}}}{m_{red}\kappa_{nc}} = \frac{d_n}{\kappa_n m_{red}} \tag{5.114}$$
$$\frac{d_n^{(c)}\sin\kappa_{nc}t_c+\frac{f_n\cos\kappa_{nc}t_c}{\kappa_{nc}}}{m_{red}\kappa_{nc}} = -\left(\delta_n(0)-\frac{f_n}{m_{red}\kappa_n^2}\right) \tag{5.115}$$

These results recover the linear result as it should.

The net effective force is found from

$$\delta_n(\Delta) = \frac{d_n^{(c)}\cos\kappa_{nc}t_c-\frac{f_n\sin\kappa_{nc}t_c}{\kappa_{nc}}}{m_{red}\kappa_{nc}}\sin\kappa_{nc}\Delta t-\frac{d_n^{(c)}\sin\kappa_{nc}t_c+\frac{f_n\cos\kappa_{nc}t_c}{\kappa_{nc}}}{m_{red}\kappa_{nc}}\cos\kappa_{nc}\Delta t$$
$$+\frac{f_n}{m_{red}\kappa_{nc}^2}=\delta_n(0)+\frac{d_n\Delta t}{m_{red}}+\frac{f_n(\Delta t)^2}{2m_{red}}-\frac{f_{n,a}^{(INT)}(\Delta t)^2}{2m_{red}} \tag{5.116}$$
$$f_{n,a}^{(INT)} = f_n+\frac{2}{\Delta t}\left[d_n\left(1-\frac{d_n^{(c)}\cos\kappa_{nc}t_c-\frac{f_n\sin\kappa_{nc}t_c}{\kappa_{nc}}}{d_n}\frac{\sin\kappa_{nc}\Delta t}{\kappa_{nc}\Delta t}\right)-\frac{f_n}{\kappa_{nc}^2\Delta t}\right.$$
$$\left.+\frac{\delta_n(0)m_{red}}{\Delta t}\left(1+\frac{d_n^{(c)}\sin\kappa_{nc}t_c+\frac{f_n\cos\kappa_{nc}t_c}{\kappa_{nc}}}{\delta_n(0)m_{red}\kappa_{nc}}\cos\kappa_{nc}\Delta t\right)\right] \tag{5.117}$$

This result reduces to linear result when $\kappa_{nt}=\kappa_{nc}=\kappa_n$. Inserting revised coefficients for bilinear gives:

$$f_{n,a}^{(INT)}=f_n+\frac{2}{\Delta t}\left[\left(d_n-\left[\left(d_n+\frac{f_n\sin\kappa_{nt}t_c}{\kappa_{nt}}\right)\frac{\cos\kappa_{nc}t_c}{\cos\kappa_{nt}t_c}-\frac{f_n\sin\kappa_{nc}t_c}{\kappa_{nc}}\right]\frac{\sin\kappa_{nc}\Delta t}{\kappa_{nc}\Delta t}\right)-\frac{f_n}{\kappa_{nc}^2\Delta t}\right.$$
$$\left.+\left(\frac{m_{red}\delta_n(0)}{\Delta t}-\left[\left(\frac{m_{red}\delta_n(0)}{\Delta t}-\frac{f_n}{\kappa_{nt}^2\Delta t}\left(1-\cos\kappa_{nt}t_c\right)\right)\frac{\kappa_{nt}\sin\kappa_{nc}t_c}{\kappa_{nc}\sin\kappa_{nt}t_c}-\frac{f_n\cos\kappa_{nc}t_c}{\kappa_{nc}^2\Delta t}\right]\cos\kappa_{nc}\Delta t\right)\right] \tag{5.118}$$

In pre-update when $f_n=0$, this results reduces to:

$$f_{n,a}^{(INT)}=\frac{2}{\Delta t}\left[d_n\left(1-\frac{\cos\kappa_{nc}t_c}{\cos\kappa_{nt}t_c}\frac{\sin\kappa_{nc}\Delta t}{\kappa_{nc}\Delta t}\right)+\frac{\delta_n(0)m_{red}}{\Delta t}\left(1-\frac{\kappa_{nt}\sin\kappa_{nc}t_c}{\kappa_{nc}\sin\kappa_{nt}t_c}\cos\kappa_{nc}\Delta t\right)\right] \tag{5.119}$$

Get f_n term by setting $d_n=\delta_n(0)=0$ or

$$f_{n,a}^{(INT)}=f_n\left[1-\frac{2}{\Delta t}\left(\frac{\sin\kappa_{nt}t_c}{\kappa_{nt}}\frac{\cos\kappa_{nc}t_c}{\cos\kappa_{nt}t_c}-\frac{\sin\kappa_{nc}t_c}{\kappa_{nc}}\right)\frac{\sin\kappa_{nc}\Delta t}{\kappa_{nc}\Delta t}\right.$$
$$\left.-\frac{2}{\kappa_{nc}^2(\Delta t)^2}\left(1-\left(\left(1-\cos\kappa_{nt}t_c\right)\frac{\kappa_{nt}\sin\kappa_{nc}t_c}{\kappa_{nc}\sin\kappa_{nt}t_c}+\cos\kappa_{nc}t_c\right)\cos\kappa_{nc}\Delta t\right)\right] \tag{5.120}$$

For energy calculations, we can find

$$\begin{aligned}\delta_n(\Delta) &= \frac{1}{m_{red}\kappa_{nc}}\left[\left(d_n+\frac{f_n\sin\kappa_{nt}t_c}{\kappa_{nt}}\right)\frac{\cos\kappa_{nc}t_c}{\cos\kappa_{nt}t_c}-\frac{f_n\sin\kappa_{nc}t_c}{\kappa_{nc}}\right]\sin\kappa_{nc}\Delta t+\frac{f_n}{m_{red}\kappa_{nc}^2}\\ &+\left[\left(\delta_n(0)-\frac{f_n}{m_{red}\kappa_{nt}^2}+\frac{f_n\cos\kappa_{nt}t_c}{m_{red}\kappa_{nt}^2}\right)\frac{\kappa_{nt}\sin\kappa_{nc}t_c}{\kappa_{nc}\sin\kappa_{nt}t_c}-\frac{f_n\cos\kappa_{nc}t_c}{m_{red}\kappa_{nc}^2}\right]\cos\kappa_{nc}\Delta t\end{aligned} \tag{5.121}$$

In pre-update when $f_n=0$, this results reduces to:

$$\delta_n(\Delta t)=\frac{d_n}{m_{red}\kappa_{nc}}\frac{\cos\kappa_{nc}t_c}{\cos\kappa_{nt}t_c}\sin\kappa_{nc}\Delta t+\delta_n(0)\frac{\kappa_{nt}\sin\kappa_{nc}t_c}{\kappa_{nc}\sin\kappa_{nt}t_c}\cos\kappa_{nc}\Delta t \tag{5.122}$$

Get f_n term by setting $d_n=\delta_n(0)=0$ or

$$\begin{aligned}m_{red}\delta_t(\Delta) &= f_n\left[\left(\frac{\sin\kappa_{nt}t_c}{\kappa_{nt}}\frac{\cos\kappa_{nc}t_c}{\cos\kappa_{nt}t_c}-\frac{\sin\kappa_{nc}t_c}{\kappa_{nc}}\right)\frac{\sin\kappa_{nc}\Delta t}{\kappa_{nc}}\right.\\ &\left.+\frac{1}{\kappa_{nc}^2}\left(1-\left(\left(1-\cos\kappa_{nt}t_c\right)\frac{\kappa_{nt}\sin\kappa_{nc}t_c}{\kappa_{nc}\sin\kappa_{nt}t_c}+\cos\kappa_{nc}t_c\right)\cos\kappa_{nc}\Delta t\right)\right]\end{aligned} \tag{5.123}$$

Total energy is found at the end of the time step. An incremental approach needs to look at results before an after the contact time.

$$\Delta E_{int} = \frac{1}{2}D_{nt}A\int_0^{t_c}\frac{d(\delta_t(t))^2}{dt}\,dt+\frac{1}{2}D_{nc}A\int_{t_c}^{\Delta t}\frac{d(\delta_t(t))^2}{dt}\,dt \tag{5.124}$$

$$= \frac{1}{2}D_{nt}A\left(\delta_t(t)\right)^2\Big|_0^{t_c}+\frac{1}{2}D_{nc}A\left(\delta_t(t)\right)^2\Big|_{t_c}^{\Delta t}=\frac{1}{2}A\left(D_{nc}(\delta_t(\Delta t))^2-D_{nt}(\delta_t(0))^2\right) \tag{5.125}$$

Summarizing, the interfacial force with $\phi=\kappa_{nc}\Delta t$ and $m=m_{red}/\Delta t$ is

$$\begin{aligned}f_{n,a}^{(INT)} &= \frac{2}{\Delta t}\left[d_n\left(1-\frac{\cos\kappa_{nc}t_c}{\cos\kappa_{nt}t_c}\frac{\sin\phi}{\phi}\right)+m\delta_n(0)\left(1-\frac{\kappa_{nt}\sin\kappa_{nc}t_c}{\kappa_{nc}\sin\kappa_{nt}t_c}\cos\phi\right)\right]\\ &+f_n\left[1-\frac{2}{\Delta t}\left(\frac{\sin\kappa_{nt}t_c}{\kappa_{nt}}\frac{\cos\kappa_{nc}t_c}{\cos\kappa_{nt}t_c}-\frac{\sin\kappa_{nc}t_c}{\kappa_{nc}}\right)\frac{\sin\phi}{\phi}\right.\\ &\left.-\frac{2\left(1-\left(\left(1-\cos\kappa_{nt}t_c\right)\frac{\kappa_{nt}\sin\kappa_{nc}t_c}{\kappa_{nc}\sin\kappa_{nt}t_c}+\cos\kappa_{nc}t_c\right)\cos\phi\right)}{\phi^2}\right]\end{aligned} \tag{5.126}$$

and interface separation is

$$\begin{aligned}\delta_n(\Delta t) &= \delta_n(0)\frac{\kappa_{nt}\sin\kappa_{nc}t_c}{\kappa_{nc}\sin\kappa_{nt}t_c}\cos\phi+\frac{d_n}{m}\frac{\cos\kappa_{nc}t_c}{\cos\kappa_{nt}t_c}\frac{\sin\phi}{\phi}\\ &+\frac{f_n}{m}\left[\left(\frac{\sin\kappa_{nt}t_c}{\kappa_{nt}}\frac{\cos\kappa_{nc}t_c}{\cos\kappa_{nt}t_c}-\frac{\sin\kappa_{nc}t_c}{\kappa_{nc}}\right)\frac{\sin\phi}{\phi}\right.\\ &\left.+\frac{\Delta t\left(1-\left(\left(1-\cos\kappa_{nt}t_c\right)\frac{\kappa_{nt}\sin\kappa_{nc}t_c}{\kappa_{nc}\sin\kappa_{nt}t_c}+\cos\kappa_{nc}t_c\right)\cos\phi\right)}{\phi^2}\right]\end{aligned} \tag{5.127}$$

An interface that starts in contact and separates would use same equation but interchange compression and tension terms.

5.6 Debonding Interface and Perfect in Compression

5.6.1 Debonded Interface

Imagine a debonded interface with no force when separated. If not debonded in compression, it is bilinear. If perfect in compression, it is should be a new approach to contact. We start with finite stiffness in compression and then try to deduce results for perfect interface.

Starting with $\delta_n(0) > 0$, the only concern is if contact occurs in the time step. The discontinuity without contact forces is:

$$\delta_n(t) = \delta_n(0) + \frac{d_n t}{m_{red}} + \frac{f_n t^2}{2m_{red}} \tag{5.128}$$

If $\delta_n(\Delta t) > 0$, interface remains debonded with zero force and energy for time step. If negative, then

$$\delta_n(0) < -\frac{\left(d_n + \frac{1}{2} f_n \Delta t\right) \Delta t}{m_{red}} \tag{5.129}$$

When this applies, contact occurs at:

$$0 = 2m_{red}\delta_n(0) + 2d_n t_c + f_n t_c^2 \tag{5.130}$$

$$t_c = \frac{-d_n \pm \sqrt{d_n^2 - 2m_{red}\delta_n(0) f_n}}{f_n} \tag{5.131}$$

$$= -\frac{d_n}{f_n} \pm \text{sign}(f_n)\sqrt{\left(\frac{d_n}{f_n}\right)^2 - \frac{m_{red}\delta_n(0)}{f_n}} \tag{5.132}$$

The velocity at time of contact is

$$\frac{d\delta_n(t_c)}{dt} = \frac{d_n}{m_{red}} + \frac{f_n t_c}{m_{red}} < 0 \quad \text{or} \quad d_n^{(c)} = d_n + f_n t_c \tag{5.133}$$

These results imply that one or the other of d_n and f_n must be negative (or both). The three possibilities are:

$$\begin{array}{lcccl} d_n < 0 \text{ and } f_n < 0 & \Longrightarrow & -\frac{d_n}{f_n} < 0 (\text{sign}(f_n) = -1) & \Longrightarrow & \text{use - root} \\ d_n > 0 \text{ and } f_n < 0 & \Longrightarrow & -\frac{d_n}{f_n} < t_c & \Longrightarrow & \text{use - root} \\ d_n < 0 \text{ and } f_n > 0 & \Longrightarrow & -\frac{d_n}{f_n} > t_c & \Longrightarrow & \text{use - root} \end{array} \tag{5.134}$$

The contact velocity becomes:

$$m_{red}\frac{d\delta_n(t_c)}{dt} = -\sqrt{d_n^2 - m_{red}\delta_n(0) f_n} \tag{5.135}$$

A special case when $f_n = 0$, which implies $d_n < 0$, and leads to

$$t_c = -\frac{m_{red}\delta_n(0)}{d_n} \qquad \text{and} \qquad \frac{d\delta_n(t_c)}{dt} = \frac{d_n}{m_{red}} \quad \text{or} \quad d_n^{(c)} = d_n \tag{5.136}$$

In summary, for all possibilities:

$$t_c = \begin{cases} -\frac{m_{red}\delta_n(0)}{d_n} & f_n = 0 \\ \frac{-d_n \pm \sqrt{d_n^2 - 2m_{red}\delta_n(0) f_n}}{f_n} & \text{otherwise} \end{cases} \tag{5.137}$$

$$d_n^{(c)} = d_n + f_n t_c = d_n' - f_n \Delta t' = -\sqrt{d_n^2 - m_{red}\delta_n(0) f_n} \tag{5.138}$$

where $\Delta t' = \Delta t - t_c$. To finish the step, proceed to post-contact methods.

5.6.2 Post Contact when Debonded Interface

For finite compression stiffness, the interfacial force and displacement from Eq. (5.104) and previous section become

$$\begin{aligned} f_{n,a}^{(INT)} &= \frac{2}{\Delta t}\left[m\delta_n(0) + d'_n\left(1 - \frac{\Delta t'}{\Delta t}\frac{\sin\phi'}{\phi'}\right)\right] + f_n\left[2\left(\frac{\Delta t'}{\Delta t}\right)^2\left(\frac{\sin\phi'}{\phi'} - \frac{(1-\cos\phi')}{(\phi')^2}\right) - 1\right] \\ \delta_t(\Delta t) &= \frac{\Delta t'}{\Delta t}\left[\frac{d'_n}{m}\frac{\sin\phi'}{\phi'} + \frac{f_n\Delta t'}{m}\left(\frac{1-\cos\phi'}{(\phi')^2} - \frac{\sin\phi'}{\phi'}\right)\right] \end{aligned} \tag{5.139}$$

where $\phi' = \kappa_{nc}\Delta t'$. This result reduces to linear interface (see Eq. (5.45)) when $t_c = 0$, $\Delta t' = \Delta t$, $\phi' = \phi$, and $\delta_n(0) = 0$ (this last because contact occurred at time zero).

5.6.3 Post Contact When Perfect in Compression

If compression is a perfect interface, the contact force would be zero up to the point of contact. We then assume that there is no penetration and that it does not bounce off within this time step. In other words $\delta_n(t) = 0$ for $t > t_c$. For it to remain zero, Eq. (5.92) implies that:

$$\int_0^{\Delta t'} dt_2 \int_0^{t_2} dt_1 F_n(0) = d_n^{(c)}\Delta t' + \frac{f_n(\Delta t')^2}{2} \tag{5.140}$$

where $\Delta t' = \Delta t - t_c$ or the time interval after contact and hypothetical force is evaluated only at $\delta_n(t) = 0$. The final separation during the time step become:

$$\delta_n(t) = \begin{cases} \delta(0) + \frac{d_n t}{m_{red}} + \frac{f_n t^2}{2m_{red}} & t < t_c \\ 0 & t > t_c \end{cases} \tag{5.141}$$

The interfacial force to mimic this evolution in the MPM time step would be

$$f_{n,a}^{(INT)} = f_n + \frac{2}{\Delta t}\left(\frac{m_{red}\delta_n(0)}{\Delta t} + d_n\right) = \frac{2}{\Delta t}\left(\frac{m_{red}\delta_n(0)}{\Delta t} + d'_n\right) - f_n \tag{5.142}$$

This result is also found by taking limit of Eq. (5.139) as $\phi' \to \infty$. This result does not depend on the contact time, t_c. Note that this could be alternative force for contact if done after the momentum update. The current method finds $f_{n,a}^{(INT)}\Delta t = d'_n$. The new momentum change (added to particle momentum and force) would be

$$\Delta\boldsymbol{p}'_a \cdot \hat{\boldsymbol{n}} = f_{n,a}^{(INT)}\Delta t = \frac{2m_{red}\delta_n(0)}{\Delta t} + 2d_n + f_n\Delta t = d'_n + d_n + \frac{2m_{red}\delta_n(0)}{\Delta t} \tag{5.143}$$

5.6.4 Other?

For interface starting in compression and moving into tension, repeat the above analysis but interchange tension and compression interface properties.

For interface starting perfect ($D < 0$ in `OSParticulas`, it stays perfect as stick contact). For imperfect interface moving into contact, can apply force to reach contact and then adjust momentum to stick after contact using $d_n^{(c)}$. In other words, change input momentum change to the new value at the point of contact and also apply the force.

5.7 Use Imperfect Interface to Soften Contact Calculations

Contact calculations treat debonded as having zero stiffness with contact has infinite stiffness. This abrupt change, could be interpreted as a bilinear interface leading to following revised contact algorithm:

1. Find displacement at end of current time step as

$$\delta_n(\Delta t) = \delta(0) + \frac{1}{m}\left(d'_n - \frac{f_n \Delta t}{2}\right) \tag{5.144}$$

   where $m = m_{red}/\Delta t$. But, if not in contact calculations just after momentum updates, simply set $\delta_n(\Delta t) = \delta_n(0)$.

2. Because interface is considered as perfect interface when in contact, we set the interfacial normal force to be

$$f_{n,a}^{(INT)} \Delta t = d'_n \tag{5.145}$$

   which is simply d_n when not just after momentum updates. It is possible force should use perfect interface results instead or

$$f_{n,a}^{(INT)} = 2\left(\frac{m_{red}\delta_n(0)}{\Delta t} + d'_n\right) - f_n \Delta t \tag{5.146}$$

   although some trials with this force did not look as good.

3. Find tangential interface force to stick or force if was perfect interface in shear

$$f_{n,t}^{(INT)} \Delta t = d'_t \tag{5.147}$$

4. If interface has adhesion, then if $\delta_n(\Delta t) > 0$, check failure based on traction stresses implied by $f_{n,a}^{(INT)}$ and $f_{n,t}^{(INT)}$. If they say adhesion not failure, the contact treated as stick conditions, but if they say failed, treat as not in contact with no momentum changes.

5. If $\delta_n(\Delta t) > 0$ the interface is not in contact and no momentum changes are done. If $\delta_n(\Delta t) < 0$, the two option are to a) assume in contact, apply $f_{n,a}^{(INT)}$ to node, and use contact low to find final tangential force. b) check sign of $f_{n,a}^{(INT)}$. If positive, assume not in contact, but if negative, assume compression and proceed like option a. Option "a" is the new "displacementOnly" option in code. Note that this option might end up applying tensile force to the node even if no tangential force. Option "b" never applies tensile normal force to the node.

A revised used of imperfect interfaces for revised contact is the find interfacial force and displacement using a finite contact stiffness as an imperfect interface parameter D_c for linear contact stiffness. The following algorithm only applies in contact calculations just after momentum update when interfacial force is known. First, find $m = m_{red}/\Delta t$, $d = D_c A_c \Delta t$, and $\phi = \sqrt{d/m}$

1. If $\phi > \pi/2$, contact is treated as perfect and finds terms using above contact algorithm or:

$$\delta_n(\Delta t) = \delta(0) + \frac{1}{m}\left(d'_n - \frac{f_n \Delta t}{2}\right) \quad \text{and} \quad f_{n,a}^{(INT)} \Delta t = d'_n \tag{5.148}$$

but, it $\phi < \pi/2$, find these terms instead

$$\delta_n(\Delta t) = \delta_n(0)\cos\phi + \frac{d'_n}{m}\frac{\sin\phi}{\phi} + \frac{f_n\Delta t}{m}\left(\frac{1-\cos\phi}{\phi^2} - \frac{\sin\phi}{\phi}\right) \tag{5.149}$$

$$f_{n,a}^{(INT)}\Delta t = 2\left[m\big(\delta_n(0) - \delta_n(\Delta t)\big) + d'_n\right] - f_n\Delta t \tag{5.150}$$

2. Proceed to step 3 of previous algorithm. Only $\delta_n(\Delta t)$ and $f_{n,a}^{(INT)}\Delta t$ may have changed.

3. Contact calculations done other places can use the previous algorithm with no interface calculations.

Chapter 6

MPM Time Step Tasks For Perfect Interface

As a check/evaluation of contact methods, this section contrasts the MPM time step tasks for a single-material mode calculation to a simulation with two identical with contact law that assumes a perfect interface (*e.g.*, the `IgnoreContact` law) and with no cracks.

6.1 Simulation and Velocity Field Initialization

Each nodal point is permanently assigned crack field [0]. When in single material velocity field, field [0] is given a single material velocity field [0]. When using multimaterial mode, crack velocity field [0] is assigned pointers to material velocity fields for materials a and b. The pointers are initialized to `NULL`. Any material velocity fields needed in a simulation are dynamically allocated as needed.

6.2 Momentum and Mass Extrapolation to the Grid

In single material mode, there is only one material velocity field with nodal values:

$$m_i = \sum_p S_{ip} m_p \quad \text{and} \quad \boldsymbol{p}_i = m_i \boldsymbol{v}_i = \sum_p S_{ip} \boldsymbol{p}_p \tag{6.1}$$

In multimaterial mode

$$m_{i,j} = \sum_{p \in j} S_{ip} m_p \quad \text{and} \quad \boldsymbol{p}_{i,j} = m_{i,j} \boldsymbol{v}_{i,j} = \sum_{p \in j} S_{ip} \boldsymbol{p}_p \tag{6.2}$$

Because $j = a$ or b only here

$$m_i = m_{i,a} + m_{i,b} \quad \text{and} \quad \boldsymbol{p}_i = \boldsymbol{p}_{i,a} + \boldsymbol{p}_{i,b} \quad \text{but} \quad \boldsymbol{v}_i \neq \boldsymbol{v}_{i,a} \neq \boldsymbol{v}_{i,b} \tag{6.3}$$

Multimaterial mode does more extrapolations for position, volume, and volume gradient, but when using `IgnoreContact`, those quantities are not used.

6.3 Revised XPIC Task

If using revised XPIC(m) for $m > 1$, the mass an momentum tasks is followed by XPIC tasks that replaces grid velocities with $\boldsymbol{v}_i(m) = m(\boldsymbol{v}_i - \boldsymbol{v}_i^*)$. In vector form:

$$\boldsymbol{v}(m) = \sum_{k=1}^{m} (-1)^{k+1} \boldsymbol{v}_k \tag{6.4}$$

where

$$\boldsymbol{v}_k = \frac{m-k+1}{k} \mathbf{S}^{+} \mathbf{S} \boldsymbol{v}_{k-1} \tag{6.5}$$

starting with

$$\boldsymbol{v}_1 = m\boldsymbol{v} \tag{6.6}$$

In single material mode, the explicit sums are:

$$(\boldsymbol{v}_k)_i = \frac{m-k+1}{k} \sum_p \sum_{j \in \Omega_p} \frac{M_p S_{pi} S_{pj}}{m_i} (\boldsymbol{v}_{k-1})_j \qquad \forall\ i \in \Omega_p \tag{6.7}$$

In multimaterial mode, $\boldsymbol{v}(m)$ will be found in each material velocity filed. Explicit sums for one material are:

$$(\boldsymbol{v}_k)_{i,a} = \frac{m-k+1}{k} \sum_{p \in a} \sum_{j \in \Omega_p} \frac{M_p S_{pi} S_{pj}}{m_{i,a}} (\boldsymbol{v}_{k-1})_{j,a} \qquad \forall\ i \in \Omega_p \tag{6.8}$$

Single material results when modeling two materials can be written:

$$m_i(\boldsymbol{v}_k)_i = \frac{m-k+1}{k} \sum_p \sum_{j \in \Omega_p} M_p S_{pi} S_{pj} \frac{(\boldsymbol{p}_{k-1})_{j,a} + (\boldsymbol{p}_{k-1})_{j,b}}{m_{j,a} + m_{j,b}} \tag{6.9}$$

Note that prior to XPIC task that

$$m_i \boldsymbol{v}_i = \boldsymbol{p}_i = \boldsymbol{p}_{i,a} + \boldsymbol{p}_{i,b} = m_{i,a}\boldsymbol{v}_{i,a} + m_{i,b}\boldsymbol{v}_{i,b} \tag{6.10}$$

but after replacing grid velocities with $\boldsymbol{v}(m)$, it appears that

$$m_i \boldsymbol{v}(m)_i \neq m_{i,a}\boldsymbol{v}(m)_{i,a} + m_{i,b}\boldsymbol{v}(m)_{i,b} \tag{6.11}$$

If true, multimaterial mode would not be able to reduce to single material mode for the case of a perfect interface.

6.4 Post Momentum and Mass Extrapolation Tasks

After the extrapolations are done, each node is revisited for more calculations to impose multimaterial effects (in that mode). The steps are

1. Calculate total mass and number of particles in each velocity field. In the process copy $\boldsymbol{p}_i$ and $\boldsymbol{p}_{i,j}$ for all material velocity fields on all nodes. Also make a list of nodes that have both materials as list of `MaterialContactNode` objects.

2. In multimaterial mode, change momenta for fields a and b by

$$\pm\Delta\boldsymbol{p}_a = \pm\left(\frac{m_a}{m_i}\boldsymbol{p}_i - \boldsymbol{p}_a\right) \tag{6.12}$$

Adding to material a (and subtracting from b) leads to

$$\boldsymbol{p}'_{i,a} = \frac{m_{i,a}}{m_i}\boldsymbol{p}_i = m_{i,a}\boldsymbol{v}_i \quad \text{and} \quad \boldsymbol{p}'_{i,a} = \frac{m_{i,b}}{m_i}\boldsymbol{p}_i = m_{i,b}\boldsymbol{v}_i \tag{6.13}$$

Note that materials a and b now have the same velocity ($= \boldsymbol{v}_i$) but have different momenta. These changes are not done be setting new momenta based of $\boldsymbol{v}_i$, but by using

$$\Delta\boldsymbol{p}_a = \frac{m_{i,a}\boldsymbol{p}_i - m_i\boldsymbol{p}_{i,a}}{m_i} = \frac{m_{i,a}\boldsymbol{p}_{i,b} - m_{i,b}\boldsymbol{p}_{i,a}}{m_i} \tag{6.14}$$

The velocity for material a (for example) would be

$$\boldsymbol{v}_{i,a} = \frac{\boldsymbol{p}_{i,a}}{m_{i,a}} + \frac{m_{i,a}\boldsymbol{p}_{i,b} - m_{i,b}\boldsymbol{p}_{i,a}}{m_{i,a}m_i} \tag{6.15}$$

Perhaps this equation would lose accuracy when $m_{i,a} << m_{i,b}$?

3. Note that $\boldsymbol{p}'_{i,a}$ replaces the store momenta (to be restored later) and the force is seeded with

$$\boldsymbol{f}_{i,a} = -\frac{\Delta\boldsymbol{p}_a}{\Delta t} \tag{6.16}$$

4. Impose velocity boundary conditions by setting $\boldsymbol{v}_i = \boldsymbol{v}^{(BC)}$, which which would changed a and b in multimaterial mode.

6.5 Update Stresses and Strains on Particle

When updating stresses and strains before finding forces and updating momenta (needed for `USAVG`$\pm$ and for `USF` update methods), extrapolate velocity gradient ($\nabla\boldsymbol{v}$) to the particles and implement the chosen constitutive laws. The extrapolations are:

$$\nabla\boldsymbol{v}_p = \sum_i \boldsymbol{v}_i \otimes \boldsymbol{G}_{ip} \quad \text{and} \quad \nabla\boldsymbol{v}_{p\in j} = \sum_i \boldsymbol{v}_{i,j} \otimes \boldsymbol{G}_{ip} \tag{6.17}$$

Because $\boldsymbol{v}_{i,a} = \boldsymbol{v}_{i,b} = \boldsymbol{v}_i$ for a perfect interface, both modes should extrapolate the same velocity to the particles resulting in same updates of stress and strain on particles.

6.6 Project Forces to the Grid

For each material point, extrapolate $f_{i,j}$ due to particle stresses particle forces (later not used often):

$$\boldsymbol{f}_{i,j} = \sum_{p\in j}\left(-m_p\frac{\boldsymbol{\tau}_p^{(n)}\cdot\boldsymbol{G}_{ip}^{(n)}}{\rho_0} + \boldsymbol{F}_p^{(n)}S_{ip}^{(n)}\right) \tag{6.18}$$

Because $j = a$ or b only here, and assuming particle stresses remain in synch, we have total nodal force as:

$$\boldsymbol{f}_i = \boldsymbol{f}_{i,a} + \boldsymbol{f}_{i,b} \quad \text{but} \quad \boldsymbol{a}_i = \frac{\boldsymbol{f}_i}{m_i} \neq \boldsymbol{a}_{i,a} \neq \boldsymbol{a}_{i,b} \tag{6.19}$$

After the force extrapolations are done, the following calculations complete the force calculations:

1. Restore momentum to the grid, but at multimaterial nodes, it will restore $\boldsymbol{p}_{i,j}$, which are momenta after contact laws changes done above, but before boundary conditions.

2. Add traction loads on particles.

$$\boldsymbol{f}_{i,T,j} = \int_{S_{T,j}} \boldsymbol{N}_i(\boldsymbol{x})\boldsymbol{T}\, dS \tag{6.20}$$

3. Add any body forces (such as gravity) assuming independent of particle state

$$\boldsymbol{f}_{i,b,j} = \sum_{p\in j} m_p S_{ip}^{(n)} \boldsymbol{b}_p = m_{i,j}\boldsymbol{b}_i \tag{6.21}$$

   Because $j = a$ or b only here, total body force is sum of force on the two materials.

4. For nodes with velocity boundary conditions, set the force such that the momenta restored in step #1 above will update to the specified velocity.

6.7 XPIC Task

The key XPIC task is to find

$$\boldsymbol{v}^* = \sum_{k=2}^{m} (-1)^k \boldsymbol{v}_k^* \tag{6.22}$$

where

$$\boldsymbol{v}_1^* = \boldsymbol{v} \tag{6.23}$$

and

$$(\boldsymbol{v}_k^*)_i = \frac{m-k+1}{k} \sum_p \sum_{j\in\Omega_p} \frac{M_p S_{pi} S_{pj}}{m_i^{(n)}} (\boldsymbol{v}_{k-1}^*)_j \qquad \forall\ i \in \Omega_p \tag{6.24}$$

In multimaterial mode, each material is done separately. But, if the interface and perfect, interface nodes will have $\boldsymbol{v}_a = \boldsymbol{v}_b = \boldsymbol{v}_i$. This state means that $\boldsymbol{v}_1^* = \boldsymbol{v}$ in both material domains. But, then separate domains will have:

$$(\boldsymbol{v}_k^*)_{i,a} = \frac{m-k+1}{k} \sum_{p\in a} \sum_{j\in\Omega_p} \frac{M_p S_{pi} S_{pj}}{m_i^{(n)}} (\boldsymbol{v}_{k-1}^*)_j \qquad \forall\ i \in \Omega_p \tag{6.25}$$

$$(\boldsymbol{v}_k^*)_{i,b} = \frac{m-k+1}{k} \sum_{p\in b} \sum_{j\in\Omega_p} \frac{M_p S_{pi} S_{pj}}{m_i^{(n)}} (\boldsymbol{v}_{k-1}^*)_j \qquad \forall\ i \in \Omega_p \tag{6.26}$$

$$(\boldsymbol{v}_k^*)_i = (\boldsymbol{v}_k^*)_{i,a} + (\boldsymbol{v}_k^*)_{i,b} \tag{6.27}$$

$$\boldsymbol{v}^* = \sum_{k=2}^{m} (-1)^k \big((\boldsymbol{v}_k^*)_a + (\boldsymbol{v}_k^*)_b\big) = \boldsymbol{v}_a^* + \boldsymbol{v}_b^* \tag{6.28}$$

6.8 Update Nodal Momenta

This task updates the nodal momenta. In single material mode:

$$\boldsymbol{p}_i^+ = \boldsymbol{p}_i + \boldsymbol{f}_i \Delta t \tag{6.29}$$

In multimaterial mode

$$\boldsymbol{p}^+_{i,j} = \boldsymbol{p}'_{i,j} + \boldsymbol{f}'_{i,j}\Delta t \quad \text{where} \quad \boldsymbol{f}'_{i,j} = \boldsymbol{f}_{i,j} - \frac{\Delta \boldsymbol{p}_{i,j}}{\Delta t} \tag{6.30}$$

includes $\boldsymbol{f}_{i,j}$ as force due to stress, force, traction, and body force extrapolations and the subtracted force is due to earlier contact calculations. Note that we can also write

$$\boldsymbol{p}^+_{i,j} = \boldsymbol{p}_{i,j} + \boldsymbol{f}_{i,j}\Delta t \tag{6.31}$$

For any interface we will recover

$$\boldsymbol{p}^+_i = \boldsymbol{p}^+_{i,a} + \boldsymbol{p}^+_{i,b} \tag{6.32}$$

including at velocity boundary conditions.

To keep contact correct, revisit contact law calculations in section 6.2, find contact momentum change $\Delta \boldsymbol{p}^+_{i,j}$ and update force to match with:

$$\Delta \boldsymbol{f}_{i,j} = \frac{\Delta \boldsymbol{p}^+_{i,j}}{\Delta t} \tag{6.33}$$

For this special case of perfect interface, material a will have

$$\Delta \boldsymbol{p}^+_{i,a} = \frac{m_{i,a}}{m_i}\boldsymbol{p}^+_i - \boldsymbol{p}^+_{i,a} \tag{6.34}$$

$$\boldsymbol{p}^{+'}_{i,a} = \frac{m_{i,a}}{m_i}\boldsymbol{p}^+_i \tag{6.35}$$

The revised material a (and similar for b) will be

$$\boldsymbol{p}^{+'}_{i,j} = \boldsymbol{p}'_{i,j} + \boldsymbol{f}''_{i,j}\Delta t = \boldsymbol{p}'_{i,j} + \left(\boldsymbol{f}_{i,a} + \frac{\Delta \boldsymbol{p}^+_{i,a} - \Delta \boldsymbol{p}_{i,a}}{\Delta t}\right)\Delta t \tag{6.36}$$

The velocity in both modes are the same ($\boldsymbol{v}_i = \boldsymbol{v}_{i,j}$). The acceleration should be based on $\boldsymbol{f}''_{i,j}/m_{i,j}$. For material a and perfect interface

$$\boldsymbol{a}_{i,a} = \frac{1}{m_{i,a}}\left(\boldsymbol{f}_{i,a} + \frac{\Delta \boldsymbol{p}^+_{i,a} - \Delta \boldsymbol{p}_{i,a}}{\Delta t}\right) \tag{6.37}$$

$$= \frac{1}{m_{i,a}}\left(\boldsymbol{f}_{i,a} + \frac{m_{i,a}}{m_i}\left(\frac{\boldsymbol{p}_i}{\Delta t} + \boldsymbol{f}_i\right) - \left(\frac{\boldsymbol{p}_{i,a}}{\Delta t} + \boldsymbol{f}_{i,a}\right) - \frac{m_{i,a}}{m_i}\frac{\boldsymbol{p}_i}{\Delta t} + \frac{\boldsymbol{p}_{i,a}}{\Delta t}\right) \tag{6.38}$$

$$= \frac{\boldsymbol{f}_i}{m_i} \tag{6.39}$$

which is correct result for a perfect interface

These contact calculations might disrupt velocity boundary conditions so they need to be reimposed. If momentum is changed, it should change force as well. Actually force started this step in synch with boundary conditions, but contact calculations might add $\Delta f_{i,j}$ that disrupts it, so it needs to be brought back in line.

6.9 Update the Particles

Update particle position and velocity using nodal velocity (from $\boldsymbol{v}_{i,j} = \boldsymbol{p}^+_{i,j}/m_{i,j}$) and acceleration (from $\boldsymbol{a}_{i,j} = \boldsymbol{f}_{i,j}/m_{i,j}$) found in material velocity field `matfld` (for type of particle). For a perfect interface, we should have (within round-off error) $\boldsymbol{v}_{i,j} = \boldsymbol{v}_i$ and $\boldsymbol{a}_{i,j} = \boldsymbol{a}_i$, meaning the particle update should be same for single and multimaterial modes.

6.10 Optional Update Strain Last

Optionally the calculations can update stain after the momentum update. This calculation is done for methods `USAVG`± and for `USL`±. For a perfect interface, these should continue to see single an multimaterial mode have the same velocity and therefore results should be identical.

Bibliography

[1] C. C. Hammerquist and J. A. Nairn. Modeling nanoindentation using the material point method. *Journal of Materials Research*, 33:1369–1381, 05 2018.

[2] J. A. Nairn. Numerical implementation of imperfect interfaces. *Computational Materials Science*, 40:525–536, 2007.

[3] J. A. Nairn, S. G. Bardenhagen, and G. S. Smith. Generalized contact and improved frictional heating in the material point method. *Computational Particle Mechanics*, 5(3):285–296, 2018.