Full Mass Matrix MPM or FMPM

John A. Nairn

December 5, 2023

Contents

Chapter 1

MPM Extrapolations

1.1 Grid To Particle Extrapolations

Let lower case denote grid (or Eulerian frame) quantities and upper case denote particle (or Lagrangian frame) quantities. We define S to be a matrix of grid-based shape functions to extrapolate from nodes to any position (normally to a particle position). Thus:

$$\boldsymbol{Q}_p = \sum_i S_{pi}\boldsymbol{q}_i \qquad \text{and} \qquad 1 = \sum_i S_{pi} \qquad \text{and} \qquad \boldsymbol{Q} = \mathsf{S}\boldsymbol{q} \tag{1.1}$$

The first extrapolates any nodal quantity, q_i, to a particle quantity, Q_p, and S_{pi} is shape function for node i at position of particle p. The second requires partition of unity for the shape functions. The third write is as a linear transformation where $\boldsymbol{Q}$ is vector of particle properties, $\boldsymbol{q}$ is vector of nodal quantities, and S is the $N \times n$ transformation matrix (N particles and n nodes).

1.2 Particle to Grid Extrapolations

Extrapolating properties from particle to grid would like inverse of S. Becase S is not square, we need to use a pseudo inverse denoted here by S^+ and derived in original Sulsky MPM paper using weighted least squares. Imagine some grid velocity give by $\boldsymbol{v}$ and them extrapolate then to the particles using $\tilde{\boldsymbol{V}} = \mathsf{S}\boldsymbol{v}$. We find S^+ such that $\boldsymbol{v} = \mathsf{S}^+\boldsymbol{V}$ minimizes squared errors between $\boldsymbol{V}$ and $\tilde{\boldsymbol{V}}$. A mass-weighted error minimization is:

$$\Omega = \sum_p M_p\big(\boldsymbol{V}_p - \tilde{\boldsymbol{V}}_p\big)^2 = \sum_p M_p\big(\boldsymbol{V}_p - \sum_i S_{pi}\boldsymbol{v}_i\big)^2 \tag{1.2}$$

$$0 = \frac{d\Omega}{d\boldsymbol{v}_j} = -2\left(\sum_p S_{pj}M_p\boldsymbol{V}_p - \sum_i\sum_p M_p S_{pj}S_{pi}\boldsymbol{v}_i\right) \tag{1.3}$$

$$\sum_i \tilde{m}_{ji}\boldsymbol{v}_i = \sum_p S^T_{jp}M_p\boldsymbol{V}_p \tag{1.4}$$

where $\tilde{m}_{ji} = \sum_p M_p S^T_{jp}S_{pi}$ is an element of the full (and symmetric) $n \times n$ mass matrix $\tilde{\mathsf{m}}$. In matrix form:

$$\tilde{\mathsf{m}}\boldsymbol{v} = \mathsf{S}^T\boldsymbol{P}, \qquad \boldsymbol{v} = \tilde{\mathsf{m}}^{-1}\mathsf{S}^T\boldsymbol{P} = \tilde{\mathsf{S}}^+\boldsymbol{V}, \qquad \text{with} \qquad \tilde{\mathsf{S}}^+ = \tilde{\mathsf{m}}^{-1}\mathsf{S}^T\mathsf{M} \tag{1.5}$$

where $\boldsymbol{P}$ is a vector of particle momenta and $\mathsf{M} = \text{diag}(\boldsymbol{M})$ is $N \times N$ diagonal matrix formed from vector $\boldsymbol{M}$ of particle masses (when using same letter, serif form is a vector and sans-serif is a matrix). The $\tilde{}$ on $\tilde{\mathsf{m}}$ and $\tilde{\mathsf{S}}^+$ indicate terms using the full mass matrix.

In standard MPM, $\tilde{\mathsf{m}}$ is replaced m, which is a diagonal lumped mass matrix, with i^{th} diagonal element defined by:

$$m_i = \sum_j \tilde{m}_{ji} = \sum_p S_{pi} M_p = \sum_p S_{ip}^T M_p \quad \text{or} \quad \boldsymbol{m} = \mathsf{S}^T \boldsymbol{M} \quad \text{and} \quad \mathsf{m} = \text{diag}(\boldsymbol{m}) \tag{1.6}$$

where $\boldsymbol{m}$ is vector of nodal masses. The pseudo inverse matrix for a lumped mass matrix (without $\tilde{}$) is:

$$\mathsf{S}^+ = \mathsf{m}^{-1}\mathsf{S}^T\mathsf{M} \quad \Longrightarrow \quad S_{ip}^+ = \sum_k \sum_l \frac{\delta_{ik}}{m_i} S_{kl}^T \delta_{lp} M_p = \frac{S_{pi} M_p}{m_i} \tag{1.7}$$

1.3 Full Mass Matrix Expansion

To write full matrix matrix in new form, we note that the $(jp)^{th}$ element of $\mathsf{S}^T\mathsf{M}$ is:

$$(\mathsf{S}^T\mathsf{M})_{jp} = \sum_q S_{jq}^T \mathsf{M}_{qp} = \sum_q S_{jq}^T M_p \delta_{qp} = M_p S_{jp}^T \tag{1.8}$$

The full mass matrix can then be written as:

$$(\mathsf{S}^T\mathsf{M}\mathsf{S})_{ji} = \sum_p (\mathsf{S}^T\mathsf{M})_{jp} S_{pi} = \tilde{m}_{ji} \qquad \Longrightarrow \qquad \tilde{\mathsf{m}} = \mathsf{S}^T\mathsf{M}\mathsf{S} \tag{1.9}$$

Using S^+ with lumped mass matrix, the full mass matrix is

$$\tilde{\mathsf{m}} = \mathsf{m}\mathsf{S}^+\mathsf{S} = \mathsf{m}\big(\mathsf{I}_n - (\mathsf{I}_n - \mathsf{S}^+\mathsf{S})\big) = \mathsf{m}\big(\mathsf{I}_n - \mathsf{A}\big) \tag{1.10}$$

where I_n is an $n \times n$ identity matrix and

$$\mathsf{A} = \mathsf{I}_n - \mathsf{S}^+\mathsf{S} \tag{1.11}$$

The full mass matrix inverse is then

$$\tilde{\mathsf{m}}^{-1} = (\mathsf{I}_n - \mathsf{A})^{-1}\mathsf{m}^{-1} = (\mathsf{I}_n + \mathsf{A} + \mathsf{A}^2 + \mathsf{A}^3 + \cdots)\mathsf{m}^{-1} \tag{1.12}$$

Using one term is lumped matrix inverse and additional terms approximate the full mass matrix better.

The psuedo-inverse shape function matrix using full mass matrix expanded with k terms is:

$$\tilde{\mathsf{S}}_k^+ = \big(\mathsf{I}_n + (\mathsf{I}_n - \mathsf{S}^+\mathsf{S}) + (\mathsf{I}_n - \mathsf{S}^+\mathsf{S})^2 + \cdots + (\mathsf{I}_n - \mathsf{S}^+\mathsf{S})^{k-1}\big)\mathsf{S}^+ \tag{1.13}$$

Increasing order approximations to $\tilde{\mathsf{S}}^+$ would be

$$\tilde{\mathsf{S}}_1^+ = \mathsf{S}^+ \tag{1.14}$$

$$\tilde{\mathsf{S}}_2^+ = \big(2\mathsf{I}_n - \mathsf{S}^+\mathsf{S}\big)\mathsf{S}^+ \tag{1.15}$$

$$\vdots \tag{1.16}$$

$$\tilde{\mathsf{S}}_k^+ = \left(\mathsf{I}_n + \sum_{j=1}^{k-1}\sum_{\ell=0}^{j} (-1)^\ell \binom{j}{\ell} \big(\mathsf{S}^+\mathsf{S}\big)^\ell\right)\mathsf{S}^+ \tag{1.17}$$

For more efficient implementation, we rearrange:

$$\tilde{\mathsf{S}}_k^+ = \left(k\mathsf{I}_n + \sum_{\ell=1}^{k-1}\sum_{j=\ell}^{k-1}(-1)^\ell \binom{j}{\ell} \left(\mathsf{S}^+\mathsf{S}\right)^\ell \right)\mathsf{S}^+ \tag{1.18}$$

$$= \left(k\mathsf{I}_n + \sum_{\ell=2}^{k}\sum_{j=\ell}^{k}(-1)^{\ell+1} \binom{j-1}{\ell-1} \left(\mathsf{S}^+\mathsf{S}\right)^{\ell-1} \right)\mathsf{S}^+ \tag{1.19}$$

$$= \left(k\mathsf{I}_n + \sum_{\ell=2}^{k} C_{k\ell} \left(\mathsf{S}^+\mathsf{S}\right)^{\ell-1} \right)\mathsf{S}^+ = \left(\sum_{\ell=1}^{k} C_{k\ell} \left(\mathsf{S}^+\mathsf{S}\right)^{\ell-1} \right)\mathsf{S}^+ \tag{1.20}$$

where

$$C_{k\ell} = (-1)^{\ell+1} \sum_{j=\ell}^{k} \binom{j-1}{\ell-1} \qquad \text{and} \qquad C_{k1} = k \tag{1.21}$$

1.4 Velocity Extrapolation

Velocity extrapolation with approximate full mass matrix becomes:

$$\boldsymbol{v} = \left(\sum_{\ell=1}^{k} C_{k\ell} \left(\mathsf{S}^+\mathsf{S}\right)^{\ell-1} \right)\mathsf{m}^{-1}\mathsf{S}^T\mathsf{M}\boldsymbol{V} = \tilde{\mathsf{m}}_k^{-1}\boldsymbol{p} \tag{1.22}$$

where

$$\tilde{\mathsf{m}}_k^{-1} = \left(\sum_{\ell=1}^{k} C_{k\ell} \left(\mathsf{S}^+\mathsf{S}\right)^{\ell-1} \right)\mathsf{m}^{-1} \tag{1.23}$$

We can simplify C_{kl} using the binomial theorem or

$$\binom{k}{\ell} = \binom{k-1}{\ell-1} + \binom{k-1}{\ell} \tag{1.24}$$

$$= \binom{k-1}{\ell-1} + \binom{k-2}{\ell-1} + \binom{k-2}{\ell} \tag{1.25}$$

$$= \binom{k-1}{\ell-1} + \binom{k-2}{\ell-1} + \binom{k-3}{\ell-1} + \binom{k-3}{\ell} \tag{1.26}$$

$$\vdots \tag{1.27}$$

$$= \binom{k-1}{\ell-1} + \binom{k-2}{\ell-1} + \binom{k-3}{\ell-1} + \cdots + \binom{\ell-1}{\ell-1} \tag{1.28}$$

$$= \sum_{j=\ell}^{k} \binom{j-1}{\ell-1} \tag{1.29}$$

In other words:

$$C_{kl} = (-1)^{\ell+1}\binom{k}{\ell} \tag{1.30}$$

We can also show by reverse induction. The relation holds for $\ell = k$. Now assume it holds for ℓ and prove it holds for $\ell - 1$:

$$\binom{k}{\ell-1} = \binom{k+1}{\ell} - \binom{k}{\ell} = \sum_{j=\ell}^{k+1}\binom{j-1}{\ell-1} - \sum_{j=\ell}^{k}\binom{j-1}{\ell-1} = \binom{k}{\ell-1} \quad \checkmark \tag{1.31}$$

Now compare to XPIC(k) (using k instead of m for XPIC order), projection onto non-null-space of velocities uses projector:

$$\mathsf{P}_k = \mathsf{I}_N - (\mathsf{I}_N - \mathsf{S}\mathsf{S}^+)^k \tag{1.32}$$

where I_N is an $N \times N$ identity matrix. The projection expands to:

$$\begin{aligned}
\mathsf{P}_k \boldsymbol{V} &= \sum_{\ell=1}^{k} (-1)^{\ell+1} \binom{k}{\ell} (\mathsf{S}\mathsf{S}^+)^\ell \boldsymbol{V} && (1.33)\\
&= \mathsf{S}\left(\sum_{\ell=1}^{k} (-1)^{\ell+1} \binom{k}{\ell} (\mathsf{S}^+\mathsf{S})^{\ell-1} \right) \mathsf{S}^+ \boldsymbol{V} && (1.34)\\
&= \mathsf{S}\left(\sum_{\ell=1}^{k} (-1)^{\ell+1} \binom{k}{\ell} (\mathsf{S}^+\mathsf{S})^{\ell-1} \right) \mathsf{m}^{-1}\mathsf{S}^T\mathsf{M}\boldsymbol{V} && (1.35)\\
&= \mathsf{S}\tilde{\mathsf{m}}_k^{-1}\boldsymbol{p} = \mathsf{S}\boldsymbol{v}(k) && (1.36)
\end{aligned}$$

In other words, $\boldsymbol{v}(k)$ defined in `Extrap.pdf` notes is identical to $\tilde{\mathsf{m}}_k^{-1}\boldsymbol{p}$. XPIC($k$) should be interpreted as finding grid velocities using an approximate full mass matrix. By using XPIC(k) methods other places in the MPM time step, the entire method can be cast as an approximate full mass matrix method.

1.4.1 Another Comparison to XPIC

Using geometric series, we can show that

$$(\mathsf{I}_n + \mathsf{A} + \mathsf{A}^2 + \mathsf{A}^3 + \cdots + \mathsf{A}^{k-1})\big(\mathsf{I}_n - \mathsf{A}\big) = \mathsf{I}_n - \mathsf{A}^k$$

where A is any matrix. We apply to $\mathsf{P}_k = \mathsf{I}_N - (\mathsf{I}_N - \mathsf{S}\mathsf{S}^+)^k$ with $\mathsf{A} = \mathsf{S}\mathsf{S}^+$ to derive

$$\begin{aligned}
\mathsf{P}_k &= \big(\mathsf{I}_n + (\mathsf{I}_n - \mathsf{S}\mathsf{S}^+) + (\mathsf{I}_n - \mathsf{S}\mathsf{S}^+)^2 + \cdots + (\mathsf{I}_n - \mathsf{S}\mathsf{S}^+)^{k-1}\big)\mathsf{S}\mathsf{S}^+ && (1.37)\\
&= \mathsf{S}\mathsf{I}_n\mathsf{S}^+ + \mathsf{S}(\mathsf{I}_n - \mathsf{S}^+\mathsf{S})\mathsf{S}^+ + \mathsf{S}(\mathsf{I}_n - \mathsf{S}^+\mathsf{S})^2\mathsf{S}^+ + \cdots + \mathsf{S}(\mathsf{I}_n - \mathsf{S}^+\mathsf{S})^{k-1}\mathsf{S}^+ && (1.38)\\
&= \mathsf{S}\big(\mathsf{I}_n + (\mathsf{I}_n - \mathsf{S}^+\mathsf{S}) + (\mathsf{I}_n - \mathsf{S}^+\mathsf{S})^2 + \cdots + (\mathsf{I}_n - \mathsf{S}^+\mathsf{S})^{k-1}\big)\mathsf{S}^+ && (1.39)\\
&= \mathsf{S}\tilde{\mathsf{S}}_k^+ && (1.40)
\end{aligned}$$

Then

$$\mathsf{P}_k = \mathsf{S}\tilde{\mathsf{S}}_k^+ = \mathsf{S}\tilde{\mathsf{m}}_k^{-1}\mathsf{S}^T\mathsf{M} \tag{1.41}$$

1.4.2 Estimating the Inverse Error

The error in $\tilde{\mathsf{m}}_k^{-1}$ is then found (using geometric series) by evaluating

$$\begin{aligned}
\tilde{\mathsf{m}}_k^{-1}\tilde{\mathsf{m}} &= (\mathsf{I}_n + \mathsf{A} + \mathsf{A}^2 + \mathsf{A}^3 + \cdots + \mathsf{A}^{k-1})\big(\mathsf{I}_n - \mathsf{A}\big) && (1.42)\\
&= \mathsf{I}_n - \mathsf{A}^k = \mathsf{I}_n - (\mathsf{I}_n - \mathsf{S}^+\mathsf{S})^k && (1.43)\\
&= \sum_{\ell=1}^{k} (-1)^{\ell+1} \binom{k}{\ell} \big(\mathsf{S}^+\mathsf{S}\big)^\ell = \left(\sum_{\ell=1}^{k} (C_{k\ell} \big(\mathsf{S}^+\mathsf{S}\big)^{\ell-1} \right) \mathsf{S}^+\mathsf{S} && (1.44)\\
&= \tilde{\mathsf{m}}_k^{-1}\mathsf{m}\mathsf{S}^+\mathsf{S} && (1.45)
\end{aligned}$$

which just gets back to starting point by substituting for $\tilde{\mathsf{m}}$.

1.5 Another Possible Change for BC and Contact

We can try to define the mass matrix to include velocity changes caused by boundary conditions or contact. The least square analysis change to

$$\Omega = \sum_p M_p \big(\boldsymbol{V}_p - \tilde{\boldsymbol{V}}_p\big)^2 = \sum_p M_p \big(\boldsymbol{V}_p - \sum_i S_{pi}\boldsymbol{v}_i - \sum_i S_{pi}\Delta\boldsymbol{v}_i\big)^2 \tag{1.46}$$

$$0 = \frac{d\Omega}{d\boldsymbol{v}_j} = -2\left(\sum_p M_p \big(\boldsymbol{V}_p - \sum_i S_{pi}\boldsymbol{v}_i - \sum_i S_{pi}\Delta\boldsymbol{v}_i\big)\right)\left(S_{pj} + \sum_k S_{pk}\frac{d\Delta\boldsymbol{v}_k}{d\boldsymbol{v}_j}\right) \tag{1.47}$$

Here $\boldsymbol{v}_i$ are velocities in absence of contact and boundary conditions and $\Delta\boldsymbol{v}_i$ is change in velocity for node i caused by contact and boundary conditions ($\Delta\boldsymbol{v}_i = 0$ for nodes remote from contact and boundary conditions).

$$\sum_p M_p\left(S_{pj} + \sum_k S_{pk}\frac{d\Delta\boldsymbol{v}_k}{d\boldsymbol{v}_j}\right)\sum_i S_{pi}(\boldsymbol{v}_i + \Delta\boldsymbol{v}_i) = \sum_p\left(S_{pj} + \sum_k S_{pk}\frac{d\Delta\boldsymbol{v}_k}{d\boldsymbol{v}_j}\right)M_p\boldsymbol{V}_p \tag{1.48}$$

Begin with only boundary conditions where $\Delta\boldsymbol{v}_k = (v_k^{(BC)} - \boldsymbol{v}_k\cdot\hat{\boldsymbol{n}}^{(BC)})\hat{\boldsymbol{n}}^{(BC)}\delta_{kb}$ where b is list of nodes with boundary conditions and $v_k^{(BC)}$ is boundary condition on node k in that list in direction $\hat{\boldsymbol{n}}^{(BC)}$. For 1D analysis, we have $d\Delta\boldsymbol{v}_k/d\boldsymbol{v}_j = -\delta_{jk}\delta_{kb}$:

$$\big(1-\delta_{jb}\big)\sum_i\sum_p M_p S_{pj}S_{pi}(\boldsymbol{v}_i + \Delta\boldsymbol{v}_i) = \big(1-\delta_{jb}\big)\sum_p S_{jp}^T M_p\boldsymbol{V}_p \tag{1.49}$$

If we set $\boldsymbol{v} = \tilde{\mathsf{m}}^{-1}\boldsymbol{p}$, then above equation gives $\boldsymbol{v}_j$ for $j\notin b$. For $j\in b$, the equation is zero equals zero, but those velocities are controlled by the boundary condition. In other words:

$$\boldsymbol{v}_j = \begin{cases} (\tilde{\mathsf{m}}^{-1}\boldsymbol{p})_j & j\notin b \\ \boldsymbol{v}_j^{(BC)} & j\in b \end{cases} \quad \text{or} \quad \boldsymbol{v} = \tilde{\mathsf{m}}^{-1}\boldsymbol{p} + \Delta\boldsymbol{v} \tag{1.50}$$

where $\Delta\boldsymbol{v}$ is change in velocity due to boundary conditions or contact.

If we interpret grid velocity as extrapolated momentum plus any velocity changes needed for contact or grid velocity conditions, the grid velocity can be rewritten as velocity extrapolation with approximate full mass matrix should change. First, rewrite the mapping matrix as:

$$\boldsymbol{v} = \left(\sum_{\ell=1}^{k} C_{k\ell}\left(\mathsf{S}^+\mathsf{S}\right)^{\ell-1}\right)\mathsf{S}^+\boldsymbol{V} \tag{1.51}$$

$$= k\mathsf{S}^+\boldsymbol{V} + \left(\sum_{\ell=2}^{k} C_{k\ell}\left(\mathsf{S}^+\mathsf{S}\right)^{\ell-2}\right)\mathsf{S}^+\mathsf{S}\boldsymbol{v}^{(L0)} \tag{1.52}$$

where $\boldsymbol{v}^{(L0)} = \mathsf{S}^+\boldsymbol{V}$ is lumped mass matrix velocity ignoring contact and boundary conditions. Now imagine contact (which likely will need to use lumped mass matrix methods) or boundary conditions (which would be more efficient with lumped mass matrix methods) causing a change in lumped grid velocity of $\Delta\boldsymbol{v}^{(L)}$ defined by the lumped grid extrapolation being:

$$\boldsymbol{v}^{(L)} = \mathsf{S}^+\boldsymbol{V} + \Delta\boldsymbol{v}^{(L)} = \boldsymbol{v}^{(L0)} + \Delta\boldsymbol{v}^{(L)} \tag{1.53}$$

and assume this calculation should start with $\boldsymbol{v}^{(L)}$ in place of $\boldsymbol{v}^{(L0)}$:

$$\begin{aligned}
\boldsymbol{v} &= k\mathsf{S}^{+}\boldsymbol{V} + \left(\sum_{\ell=2}^{k} C_{k\ell}\left(\mathsf{S}^{+}\mathsf{S}\right)^{\ell-2}\right)\left(\mathsf{S}^{+}\mathsf{S}\mathsf{S}^{+}\boldsymbol{V} + \mathsf{S}^{+}\mathsf{S}\Delta\boldsymbol{v}^{(L)}\right) && (1.54)\\
&= \tilde{\mathsf{m}}_k^{-1}\boldsymbol{p}^{(0)} + \left(\sum_{\ell=2}^{k} C_{k\ell}\left(\mathsf{S}^{+}\mathsf{S}\right)^{\ell-1}\right)\mathsf{m}^{-1}\mathsf{m}\Delta\boldsymbol{v}^{(L)} && (1.55)\\
&= \tilde{\mathsf{m}}_k^{-1}\boldsymbol{p}^{(0)} + \left(\tilde{\mathsf{m}}_k^{-1} - k\mathsf{m}^{-1}\right)\Delta\boldsymbol{p}^{(L)} && (1.56)
\end{aligned}$$

A different approach seems to get a different answer. Rewrite the pseudo-inverse as:

$$\begin{aligned}
\tilde{\mathsf{S}}_k^{+} &= \mathsf{S}^{+} + \left(\mathsf{I}_n + (\mathsf{I}_n - \mathsf{S}^{+}\mathsf{S}) + \cdots + (\mathsf{I}_n - \mathsf{S}^{+}\mathsf{S})^{k-2}\right)\mathsf{S}^{+}(\mathsf{I}_N - \mathsf{S}\mathsf{S}^{+}) && (1.57)\\
&= \mathsf{S}^{+} + \tilde{\mathsf{S}}_{k-1}^{+}(\mathsf{I}_N - \mathsf{S}\mathsf{S}^{+}) && (1.58)
\end{aligned}$$

The grid velocity calculation becomes

$$\boldsymbol{v} = \mathsf{S}^{+}\boldsymbol{V} + \tilde{\mathsf{S}}_{k-1}^{+}(\boldsymbol{V} - \mathsf{S}\mathsf{S}^{+}\boldsymbol{V}) = \mathsf{S}^{+}\boldsymbol{V} + \tilde{\mathsf{S}}_{k-1}^{+}(\boldsymbol{V} - \mathsf{S}\boldsymbol{v}^{(L0)}) \tag{1.59}$$

Again, we start with $\boldsymbol{v}^{(L)}$ in place of $\boldsymbol{v}^{(L0)}$:

$$\begin{aligned}
\boldsymbol{v} &= \mathsf{S}^{+}\boldsymbol{V} + \tilde{\mathsf{S}}_{k-1}^{+}(\boldsymbol{V} - \mathsf{S}\mathsf{S}^{+}\boldsymbol{V} - \mathsf{S}\Delta\boldsymbol{v}^{(L)}) && (1.60)\\
&= \tilde{\mathsf{S}}_{k}^{+}\boldsymbol{V} - \tilde{\mathsf{S}}_{k-1}^{+}\mathsf{S}\Delta\boldsymbol{v}^{(L)} && (1.61)\\
&= \tilde{\mathsf{m}}_k^{-1}\mathsf{S}^{T}\mathsf{M}\boldsymbol{V} - \tilde{\mathsf{m}}_{k-1}^{-1}\mathsf{S}^{T}\mathsf{M}\mathsf{S}\Delta\boldsymbol{v}^{(L)} && (1.62)\\
&= \tilde{\mathsf{m}}_k^{-1}\boldsymbol{p}^{(0)} - \tilde{\mathsf{m}}_{k-1}^{-1}\mathsf{S}^{T}\mathsf{M}\mathsf{S}\mathsf{m}^{-1}\Delta\boldsymbol{p}^{(L)} && (1.63)\\
&= \tilde{\mathsf{m}}_k^{-1}\boldsymbol{p}^{(0)} - \tilde{\mathsf{m}}_{k-1}^{-1}\mathsf{S}^{+}\mathsf{S}\Delta\boldsymbol{p}^{(L)} && (1.64)
\end{aligned}$$

The new terms can possible be rewritten:

$$\begin{aligned}
\tilde{\mathsf{m}}_k^{-1} - \tilde{\mathsf{m}}_{k-1}^{-1} &= (\mathsf{I}_n - \mathsf{S}^{+}\mathsf{S})^{k-1}\mathsf{m}^{-1} && (1.65)\\
&= \mathsf{m}^{-1} - \left(\sum_{\ell=1}^{k-1}\binom{k-1}{\ell}(-1)^{\ell+1}(\mathsf{S}^{+}\mathsf{S})^{\ell-1}\mathsf{m}^{-1}\right)\mathsf{S}^{+}\mathsf{S} && (1.66)\\
&= \mathsf{m}^{-1} - \tilde{\mathsf{m}}_{k-1}^{-1}\mathsf{S}^{+}\mathsf{S} && (1.67)\\
-\tilde{\mathsf{m}}_{k-1}^{-1}\mathsf{S}^{+}\mathsf{S} &= \tilde{\mathsf{m}}_k^{-1} - \tilde{\mathsf{m}}_{k-1}^{-1} - \mathsf{m}^{-1} && (1.68)
\end{aligned}$$

Then

$$\boldsymbol{v} = \tilde{\mathsf{m}}_k^{-1}\left(\boldsymbol{p}^{(0)} + \Delta\boldsymbol{p}^{(L)}\right) - \left(\tilde{\mathsf{m}}_{k-1}^{-1} + \mathsf{m}^{-1}\right)\Delta\boldsymbol{p}^{(L)} \tag{1.69}$$

The contact section below seems to give a third answer. Which is correct?

1.6 Components of Full Mass Matrix Inverse

Give full mass matrix inverse of

$$\tilde{\mathsf{m}}_k^{-1} = \left(\sum_{\ell=1}^{k} C_{k\ell}\left(\mathsf{S}^{+}\mathsf{S}\right)^{\ell-1}\right)\mathsf{m}^{-1} \tag{1.70}$$

can we evaluate individual components (if needed in the algorithm). We define vector

$$\boldsymbol{m}^{(j)} = \left((\tilde{\mathsf{m}}_k)^{-1}_{1j}, (\tilde{\mathsf{m}}_k)^{-1}_{2j}, \ldots, (\tilde{\mathsf{m}}_k)^{-1}_{nj} \right) = \tilde{\mathsf{m}}^{-1}_k \boldsymbol{\nu} \tag{1.71}$$

where elements of vector $\boldsymbol{\nu}$ are $\nu_i = \delta_{ij}$ and we have the components of the inverse from $(\tilde{\mathsf{m}}_k)^{-1}_{ij} = \boldsymbol{m}^{(j)}_i$. We can write

$$\boldsymbol{m}^{(j)} = \left(\sum_{\ell=1}^{k} C_{k\ell} \left(\mathsf{S}^{+}\mathsf{S} \right)^{\ell-1} \right) \boldsymbol{\nu} \tag{1.72}$$

where vector $\boldsymbol{\nu}$ is redefined with components $\nu_i = \delta_{ij}/m_j$. This result is identical to XPIC calculations for velocity. We can thus extract for j of inverse matrix by seeding nodal velocities with $\nu^*_i = k\delta_{ij}/m_j$ and using XPIC methods to find final $\boldsymbol{m}^{(j)} = \boldsymbol{\nu}^*$.

Chapter 2

The MPM Algorithm - Single Material Mode

This chapter goes through each MPM task, writes it using vector and matrix notation, and then discusses changes needed to implement full mass matrix interpretation of XPIC(m) denoted as FMPM. The tasks are for single-velocity field mode. Dealing with cracks, multiple materials, and contact is covered in subsequent chapters.

2.1 MPM Time Step Tasks

The section outlines the various tasks in `OSParticulas` and where changes are needed to implement full mass matrix.

2.1.1 Extrapolate Mass and Momentum to the Grid

The first task is to extrapolate mass and momenta to the grid using

$$\boldsymbol{p}_i = \sum_p S_{ip}\boldsymbol{P}_p \quad \text{or} \quad \boldsymbol{p} = \mathsf{S}^T\boldsymbol{P} = \mathsf{S}^T\mathsf{M}\boldsymbol{V} \tag{2.1}$$

with particle mass matrix $\mathsf{M} = \text{diag}(\boldsymbol{M})$ and nodal lumped mass from $\boldsymbol{m} = \mathsf{S}^T\boldsymbol{M}$. The full mass matrix is

$$\tilde{\mathsf{m}} = \mathsf{S}^T\mathsf{M}\mathsf{S} \tag{2.2}$$

But it is neither calculated nor stored.

Post Initial Extrapolations

The tasks done in `OSParticulas` are:

- Find total nodal mass by summing masses from material and crack velocity field and identify material and crack contact nodes.
- Adjust momenta on multimaterial and crack contact nodes. This task is not done in single velocity field MPM.
- Change nodal nodal momenta to match velocity boundary conditions. The details are given in section 2.4.

Althought steps 2 and 3 are done every time step in `OSParticulas`, I think some can be skipped depending on update method (*e.g.*, velocity BCs not needed if using `USL±` update methods).

2.1.2 Update Strains First

If being done, this task updates the particle strains by extrapolating velocity gradient from the grid to the particles, but this should now be based on velocities using full mass matrix. The new task is to find grid velocity using

$$\boldsymbol{v} = \tilde{\mathsf{S}}_k^+ \boldsymbol{V} = \tilde{\mathsf{m}}_k^{-1} \mathsf{S}^T \mathsf{M} \boldsymbol{V} = \left(\sum_{\ell=1}^{k} C_{k\ell} \left(\mathsf{S}^+ \mathsf{S} \right)^{\ell-1} \right) \boldsymbol{v}^{(L)} \tag{2.3}$$

where $\boldsymbol{v}_i^{(L)} = \boldsymbol{p}_i'/m_i$ are velocities from for momenta corrected for contact (not done in single velocity mode) and boundary conditions (see section 2.4). This calculation is identical to XPIC(k) task of finding $\boldsymbol{v}(k)$. The XPIC(k) style implementation is

$$\boldsymbol{v} \quad = \quad \sum_{\ell=1}^{k} \boldsymbol{v}_\ell \qquad \text{where} \qquad \boldsymbol{v}_\ell = C_{k\ell} (\mathsf{S}^+ \mathsf{S})^{\ell-1} \boldsymbol{v}^{(L)} \tag{2.4}$$

As a recursion relation:

$$\boldsymbol{v}_\ell = C_{k,\ell-1} \frac{C_{k\ell}}{C_{k,\ell-1}} (\mathsf{S}^+ \mathsf{S})(\mathsf{S}^+ \mathsf{S})^{\ell-2} \boldsymbol{v}^{(L)} = \frac{C_{k\ell}}{C_{k,\ell-1}} (\mathsf{S}^+ \mathsf{S}) \boldsymbol{v}_{\ell-1} \quad \text{seeded with} \quad \boldsymbol{v}_1 = k \boldsymbol{v}^{(L)} \tag{2.5}$$

From `Extrap.pdf` notes after a sign addition to keep -1 term in definition:

$$\frac{C_{k\ell}}{C_{k,\ell-1}} = -\frac{k-\ell+1}{k} \tag{2.6}$$

Once velocities are found, extrapolate to particle using gradient shape functions:

$$\nabla \boldsymbol{v}_p = \sum_i \boldsymbol{v}_i \otimes \boldsymbol{G}_{pi} = \sum_i \boldsymbol{v}_i^T \boldsymbol{G}_{pi} \tag{2.7}$$

where vectors are row vectors and transposing switches between row and column vectors. Define gradient shape function matrix as $N \times n$ matrix G where each element is a column vector:

$$(\mathsf{G})_{pi} = (\boldsymbol{G}_{pi})^T = \left(\frac{\int \chi_p(\boldsymbol{x}) \nabla N_i(\boldsymbol{x}) d\boldsymbol{x}}{\int \chi_p(\boldsymbol{x}) d\boldsymbol{x}} \right)^T \tag{2.8}$$

This extrapolation can be cast in matrix form as

$$\nabla \boldsymbol{v}^T = \sum_i (\mathsf{G})_{pi} \boldsymbol{v}_i \quad \text{or} \quad \nabla \boldsymbol{v}^T = \mathsf{G} \boldsymbol{v} = \mathsf{G} \tilde{\mathsf{m}}_k^{-1} \boldsymbol{p}' \tag{2.9}$$

where $\nabla \boldsymbol{v}^T$ is vector of transposes of the particle velocity gradient tensors.

2.1.3 Extrapolate Forces

The force extrapolation to node i is:

$$\boldsymbol{f}_i = -\sum_p \frac{M_p \boldsymbol{\tau}_p (\mathsf{G})_{pi}}{\rho_{p,0}} + \boldsymbol{f}_{i,T} + \sum_p \left(M_p S_{ip} \boldsymbol{b}_p + \boldsymbol{F}_p S_{ip} \right) \tag{2.10}$$

The first term is internal force and the rest are external forces (tractions, body forces, and particle forces). Note that $\boldsymbol{\tau}_p$ is Kirchoff stress tensor. In matrix form (and because $\boldsymbol{\tau}_p$ is symmetric):

$$\boldsymbol{f} = -\mathsf{M}\mathsf{G}^T \tilde{\boldsymbol{\tau}} + \boldsymbol{f}_T + \mathsf{M}\mathsf{S}^T \boldsymbol{b} + \mathsf{S}^T \boldsymbol{F} \tag{2.11}$$

where $(\mathsf{G}^T)_{ip} = \boldsymbol{G}_{pi}$ are row vectors and $\tilde{\boldsymbol{\tau}}$ is vector with $\boldsymbol{\tau}_p / \rho_{p,0}$ in element p (and $\rho_{p,0}$ is undeformed particle density). This task is unchanged when using full mass matrix.

Post Force Extrapolation

The tasks done after extrapolating forces (to finish force calculations) in `OSParticuals` are:

- Traction force are done here it code (*i.e.*, the $\boldsymbol{f}_T$ terms in force extrapolations).
- Add tractions on crack surfaces from cohesive laws (can be imagined as part of traction forces)
- Add body forces and optional add damping proportional to $\boldsymbol{v}_i$ or $\boldsymbol{f}_i$. Body forces done here assume they depend only on grid position on not on state of the particle.
- Change forces to implement velocity boundary conditions (see section 2.4). Not that here we change force while prior to strain updates, we change momenta.

2.1.4 Update Grid Momenta

Update grid momenta using contact forces:

$$\boldsymbol{p}^+ = \boldsymbol{p} + \boldsymbol{f} \Delta t \tag{2.12}$$

Post Momentum Update

The tasks after the momentum update are:

- Crack and multimaterial contact calculations. These will need to use lumped mass matrix methods. This task is not done in single velocity field MPM.
- Reimpose velocity boundary conditions in particle momenta. In regular MPM, this step is done to handle cases where contact calculations conflicted with boundary conditions. In FMPM, this calculation may or may not be needed and will change velocities instead of momenta or forces (see section 2.4).

2.1.5 Update Particles

The particle update (not including damping for now) uses PIC or

$$\boldsymbol{V}^{(n+1)} = \mathsf{S}\tilde{\mathsf{m}}_k^{-1} \boldsymbol{p}^+ = \mathsf{S}(\tilde{\mathsf{m}}_k^{-1} \boldsymbol{p} + \tilde{\mathsf{m}}_k^{-1} \boldsymbol{f} \Delta t) = \mathsf{S}(\boldsymbol{v}(k) + \tilde{\mathsf{m}}_k^{-1} \boldsymbol{f} \Delta t) \tag{2.13}$$

We find acceleration $\boldsymbol{a} = \tilde{\mathsf{m}}_k^{-1} \boldsymbol{f}$ using an XPIC task. More details and the position update are given in section 2.3.

2.1.6 Update Strains Last

For USL- and USAVG-, this tasks can used the final velocities found in the particle update tasks. If using USL+ or USAVG+, FMPM will need to do contact, impose boundary conditions, and finally re-use XPIC(k) task on re-extrapoalted momenta prior to the strain update.

2.1.7 Move Cracks Tasks

Use the correct velocities, but not done in single-material mode MPM.

2.2 Bare Bones MPM Algorithm

The following is bare-bones definition of MPM algorithm including particle tractions and velocity boundary conditions. It omits cracks, contact/multimaterial mode, damping, and coupled transport tasks.

1. Extrapolate mass and momentum to the grid:
$$\boldsymbol{m}\mathsf{S}^T\boldsymbol{M} \qquad \text{and} \qquad \boldsymbol{p} = \mathsf{S}^T\mathsf{M}\boldsymbol{V} \tag{2.14}$$
2. For USF or USAVG± update options, update particle stress and strains as explained below.
3. Extrapolate forces to the grid including forces due to traction boundary conditions and body forces:
$$\boldsymbol{f} = -\mathsf{M}\mathsf{G}^T\tilde{\boldsymbol{\tau}} + \boldsymbol{f}_T + \mathsf{M}\mathsf{S}^T\boldsymbol{b} + \mathsf{S}^T\boldsymbol{F} \tag{2.15}$$
4. Update grid momenta
$$\boldsymbol{p}^+ = \boldsymbol{p} + \boldsymbol{f}\Delta t \tag{2.16}$$
5. Find grid velocity using $\boldsymbol{v}^+ = \tilde{\mathsf{m}}_k^{-1}\boldsymbol{p}^+$ and then update particle velocity and position using
$$\boldsymbol{V}^{(n+1)} = \mathsf{S}\boldsymbol{v}^+ \qquad \text{and} \qquad \boldsymbol{X}^{(n+1)} = \boldsymbol{X}^{(n)} + \frac{1}{2}\left(\boldsymbol{V}^{(n+1)} + \boldsymbol{V}^{(n)}\right)\Delta t \tag{2.17}$$
6. For USL± or USAVG± update options:
   (a) For USL- or USAVG-, this update can use $\boldsymbol{v}^+$ just calculated in the particle update tasks and then update particle stress and strains as explained below but skip the first step.
   (b) For USL+ or USAVG+, re-extrapolate $\boldsymbol{p} = \mathsf{S}^T\mathsf{M}\boldsymbol{V}$ and then update particle stress and strains as explained below.

Once or twice per time step, the current grid velocity are used as input to update stresses and strains on the particle. The process for this update is:

1. Change current momenta to $\boldsymbol{p}' = \boldsymbol{p} + \boldsymbol{f}^{(BC)}\Delta t$ and find grid velocity from $\boldsymbol{v} = \tilde{\mathsf{m}}_k^{-1}\boldsymbol{p}'$
2. Extrapolate velocity gradient to the particles
$$\nabla\boldsymbol{v}^T = \mathsf{G}\boldsymbol{v} \tag{2.18}$$
3. Update stress and strains using particle's constitutive law. Note that $\nabla\boldsymbol{v}^T$ is spatial gradient. Also not that for USAVG±, the update uses $\Delta t/2$ for the time step.

2.3 Particle Update Details

We can express particle velocity and position update in a generic form as

$$\boldsymbol{V}^{(n+1)} = \boldsymbol{V}^{(n)} + \mathbb{A}^{(n)}\Delta t \tag{2.19}$$

$$\boldsymbol{X}^{(n+1)} = \boldsymbol{X}^{(n)} + \mathbb{V}^{(n)}\Delta t + \frac{1}{2}\mathbb{A}^{(n)}(\Delta t)^2 \tag{2.20}$$

All MPM methods can be expressed in this form. For example, give any equations for particle velocity and position updates, we can derive that method's implied particle acceleration and initial velocity as:

$$\mathbb{A}^{(n)}\Delta t = \boldsymbol{V}^{(n+1)} - \boldsymbol{V}^{(n)} \tag{2.21}$$

$$\mathbb{V}^{(n)}\Delta t = \boldsymbol{X}^{(n+1)} - \boldsymbol{X}^{(n)} - \frac{1}{2}\left(\boldsymbol{V}^{(n+1)} - \boldsymbol{V}^{(n)}\right)\Delta t \tag{2.22}$$

Note that the net Lagrangian velocity imposed by the position update is

$$\left(\frac{\Delta \boldsymbol{X}}{\Delta t}\right) = \mathbb{V}^{(n)} + \frac{1}{2}\mathbb{A}^{(n)}\Delta t \tag{2.23}$$

while the average Lagrangian velocity imposed by the velocity update is

$$\langle \boldsymbol{V} \rangle_{\boldsymbol{X}} = \boldsymbol{V}^{(n)} + \frac{1}{2}\mathbb{A}^{(n)}\Delta t \tag{2.24}$$

In other words, for methods with $\mathbb{V}^{(n)} \neq \boldsymbol{V}^{(n)}$, the method will propagate an error between expected Lagrangian updates for particle position and velocity of:

$$\text{error} \propto \left(\frac{\Delta \boldsymbol{X}}{\Delta t}\right) - \langle \boldsymbol{V} \rangle_{\boldsymbol{X}} = \mathbb{V}^{(n)} - \boldsymbol{V}^{(n)} \tag{2.25}$$

2.3.1 FMPM Updates

The update proposed for FMPM(k), reduces to generic form using

$$\mathbb{A}^{(n)}\Delta t = \mathsf{S}\tilde{\mathsf{m}}_k^{-1}\boldsymbol{p}^+ - \boldsymbol{V}^{(n)} \qquad \text{and} \qquad \mathbb{V}^{(n)} = \boldsymbol{V}^{(n)} \tag{2.26}$$

The error in Lagrangian updates is zero.

2.3.2 Prior FLIP MPM Updates

Now contrast this to standard FLIP MPM updates of:

$$\boldsymbol{V}^{(n+1)} = \boldsymbol{V}^{(n)} + \mathsf{S}\mathsf{m}^{-1}\boldsymbol{f}\Delta t \tag{2.27}$$

$$\boldsymbol{X}^{(n+1)} = \boldsymbol{X}^{(n)} + \left(\mathsf{S}\mathsf{m}^{-1}\boldsymbol{p}^+ - \frac{\psi}{2}\mathsf{S}\mathsf{m}^{-1}\boldsymbol{f}\Delta t\right)\Delta t \tag{2.28}$$

In other words, find acceleration and updated velocity on the grid using lumped mass matrix followed by velocity update using acceleration extrapolated to the particle and position update using updated velocity and a second-order term involving acceleration. The original MPM papers (and most papers in the literature) ignore the second order term in position update because it is small. This common update is recovered by setting $\psi = 0$. Using $\psi = 1$ illustrates changes by including a second order term.

The standard MPM FLIP form reduces to generic form using:

$$\mathbb{A}^{(n)}\Delta t = \mathsf{Sm}^{-1}\boldsymbol{f}\Delta t \quad \text{and} \quad \mathbb{V}^{(n)} = \mathsf{Sm}^{-1}\left(\frac{(1-\psi)\boldsymbol{p}^{+}+(1+\psi)\boldsymbol{p}}{2}\right) \tag{2.29}$$

The error in Lagrangian updates become

$$\text{error} \propto \left(\frac{\Delta \boldsymbol{X}}{\Delta t}\right) - \langle \boldsymbol{V}\rangle_X = \mathsf{Sm}^{-1}\left(\frac{(1-\psi)\boldsymbol{p}^{+}+(1+\psi)\boldsymbol{p}}{2}\right) - \boldsymbol{V}^{(n)} \tag{2.30}$$

If $\psi = 0$ (which is most common in the literature)

$$\mathbb{A}^{(n)}\Delta t = \mathsf{Sm}^{-1}\boldsymbol{f}\Delta t \qquad \text{and} \qquad \mathbb{V}^{(n)} = \mathsf{Sm}^{-1}\left(\frac{\boldsymbol{p}^{+}+\boldsymbol{p}}{2}\right) \tag{2.31}$$

Extending to second order in position update with $\psi = 1$ gives

$$\mathbb{A}^{(n)}\Delta t = \mathsf{Sm}^{-1}\boldsymbol{f}\Delta t \qquad \text{and} \qquad \mathbb{V}^{(n)} = \mathsf{Sm}^{-1}\boldsymbol{p}, \tag{2.32}$$

Both values of ψ result in Lagrangian update error, but this was not an error in original MPM development. One can easily eliminate that error by choosing $\mathbb{V}^{(n)} = \boldsymbol{V}^{(n)}$ (as used in FMPM update). This choice would change the standard FLIP MPM second-order update for position to be

$$\boldsymbol{X}^{(n+1)} = \boldsymbol{X}^{(n)} + \left(\boldsymbol{V}^{(n)} + \frac{1}{2}\mathsf{Sm}^{-1}\boldsymbol{f}\Delta t\right)\Delta t \tag{2.33}$$

Although this eliminates Lagrangian update errors (because $\mathbb{V}^{(n)} = \boldsymbol{V}^{(n)}$), it gives very poor results. The solution to stabilize FLIP update was to replace velocity used in the position update with the smooth velocity defined above (see Eq. (2.29)). This suggestions was likely made in early days of FLIP updates (by Brackbilll) and perhaps has been forgotten that it was a trick to improve updates rather than ideal choice based on Lagrangian particles. An advantage of FMPM is one no longer needs an *ad hoc* change in velocity used in particle update.

2.3.3 Prior XPIC Updates

Prior to recognizing that XPIC(m) calculations done on grid momenta is the same as using $\tilde{\mathsf{m}}^{-1}_{k=m}$ on those momenta, the prior XPIC($m = k$) update now using mass matrix notation was proposed as:

$$\boldsymbol{V}^{(n+1)} = \mathsf{S}\tilde{\mathsf{m}}_k^{-1}\boldsymbol{p} + \mathsf{Sm}^{-1}\boldsymbol{f}\Delta t \tag{2.34}$$

$$\boldsymbol{X}^{(n+1)} = \boldsymbol{X}^{(n)} + \left[\mathsf{Sm}^{-1}\boldsymbol{p} + \frac{\boldsymbol{V}^{(n+1)} - \boldsymbol{V}^{(n)}}{2}\right]\Delta t \tag{2.35}$$

These updates reduce to generic form using:

$$\mathbb{A}^{(n)}\Delta t = \mathsf{S}\tilde{\mathsf{m}}_k^{-1}\boldsymbol{p}^{+} + \mathsf{S}\left(\mathsf{m}^{-1} - \tilde{\mathsf{m}}_k^{-1}\right)\boldsymbol{f}\Delta t - \boldsymbol{V}^{(n)} \quad \text{and} \quad \mathbb{V}^{(n)} = \mathsf{Sm}^{-1}\boldsymbol{p} \tag{2.36}$$

with Lagrangian update error

$$\text{error} \propto \left(\frac{\Delta \boldsymbol{X}}{\Delta t}\right) - \langle \boldsymbol{V}\rangle_X = \mathsf{Sm}^{-1}\boldsymbol{p} - \boldsymbol{V}^{(n)} \tag{2.37}$$

Note that XPIC(m) is identical to standard FLIP (with $\psi = 1$) except uses a different effective acceleration. That acceleration reduced noise to help stabilize FLIP simulations. The new FMPM(k) updates improves further by using $\tilde{\mathsf{m}}_k$ in place of m in $\mathbb{A}^{(n)}$ and setting $\mathbb{V}^{(n)} = \boldsymbol{V}^{(n)}$.

2.3.4 Various PIC Updates

XPIC(1) update reduces to a PIC method:

$$\boldsymbol{V}^{(n+1)} = \mathsf{Sm}^{-1}\boldsymbol{p}^{+} \quad \text{and} \quad \boldsymbol{X}^{(n+1)} = \boldsymbol{X}^{(n)} + \left[\mathsf{Sm}^{-1}\boldsymbol{p} + \frac{\mathsf{Sm}^{-1}\boldsymbol{p}^{+} - \boldsymbol{V}^{(n)}}{2}\right]\Delta t \tag{2.38}$$

These reduce to generic form

$$\mathbb{A}^{(n)}\Delta t = \mathsf{Sm}^{-1}\boldsymbol{p}^{+} - \boldsymbol{V}^{(n)} \qquad \text{and} \qquad \mathbb{V}^{(n)} = \mathsf{Sm}^{-1}\boldsymbol{p} \tag{2.39}$$

In contrast, FMPM(1) reduces to new PIC method:

$$\boldsymbol{V}^{(n+1)} = \mathsf{Sm}^{-1}\boldsymbol{p}^{+} \quad \text{and} \quad \boldsymbol{X}^{(n+1)} = \boldsymbol{X}^{(n)} + \left[\frac{\mathsf{Sm}^{-1}\boldsymbol{p}^{+} + \boldsymbol{V}^{(n)}}{2}\right]\Delta t \tag{2.40}$$

These reduce to generic form

$$\mathbb{A}^{(n)}\Delta t = \mathsf{Sm}^{-1}\boldsymbol{p}^{+} - \boldsymbol{V}^{(n)} \qquad \text{and} \qquad \mathbb{V}^{(n)} = \boldsymbol{V}^{(n)} \tag{2.41}$$

Calculations show that PIC derived from FMPM(1) is more stable than the one derived from XPIC(1) (especially for problems where PIC works well such as MMS with linear velocities).

Some papers (in animation, such as Stomakhin, *et al.*) suggested blending PIC and FLIP, but they used the wrong-headed, position update in their FLIP implementation of

$$\boldsymbol{X}^{(n+1)} = \boldsymbol{X}^{(n)} + \boldsymbol{V}^{(n+1)}\Delta t$$

instead of standard first order FLIP update ($\psi = 0$) of

$$\boldsymbol{X}^{(n+1)} = \boldsymbol{X}^{(n)} + \mathsf{Sm}^{-1}\boldsymbol{p}^{+}\Delta t$$

Problems with this wrong-headed, position update were pointed out by Brackbill. The problems with that update probably explains why they needed to blend PIC and FLIP, but they blended only the velocity update and not the position update (or mistakenly thought FLIP and PIC position updates are the same). For a fraction α of FLIP, their complete updates are

$$\begin{aligned}
\boldsymbol{V}^{(n+1)} &= \alpha\left(\boldsymbol{V}^{(n)} + \mathsf{Sm}^{-1}\boldsymbol{f}\Delta t\right) + (1-\alpha)\mathsf{Sm}^{-1}\boldsymbol{p}^{+} = \alpha\boldsymbol{V}^{(n)} + \mathsf{Sm}^{-1}\left(\boldsymbol{p}^{+} - \alpha\boldsymbol{p}\right)\\
\boldsymbol{X}^{(n+1)} &= \boldsymbol{X}^{(n)} + \boldsymbol{V}^{(n+1)}\Delta t
\end{aligned}$$

These correspond to general form

$$\begin{aligned}
\mathbb{A}^{(n)}\Delta t &= \mathsf{Sm}^{-1}\left(\boldsymbol{p}^{+} - \alpha\boldsymbol{p}\right) - (1-\alpha)\boldsymbol{V}^{(n)}\\
\mathbb{V}^{(n)} &= \frac{\boldsymbol{V}^{(n)} + \boldsymbol{V}^{(n+1)}}{2} = \frac{(1+\alpha)\boldsymbol{V}^{(n)} + \mathsf{Sm}^{-1}\left(\boldsymbol{p}^{+} - \alpha\boldsymbol{p}\right)}{2}
\end{aligned}$$

Note that $\alpha = 1$ gives a poor version of FLIP with acceleration correct, but effective velocity changed to

$$\mathbb{V}^{(n)} = \boldsymbol{V}^{(n)} + \frac{1}{2}\mathsf{Sm}^{-1}\boldsymbol{f}\Delta t$$

Picking $\alpha = 0$ gives another variant for PIC with same acceleration as others, but effective velocity changed to

$$\mathbb{V}^{(n)} = \frac{\boldsymbol{V}^{(n)} + \mathsf{Sm}^{-1}\boldsymbol{p}^{+}}{2}$$

Table 2.1: Inputs are velocities and acceleration extrapolated to the particles ($v_p = \mathsf{S}v^+$, $v_m = \mathsf{S}v$, $v_s = \mathsf{S}v^*$, $a = \mathsf{S}\mathsf{m}^{-1}f$) and the current time step ($dt = \Delta t$ or $ht = \Delta t/2$). Outputs are updated particle velocity, V_p, and position, X_p.The "M", "A", and"C" in last line are the number of multiplications, additions, and memory copies in each update method.

FLIP(Std)	PIC(Std)	FMPM(0)/XPIC(0)	XPIC(k>0)	FMPM(k>0)
$V_p \mathrel{+}= a\,dt$ $X_p \mathrel{+}= v_p dt$	$V_p = v_p$ $X_p \mathrel{+}= v_p dt$	$dV = a\,dt$ $V_p \mathrel{+}= dV$ $X_p \mathrel{+}= (v_p - 0.5dV)dt$	$dV = -V_p$ $V_p = k(v_m - v_s) + a\,dt$ $dV \mathrel{+}= V_p$ $X_p \mathrel{+}= (v_m + 0.5dV)dt$	$V_p = v_p$ $X_p \mathrel{+}= (V_p + v_p)ht$
a, v_p	v_p	a, v_p	a, v_m, v_s	v_p
2M, 2A	1M, 1A, 1C	3M, 3A, 1C	4M, 5A, 2C	1M, 2A, 1C

2.3.5 Particle Update Code Comparison

Table 2.1 gives codes steps required for each update method. The list line gives computation cost in terms of multiplication, additions, and memory copies. XPIC(k) needs most calculations because it needs the non-updated velocity on the grid in v_m. FMPM(k) and XPIC(k) also have computational cost in finding v^+. The table is only the cost after this value is found. Some methods only need some inputs and thus can save cost by not calculating those not needed.

Note that FMPM(k) is more efficient than XPIC(k) because of its more consistent treatment of velocities. Also note that neither PIC(Std) nor FMPM(k) even need acceleration as an input. Notice that neither XPIC(1) nor FMPM(1) reduce to PIC(Std). Instead they give improved forms of PIC with FMPM(1) the best. Finally, FMPM(0)/XPIC(0) is only different than FLIP(Std) because it is using a second order position update.

2.4 Velocity Boundary Conditions

One way to view velocity boundary conditions is that they add a force to each node, $\boldsymbol{f}^{(BC)}$, such that the updated momentum changes to:

$$\boldsymbol{p}^{+\prime} = \boldsymbol{p} + (\boldsymbol{f} + \boldsymbol{f}^{(BC)})\Delta t \tag{2.42}$$

and the final velocities given by:

$$\boldsymbol{v}^{+\prime} = \tilde{\mathsf{m}}_k^{-1}\boldsymbol{p}^{+\prime} \tag{2.43}$$

satisfy boundary conditions on controlled nodes. First, imagine a single node i with velocity boundary condition b setting the velocity on node i to $v_i^{(b)}$ in direction $\hat{\boldsymbol{n}}^{(b)}$. To satisfy this boundary condition with approximate full mass matrix inverse, we need

$$\boldsymbol{v}_i^{+\prime}\cdot\hat{\boldsymbol{n}}^{(b)} = v_i^{(b)} = \sum_j \tilde{\mathsf{m}}_{k,ij}^{-1}\left(\boldsymbol{p}_j + (\boldsymbol{f}_j + \boldsymbol{f}_j^{(b)})\Delta t\right)\cdot\hat{\boldsymbol{n}}^{(b)} \tag{2.44}$$

$$\sum_j \tilde{\mathsf{m}}_{k,ij}^{-1}\boldsymbol{f}_j^{(b)}\cdot\hat{\boldsymbol{n}}^{(b)}\Delta t = v_i^{(b)} - \sum_j \tilde{\mathsf{m}}_{k,ij}^{-1}\boldsymbol{p}_j^{+}\cdot\hat{\boldsymbol{n}}^{(b)} \tag{2.45}$$

$$\left(\tilde{\mathsf{m}}_k^{-1}\boldsymbol{f}^{(b)}\right)_i\cdot\hat{\boldsymbol{n}}^{(b)}\Delta t = v_i^{(b)} - \left(\tilde{\mathsf{m}}_k^{-1}\boldsymbol{p}^{+}\right)_i\cdot\hat{\boldsymbol{n}}^{(b)} \tag{2.46}$$

$$\boldsymbol{a}_i^{(b)}\cdot\hat{\boldsymbol{n}}^{(b)}\Delta t = v_i^{(b)} - \boldsymbol{v}_i^{+}\cdot\hat{\boldsymbol{n}}^{(b)} \tag{2.47}$$

where

$$\boldsymbol{v}^+ = \tilde{\mathsf{m}}_k^{-1}\boldsymbol{p}^+ = \tilde{\mathsf{m}}_k^{-1}(\boldsymbol{p} + \boldsymbol{f}\Delta t) \tag{2.48}$$

are grid velocities found by ignoring boundary conditions and

$$\boldsymbol{a}^{(b)} = \tilde{\mathsf{m}}_k^{-1}\boldsymbol{f}^{(b)} \tag{2.49}$$

are accelerations due to all boundary condition forces resulting for this boundary condition. Assuming accelerations and force are in direction $\hat{\boldsymbol{n}}^{(b)}$, the BC acceleration becomes

$$\boldsymbol{a}_i^{(b)}\Delta t = \left(\tilde{\mathsf{m}}_k^{-1}\boldsymbol{f}^{(b)}\right)_i \Delta t = \left(v_i^{(b)} - \left(\tilde{\mathsf{m}}_k^{-1}\boldsymbol{p}^+\right)_i \cdot \hat{\boldsymbol{n}}^{(b)}\right)\hat{\boldsymbol{n}}^{(b)} = \left(v_i^{(b)} - \boldsymbol{v}_i^+ \cdot \hat{\boldsymbol{n}}^{(b)}\right)\hat{\boldsymbol{n}}^{(b)} \tag{2.50}$$

The above analysis allows the boundary condition forces to appear on more than one node, but their distribution cannot be found from this single equation derived from one boundary condition. This result contrasts to lumped mass matrix methods. For a lumped mass matrix, the above equation simplifies to:

$$\boldsymbol{f}_i^{(b)} \cdot \hat{\boldsymbol{n}}^{(b)}\Delta t = m_i v_i^{(b)} - \boldsymbol{p}_i^+ \cdot \hat{\boldsymbol{n}}^{(b)} \quad \text{and} \quad \boldsymbol{a}_i^{(b)} = \frac{\boldsymbol{f}_i^{(b)}}{m_i} \tag{2.51}$$

All force and acceleration is on the boundary condition node and all determined by the boundary condition equation.

In FMPM, each boundary condition still provides only one equation, but that is not enough information to determine force distribution among the nodes. Several options for FMPM were considered and all need some assumption about how to distribute force due to a boundary condition on a node. The various options with some comments are:

Lumped Method The simplest approach is to use prior lumped mass matrix methods without modification. The boundary condition force equation simplifies to

$$\boldsymbol{f}^{(L)}\Delta t = \sum_b \Delta\boldsymbol{p}^{(b)} \quad \text{where} \quad \Delta\boldsymbol{p}_j^{(b)} = \left(m_i v_i^{(b)} - \boldsymbol{p}_i^+ \cdot \hat{\boldsymbol{n}}^{(b)}\right)\hat{\boldsymbol{n}}^{(b)}\delta_{ij} \tag{2.52}$$

This equation assumes force due to boundary condition b is in the $\hat{\boldsymbol{n}}^{(b)}$ direction. This general form accommodates any number of boundary conditions and any nodes. A single node can have multiple boundary conditions in the same direction or in orthogonal directions on that node.

Remark: The forces are added only to nodes with boundary conditions, but when full mass matrix inverse is used to find grid velocities, those forces will also perturb nodes near the boundary conditions. It might be beneficial to boundary condition performance to distribute its effects of more than just the one boundary condition node.

Remark: The full mass matrix inverse will not project lumped force into boundary condition velocities. As a consequence, force on one boundary condition may perturb velocities on other nodes with boundary conditions. This approach, will only approximate boundary condition velocities in nodes with boundary conditions.

Velocity Method This also-simple approach applies boundary conditions after each calculation of grid velocity using full mass matrix methods. For example, in the particle update, the final updated velocities become sum of updated velocities ignoring boundary conditions with velocities changes imposed by boundary conditions:

$$\boldsymbol{v}^{+\prime} = \boldsymbol{v}^+ + \sum_b \Delta\boldsymbol{v}^{(b)} \quad \text{where} \quad \Delta\boldsymbol{v}_j^{(b)} = \left(v_i^{(b)} - \boldsymbol{v}_i^+ \cdot \hat{\boldsymbol{n}}^{(b)}\right)\hat{\boldsymbol{n}}^{(b)}\delta_{ij} \tag{2.53}$$

Remark: This approach corresponds to distributing forces over many nodes such that the resulting accelerations ($\boldsymbol{a}^{(b)} = \tilde{\mathsf{m}}_k^{-1}\boldsymbol{f}^{(b)}$) are localized only on nodes with boundary conditions. The grid velocities will exactly match boundary condition values, but nodes without boundary conditions (but near them) will feel no influence of the boundary.

Remark: By doing this calculation after finding grid velocities that ignore boundary condition, this approach avoids the need to ever calculate $\boldsymbol{f}^{(b)}$. One drawback is that reaction force at boundary conditions are not available. If desired for output, the reaction forces could be found using full mass matrix — $\boldsymbol{f}^{(b)} = \tilde{\mathsf{m}}\boldsymbol{a}^{(b)}$ (although maybe this needs to replace $\tilde{\mathsf{m}}$ with $\tilde{\mathsf{m}}_k$?).

Unlumped Method #1 One approach to unlumping the boundary condition methods is to spread out the lumped method using the full mass matrix. This method would find forces from

$$\boldsymbol{f}^{(\#1)}\Delta t = \sum_j \frac{\tilde{\mathsf{m}}_{ij}}{m_j}\boldsymbol{f}_j^{(L)} = \tilde{\mathsf{m}}\mathsf{m}^{-1}\boldsymbol{f}^{(L)} \tag{2.54}$$

Remark: An algorithm for this method is:

1. Calculate change in grid velocities using lumped methods on nodes with boundary conditions. In other words, find

$$\Delta\boldsymbol{v}^{(BC)} = \sum_b \Delta\boldsymbol{v}^{(b)} \quad \text{where} \quad \Delta\boldsymbol{v}_i^{(b)} = \begin{cases} \left(v_i^{(b)} - \frac{\boldsymbol{p}_i^+}{m_i}\cdot\hat{\boldsymbol{n}}^{(b)}\right)\hat{\boldsymbol{n}}^{(b)} & i = j_b \\ 0 & i \neq j_b \end{cases}$$

2. Use full mass matrix to find:

$$\boldsymbol{p}^{(BC)} = \boldsymbol{f}^{(BC)}\Delta t = \tilde{\mathsf{m}}\Delta\boldsymbol{v}^{(BC)} \tag{2.55}$$

   More explicitly:

$$\begin{aligned}\boldsymbol{p}_i^{(BC)} &= \sum_b\sum_j (\tilde{\mathsf{m}})_{ij}\boldsymbol{v}_j^{(b)} = \sum_b (\tilde{\mathsf{m}})_{ij_b}\left(v_{j_b}^{(b)} - \frac{\boldsymbol{p}_i^+}{m_i}\cdot\hat{\boldsymbol{n}}^{(b)}\right)\hat{\boldsymbol{n}}^{(b)} \\ &= \sum_b\sum_p M_p S_{pi} S_{pj_b}\left(v_{j_b}^{(b)} - \frac{\boldsymbol{p}_i^+}{m_i}\cdot\hat{\boldsymbol{n}}^{(b)}\right)\hat{\boldsymbol{n}}^{(b)}\end{aligned}$$

   This results would take another extrapolation over particles, but particles with $S_{pj_b} = 0$ can be skipped (*i.e.*, those not in the domain of any boundary condition node.

3. Once boundary condition forces are found, add them to grid forces (or grid momentum in certain strain update methods). The reason to rely on a lumped $\Delta\boldsymbol{v}^{(b)}$ is to allow this calculation to be made before the XPIC calculations to convert momenta to velocities.

Remark: This approach does not reduced to lumped methods when using a lumped mass matrix unless $\tilde{\mathsf{m}}$ is replaced by m in above equations. Perhaps it should be replace, in general, by $\tilde{\mathsf{m}}_k$. The next option does that is two places.

Unlumped Method #2 A second approach to unlumping the boundary conditions is use full mass matrix in the $\Delta\boldsymbol{v}$ calculation as follows:

1. Calculate change in grid velocities using full mass-matrix methods on nodes with boundary conditions. In other words, find

$$\Delta \boldsymbol{v}^{(BC)} = \sum_b \Delta \boldsymbol{v}^{(b)} \quad \text{where} \quad \Delta \boldsymbol{v}_i^{(b)} = \begin{cases} \left(v_i^{(b)} - (\tilde{\mathrm{m}}_k^{-1}\boldsymbol{p}^+)_i \cdot \hat{\boldsymbol{n}}^{(b)}\right)\hat{\boldsymbol{n}}^{(b)} & i = j_b \\ 0 & i \neq j_b \end{cases} \tag{2.56}$$

   where BC b is on j_b.

2. Use the approximate full mass matrix to find:

$$\boldsymbol{p}^{(BC)} = \boldsymbol{f}^{(BC)}\Delta t = \tilde{\mathrm{m}}_k \Delta \boldsymbol{v}^{(BC)}$$

   More explicitly:

$$\boldsymbol{p}_i^{(BC)} = \sum_b \sum_j (\tilde{\mathrm{m}}_k)_{ij} \boldsymbol{v}_j^{(b)} = \sum_b (\tilde{\mathrm{m}}_k)_{ij_b} \left(v_{j_b}^{(b)} - (\tilde{\mathrm{m}}_k^{-1}\boldsymbol{p}^+)_{j_b} \cdot \hat{\boldsymbol{n}}^{(b)}\right)\hat{\boldsymbol{n}}^{(b)}$$

3. If this boundary condition force could be evaluated (efficiently enough), the resulting nodal velocity would be

$$\begin{aligned} \boldsymbol{v}^{+'} &= \tilde{\mathrm{m}}_k^{-1}\left(\boldsymbol{p}^+ + \boldsymbol{p}^{(BC)}\right) \\ \boldsymbol{v}_i^{+'} &= (\tilde{\mathrm{m}}_k^{-1}\boldsymbol{p}^+)_i + \sum_b \sum_j (\tilde{\mathrm{m}}_k^{-1})_{ij} (\tilde{\mathrm{m}}_k)_{jj_b} \left(v_{j_b}^{(b)} - (\tilde{\mathrm{m}}_k^{-1}\boldsymbol{p}^+)_{j_b} \cdot \hat{\boldsymbol{n}}^{(b)}\right)\hat{\boldsymbol{n}}^{(b)} \\ &= (\tilde{\mathrm{m}}_k^{-1}\boldsymbol{p}^+)_i + \sum_b \delta_{ij_b} \left(v_{j_b}^{(b)} - (\tilde{\mathrm{m}}_k^{-1}\boldsymbol{p}^+)_{j_b} \cdot \hat{\boldsymbol{n}}^{(b)}\right)\hat{\boldsymbol{n}}^{(b)} \\ &= \boldsymbol{v}_i^+ + \sum_b \Delta \boldsymbol{v}_i^{(b)} \end{aligned}$$

Remark: leaving aside the question of whether or not it would be possible to find $\tilde{\mathrm{m}}_k$ used in the first two steps above, the final calculation seems to revert to the velocity method (including its limitations). An option to change might be to replace $\tilde{\mathrm{m}}_k$ with $\tilde{\mathrm{m}}$. The new result would get forces on other nodes, but would not exactly satisfy the boundary conditions. Or they could be combined to add the forces to other nodes, but enforce correct velocity on BC nodes?

Collection of Boundary Conditions If we limit force to nodes with boundary conditions and analyze all boundary conditions, we could solve the simultaneous equations for forces distributed over those nodes that lead to correct velocities on those nodes. In other words, collect all BCs in the same $\hat{\boldsymbol{n}}^{(BC)}$ direction. Then, for n_{BC} conditions in that direction, Eq. (2.47) would give n_{BC} equations with n_{BC} unknowns for the components of force. Two implementations options in decreasing order of computational cost are:

1. *Solve the Full Equation*: Find $\boldsymbol{v}^+ = \tilde{\mathrm{m}}_k^{-1}\boldsymbol{p}^+$ using full mass matrix. For each node with a BC, evaluate right size of Eq. (2.47). For each node i with a BC, find mass matrix components using $\boldsymbol{m}^{(j)} = \tilde{\mathrm{m}}_k^{-1}\boldsymbol{v}^{(j)}$ where elements of $\boldsymbol{v}^{(j)} = \delta_{ij}$. Assemble these elements into matrix and solve linear system for find BC forces. This method requires $n_{BC} + 1$ full mass matrix calculations and inversion of an $n_{BC} \times n_{BC}$ matrix.
2. *Partial Full Mass Matrix Method*: Find $\boldsymbol{v}^+ = \tilde{\mathrm{m}}_k^{-1}\boldsymbol{p}^+$ using full mass matrix. For each node with a BC, evaluate right size of Eq. (2.47). Finally solve for forces by setting elements of $(\tilde{\mathrm{m}}_k^{-1})_{ij} \to \delta_{ij}/m_i$ (*i.e.*, use lumped mass matrix to find force). This methods requires only one full mass matrix calculation.

Chapter 3

FMPM With Contact

3.1 Full Mass Matrix Equations

This section attempts to pose MPM contact using full mass matrices and vector equations. In multi-material mode, the extrapolations extrapolate each material to its own velocity field. We assume two materials α and β. The initial extrapolations will find $\boldsymbol{p}^\alpha$, $\boldsymbol{p}^\beta$, $\tilde{\mathsf{m}}^\alpha$, $\tilde{\mathsf{m}}^\beta$, m^α, and m^β all related to total momentum, mass matrix and lumped mass matrix by:

$$\boldsymbol{p} = \boldsymbol{p}^\alpha + \boldsymbol{p}^\beta, \qquad \tilde{\mathsf{m}} = \tilde{\mathsf{m}}^\alpha + \tilde{\mathsf{m}}^\beta, \qquad \text{and} \qquad \mathsf{m} = \mathsf{m}^\alpha + \mathsf{m}^\beta \tag{3.1}$$

Formally, materials α and β define separate spaces with different number of active nodes. Here the separate material matrices are expanded to the size of the all active nodes. The inverse of such matrix is the inverse found on with its local size and then expanded to full size by filling inactive rows and columns with zeros.

Now consider a set of velocities on the grid, found either from current or updated momenta on the grid by $\boldsymbol{v}^\alpha = (\tilde{\mathsf{m}}^\alpha)^{-1}\boldsymbol{p}^\alpha$ (and similar for β). The goal in contact mechanics is to change these velocities to $\boldsymbol{v}^\alpha + \Delta\boldsymbol{v}^\alpha$ and $\boldsymbol{v}^\beta + \Delta\boldsymbol{v}^\beta$ where the velocity changes satisfy

$$\Delta\boldsymbol{v}^\beta - \Delta\boldsymbol{v}^\alpha = \mathsf{k}\hat{\boldsymbol{t}} = \begin{cases} k_i\hat{\boldsymbol{t}}_i - \left(\boldsymbol{v}_i^\beta - \boldsymbol{v}_i^\alpha\right) & \text{nodes in contact} \\ 0 & \text{nodes not in contact} \end{cases} \tag{3.2}$$

To conserve momentum, the velocity changes are related to equal and opposite momenta added to each multimaterial node (note that $\Delta\boldsymbol{p}$ must be zero on any node with only material α or only material β):

$$\Delta\boldsymbol{v}^\alpha = (\tilde{\mathsf{m}}^\alpha)^{-1}\Delta\boldsymbol{p} \qquad \text{and} \qquad \Delta\boldsymbol{v}^\beta = -(\tilde{\mathsf{m}}^\beta)^{-1}\Delta\boldsymbol{p} \tag{3.3}$$

This condition restricts motion on contact nodes to the tangential direction on node i ($\hat{\boldsymbol{t}}_i$) where k_i depends on contact law (*e.g.*, $k_i = 0$ for frictionless sliding). By analogy to lumped mass matrix methods, we define $\tilde{\mathsf{m}}^{(red)} = \left((\tilde{\mathsf{m}}^\alpha)^{-1} + (\tilde{\mathsf{m}}^\beta)^{-1}\right)^{-1}$, which simplifies to:

$$\tilde{\mathsf{m}}^{(red)} = \left((\tilde{\mathsf{m}}^\beta)^{-1}\tilde{\mathsf{m}}^\beta(\tilde{\mathsf{m}}^\alpha)^{-1} + (\tilde{\mathsf{m}}^\alpha)^{-1}\tilde{\mathsf{m}}^\alpha(\tilde{\mathsf{m}}^\beta)^{-1}\right)^{-1} = \tilde{\mathsf{m}}^{-1}\tilde{\mathsf{m}}^\alpha\tilde{\mathsf{m}}^\beta \tag{3.4}$$

This simplifcation used property that symmetric matrices commute. Solving for $\Delta\boldsymbol{p}$:

$$\Delta\boldsymbol{p} = -\tilde{\mathsf{m}}^{(red)}\mathsf{k}\hat{\boldsymbol{t}} \tag{3.5}$$

Finding the velocity difference directly from momenta (and exploiting symmetric mass matrics) gives:

$$\boldsymbol{v}^\beta - \boldsymbol{v}^\alpha = (\tilde{\mathsf{m}}^\beta)^{-1}\boldsymbol{p}^\beta - (\tilde{\mathsf{m}}^\alpha)^{-1}\boldsymbol{p}^\alpha = (\tilde{\mathsf{m}}^\alpha)^{-1}(\tilde{\mathsf{m}}^\beta)^{-1}\left(\tilde{\mathsf{m}}^\alpha\boldsymbol{p}^\beta - \tilde{\mathsf{m}}^\beta\boldsymbol{p}^\alpha\right) \tag{3.6}$$

The momentum change becomes:

$$\Delta \boldsymbol{p} = \tilde{\mathsf{m}}^{(red)} \begin{cases} \left[(\tilde{\mathsf{m}}^{\alpha})^{-1}(\tilde{\mathsf{m}}^{\beta})^{-1}\left(\tilde{\mathsf{m}}^{\alpha}\boldsymbol{p}^{\beta} - \tilde{\mathsf{m}}^{\beta}\boldsymbol{p}^{\alpha}\right)\right]_i - k_i \hat{\boldsymbol{t}}_i & \text{nodes in contact} \\ 0 & \text{nodes not in contact} \end{cases} \tag{3.7}$$

Each component of $\Delta \boldsymbol{p}$ is

$$(\Delta \boldsymbol{p})_i = \sum_j \left(\tilde{\mathsf{m}}_{ij}^{(red)} \left[(\tilde{\mathsf{m}}^{\alpha})^{-1}(\tilde{\mathsf{m}}^{\beta})^{-1}\left(\tilde{\mathsf{m}}^{\alpha}\boldsymbol{p}^{\beta} - \tilde{\mathsf{m}}^{\beta}\boldsymbol{p}^{\alpha}\right)\right]_j - \tilde{\mathsf{m}}_{ij}^{(red)} k_j \hat{\boldsymbol{t}}_j \right) \delta_{jc} \tag{3.8}$$

where $\delta_{jc} = 1$ for node j in contact and 0 otherwise. When using a lumped mass matrix, $\tilde{\mathsf{m}}_{ij}^{(red)} = m_i^{\alpha} m_i^{\beta} \delta_{ij}/m_i = m_i^{(red)} \delta_{ij}$. The momentum change on contact nodes become separate equations:

$$\Delta \boldsymbol{p}_i = \left(\frac{m_i^{\alpha}\boldsymbol{p}_i^{\beta} - m_i^{\beta}\boldsymbol{p}_i^{\alpha}}{m_i} - m_i^{(red)} \mathsf{k}_i \hat{\boldsymbol{t}}_i \right) \delta_{ic} \tag{3.9}$$

Which agrees with MPM contact papers with general contact laws.

The next step in contact analysis is to find the apparent normal force on the interface by finding the momenta in the normal direction on each contact node to prevent interpenetration (or for materials to move with center of mass velocity in the normal direction). The starting point is to solve for sticking conditions or:

$$\left(\boldsymbol{v}^{\beta} + \Delta \boldsymbol{v}^{\beta} - \boldsymbol{v}^{\alpha} - \Delta \boldsymbol{v}^{\alpha}\right)_i \delta_{ic} = 0 \tag{3.10}$$

This analysis is identical to above analysis except $k_i = 0$ on all contact nodes. Each component of $\Delta \boldsymbol{p}(0)$ for sticking momenta is

$$\Delta \boldsymbol{p}(0)_i = \sum_j \left(\tilde{\mathsf{m}}_{ij}^{(red)} \left[(\tilde{\mathsf{m}}^{\alpha})^{-1}(\tilde{\mathsf{m}}^{\beta})^{-1}\left(\tilde{\mathsf{m}}^{\alpha}\boldsymbol{p}^{\beta} - \tilde{\mathsf{m}}^{\beta}\boldsymbol{p}^{\alpha}\right)\right]_j \right) \delta_{jc} \tag{3.11}$$

For lumped mass matrix method, this results reduces to:

$$\Delta \boldsymbol{p}(0)_i = \frac{m_i^{\alpha}\boldsymbol{p}_i^{\beta} - m_i^{\beta}\boldsymbol{p}_i^{\alpha}}{m_i} \delta_{ic} \tag{3.12}$$

It is not clear how to proceed in full mass matrix mode. All momenta changes are coupled to all other contact nodes (if nearby at least). Sticking conditions on one node will depend on whether or not other nodes stick. The only workable solution seems top be to use lumped mass matrix and then decide how to handle the calculated momenta changes in the full mass matrix methods that follow.

3.2 XPIC(k) with Contact

Prior work on contact adds a new term to $\mathsf{P}_k \boldsymbol{V}$ in the particle update resulting in:

$$\begin{aligned} \mathsf{P}_k \boldsymbol{V}^{(n)} &= \left(\mathsf{I}_N - (\mathsf{I}_N - \mathsf{S}\mathsf{S}^{+\alpha})^k\right)\boldsymbol{V} + (\mathsf{I}_N - \mathsf{S}\mathsf{S}^{+\alpha})^{k-1}\mathsf{S}\Delta \boldsymbol{v}^{\alpha} & (3.13) \\ &= \left(\mathsf{I}_N - (\mathsf{I}_N - \mathsf{S}\mathsf{S}^{+\alpha})^k\right)\boldsymbol{V} + \mathsf{S}(\mathsf{I} - \mathsf{S}^{+\alpha}\mathsf{S})^{k-1}\Delta \boldsymbol{v}^{\alpha} & (3.14) \end{aligned}$$

Noting now that

$$(\tilde{\mathsf{m}}_k^{\alpha})^{-1} - (\tilde{\mathsf{m}}_{k-1}^{\alpha})^{-1} = (\mathsf{I}_n - \mathsf{S}^{+\alpha}\mathsf{S})^{k-1}(\mathsf{m}^{\alpha})^{-1} \tag{3.15}$$

where $(\tilde{\mathsf{m}}_k^\alpha)^{-1}$ is full mass matrix inverse based on $\mathsf{S}^{+\alpha}$ and m^α, this contact term with $\Delta\boldsymbol{v}^\alpha$ becomes:

$$\mathsf{P}_k^{(c)}\boldsymbol{V}^{(n)} = \mathsf{S}\left((\tilde{\mathsf{m}}_k^\alpha)^{-1} - (\tilde{\mathsf{m}}_{k-1}^\alpha)^{-1}\right)\mathsf{m}^\alpha\Delta\boldsymbol{v}^\alpha = \mathsf{S}\left((\tilde{\mathsf{m}}_k^\alpha)^{-1} - (\tilde{\mathsf{m}}_{k-1}^\alpha)^{-1}\right)\Delta\boldsymbol{p}^\alpha \tag{3.16}$$

where $\Delta\boldsymbol{p}^\alpha = \mathsf{m}\Delta\boldsymbol{v}^\alpha$ is found with lumped mass matrix. The full projection becomes

$$\begin{aligned}\mathsf{P}_k\boldsymbol{V}^{(n)} &= \mathsf{S}\left((\tilde{\mathsf{m}}_k^\alpha)^{-1}\boldsymbol{p}^{\alpha 0} + (\tilde{\mathsf{m}}_k^\alpha)^{-1}\Delta\boldsymbol{p}^\alpha - (\tilde{\mathsf{m}}_{k-1}^\alpha)^{-1}\Delta\boldsymbol{p}^\alpha\right) &(3.17)\\ &= \mathsf{S}\left((\tilde{\mathsf{m}}_k^\alpha)^{-1}\boldsymbol{p}^\alpha - (\tilde{\mathsf{m}}_{k-1}^\alpha)^{-1}\Delta\boldsymbol{p}^\alpha\right) &(3.18)\end{aligned}$$

where $\boldsymbol{p}^\alpha = \boldsymbol{p}^{\alpha 0} + \Delta\boldsymbol{p}^\alpha$ are extrapolated momenta after correction of initially extrapolated momenta, $\boldsymbol{p}^{\alpha 0}$, by contact calculations in $\Delta\boldsymbol{p}^\alpha$ (which use lumped mass matrix methods).

3.3 Multimaterial Particle Updates

The derived update for multimaterial XPIC(k) in revised contact paper, now using full mass matrix notation, is

$$\begin{aligned}\boldsymbol{V}^{(n+1)} &= \mathsf{P}_k\boldsymbol{V}^{(n)} + \mathsf{S}\,(\mathsf{m}^\alpha)^{-1}\left(\boldsymbol{f}^{\alpha 0} + \boldsymbol{f}^{\alpha C}\right)\Delta t &(3.19)\\ &= \mathsf{S}\left((\tilde{\mathsf{m}}_k^\alpha)^{-1}\boldsymbol{p}^\alpha - (\tilde{\mathsf{m}}_{k-1}^\alpha)^{-1}\Delta\boldsymbol{p}^\alpha + (\mathsf{m}^\alpha)^{-1}\,\Delta\boldsymbol{p}^{\alpha+} + (\mathsf{m}^\alpha)^{-1}\,\boldsymbol{f}^{\alpha 0}\Delta t\right) &(3.20)\\ \boldsymbol{X}^{(n+1)} &= \boldsymbol{X}^{(n)} + \left[\mathsf{S}\mathsf{m}^{-1}\boldsymbol{p}^\alpha + \frac{\boldsymbol{V}^{(n+1)} - \boldsymbol{V}^{(n)}}{2}\right]\Delta t &(3.21)\end{aligned}$$

where $\Delta\boldsymbol{p}^{\alpha+}$ enters in contact force calculation of $\boldsymbol{f}^{\alpha C}\Delta t = \Delta\boldsymbol{p}^{\alpha+}$ by the usual lumped-mass contact calculation. This update appears to work well although it does not revert exactly to single material mode when the interface is perfect.

The only difference in effective acceleration between FMPM(k) and XPIC(k) in single material mode are that $(\mathsf{m})^{-1}$ is replaced by $(\tilde{\mathsf{m}}_k)^{-1}$. Perhaps FMPM($k$) for contact should take XPIC(k) for contact and make the same replacement (using α version) in effective acceleration. The FMPM(k) update with contact in multimaterial mode becomes:

$$\begin{aligned}\boldsymbol{V}^{(n+1)} &= \mathsf{S}\left((\tilde{\mathsf{m}}_k^\alpha)^{-1}\boldsymbol{p}^{\alpha+} - (\tilde{\mathsf{m}}_{k-1}^\alpha)^{-1}\Delta\boldsymbol{p}^\alpha\right) &(3.22)\\ \boldsymbol{X}^{(n+1)} &= \boldsymbol{X}^{(n)} + \left[\frac{\boldsymbol{V}^{(n+1)} + \boldsymbol{V}^{(n)}}{2}\right]\Delta t &(3.23)\end{aligned}$$

where $\boldsymbol{p}^{\alpha+} = \boldsymbol{p}^\alpha + \Delta\boldsymbol{p}^{\alpha+} + \boldsymbol{f}^{\alpha C}\Delta t$.

Questions:

1. Does this work in USL-. Test: redo code to use XPIC contact method and test in shock
2. Another difference in XPIC is that it uses lumped mass matrix for extrapolating velocity gradient to the particle. Should FMPM do that too. Test: hack USL- to recalculate grid velocity by lumped method prior to update. See how that looks in prior FMPM without BCs or contact. Try in contact too.
3. After deciding (again) between lumped and full in velocity gradient extrapolation, try other methods again. First without BCs or contact. Second with BCs. Last next item does with contact.
4. If FMPM should use full mass velocity, how should it correct for contact in other full mass locations? Test: Is lumped method best (if yes likely done). If not, try various methods.