A Polynomial Particle-In-Cell Method:
Supplementary Technical Document

1 List of Basis

1.1 Linear interpolation

Polynomials of the form

$$s(\mathbf{z}) = \prod_{\beta=1}^{d} z_\beta^{i_\beta}, \quad i_\beta = 0, 1$$

are all we need for linear interpolation. We have $\mathcal{S}_{pr}^n \cdot \left(\mathbf{m}_p^n \mathcal{S}_{pt}^n\right) = 0$ forAll $r \neq t$.

1.2 Quadratic interpolation

For quadratic interpolation, by replacing $z_\beta^{i_\beta}$ with $g_\beta(w) = w^2 - \frac{x_{p\beta}^n\left(\Delta x^2 - 4(x_{p\beta}^n)^2\right)}{\Delta x^2} w - \frac{\Delta x^2}{4}$ in

$$s(\mathbf{z}) = \prod_{\beta=1}^{d} z_\beta^{i_\beta}, \quad i_\beta = 0, 1, 2$$

whenever $i_\beta = 2$, we get the full set of basis. For completeness we list all the basis below.

In 2D,

$$\begin{aligned}
&s_1(\mathbf{x}_{\mathbf{i}_{kp}^n}^n - \mathbf{x}_p^n) = 1 \\
&s_2(\mathbf{x}_{\mathbf{i}_{kp}^n}^n - \mathbf{x}_p^n) = x_{\mathbf{i}_{kp1}} - x_{p1}^n && s_3(\mathbf{x}_{\mathbf{i}_{kp}^n}^n - \mathbf{x}_p^n) = x_{\mathbf{i}_{kp2}} - x_{p2}^n \\
&s_4(\mathbf{x}_{\mathbf{i}_{kp}^n}^n - \mathbf{x}_p^n) = (x_{\mathbf{i}_{kp1}} - x_{p1}^n)(x_{\mathbf{i}_{kp2}} - x_{p2}^n) \\
&s_5(\mathbf{x}_{\mathbf{i}_{kp}^n}^n - \mathbf{x}_p^n) = g_1(x_{\mathbf{i}_{kp1}} - x_{p1}^n) && s_6(\mathbf{x}_{\mathbf{i}_{kp}^n}^n - \mathbf{x}_p^n) = g_2(x_{\mathbf{i}_{kp2}} - x_{p2}^n) \\
&s_7(\mathbf{x}_{\mathbf{i}_{kp}^n}^n - \mathbf{x}_p^n) = g_1(x_{\mathbf{i}_{kp1}} - x_{p1}^n)(x_{\mathbf{i}_{kp2}} - x_{p2}^n) && s_8(\mathbf{x}_{\mathbf{i}_{kp}^n}^n - \mathbf{x}_p^n) = g_2(x_{\mathbf{i}_{kp2}} - x_{p2}^n)(x_{\mathbf{i}_{kp1}} - x_{p1}^n) \\
&s_9(\mathbf{x}_{\mathbf{i}_{kp}^n}^n - \mathbf{x}_p^n) = g_1(x_{\mathbf{i}_{kp1}} - x_{p1}^n) g_2(x_{\mathbf{i}_{kp2}} - x_{p2}^n)
\end{aligned}$$

In 3D,

$$\begin{aligned}
&s_1(\mathbf{x}_{\mathbf{i}_{kp}^n}^n - \mathbf{x}_p^n) = 1 \\
&s_2(\mathbf{x}_{\mathbf{i}_{kp}^n}^n - \mathbf{x}_p^n) = x_{\mathbf{i}_{kp1}} - x_{p1}^n \\
&s_3(\mathbf{x}_{\mathbf{i}_{kp}^n}^n - \mathbf{x}_p^n) = x_{\mathbf{i}_{kp2}} - x_{p2}^n \\
&s_4(\mathbf{x}_{\mathbf{i}_{kp}^n}^n - \mathbf{x}_p^n) = x_{\mathbf{i}_{kp3}} - x_{p3}^n \\
&s_5(\mathbf{x}_{\mathbf{i}_{kp}^n}^n - \mathbf{x}_p^n) = (x_{\mathbf{i}_{kp1}} - x_{p1}^n)(x_{\mathbf{i}_{kp2}} - x_{p2}^n) \\
&s_6(\mathbf{x}_{\mathbf{i}_{kp}^n}^n - \mathbf{x}_p^n) = (x_{\mathbf{i}_{kp1}} - x_{p1}^n)(x_{\mathbf{i}_{kp3}} - x_{p3}^n) \\
&s_7(\mathbf{x}_{\mathbf{i}_{kp}^n}^n - \mathbf{x}_p^n) = (x_{\mathbf{i}_{kp2}} - x_{p2}^n)(x_{\mathbf{i}_{kp3}} - x_{p3}^n) \\
&s_8(\mathbf{x}_{\mathbf{i}_{kp}^n}^n - \mathbf{x}_p^n) = (x_{\mathbf{i}_{kp1}} - x_{p1}^n)(x_{\mathbf{i}_{kp2}} - x_{p2}^n)(x_{\mathbf{i}_{kp3}} - x_{p3}^n) \\
&s_9(\mathbf{x}_{\mathbf{i}_{kp}^n}^n - \mathbf{x}_p^n) = g_1(x_{\mathbf{i}_{kp1}} - x_{p1}^n) \\
&s_{10}(\mathbf{x}_{\mathbf{i}_{kp}^n}^n - \mathbf{x}_p^n) = g_2(x_{\mathbf{i}_{kp2}} - x_{p2}^n) \\
&s_{11}(\mathbf{x}_{\mathbf{i}_{kp}^n}^n - \mathbf{x}_p^n) = g_3(x_{\mathbf{i}_{kp3}} - x_{p3}^n) \\
&s_{12}(\mathbf{x}_{\mathbf{i}_{kp}^n}^n - \mathbf{x}_p^n) = g_1(x_{\mathbf{i}_{kp1}} - x_{p1}^n) g_2(x_{\mathbf{i}_{kp2}} - x_{p2}^n) \\
&s_{13}(\mathbf{x}_{\mathbf{i}_{kp}^n}^n - \mathbf{x}_p^n) = g_2(x_{\mathbf{i}_{kp2}} - x_{p2}^n) g_3(x_{\mathbf{i}_{kp3}} - x_{p3}^n) \\
&s_{14}(\mathbf{x}_{\mathbf{i}_{kp}^n}^n - \mathbf{x}_p^n) = g_1(x_{\mathbf{i}_{kp1}} - x_{p1}^n) g_3(x_{\mathbf{i}_{kp3}} - x_{p3}^n) \\
&s_{15}(\mathbf{x}_{\mathbf{i}_{kp}^n}^n - \mathbf{x}_p^n) = g_1(x_{\mathbf{i}_{kp1}} - x_{p1}^n) g_2(x_{\mathbf{i}_{kp2}} - x_{p2}^n) g_3(x_{\mathbf{i}_{kp3}} - x_{p3}^n)
\end{aligned}$$

$$
\begin{aligned}
s_{16}(\mathbf{x}_{\mathbf{i}_{kp}^n}^n - \mathbf{x}_p^n) &= g_1(x_{\mathbf{i}_{kp1}} - x_{p1}^n)(x_{\mathbf{i}_{kp2}} - x_{p2}^n) \\
s_{17}(\mathbf{x}_{\mathbf{i}_{kp}^n}^n - \mathbf{x}_p^n) &= g_1(x_{\mathbf{i}_{kp1}} - x_{p1}^n)(x_{\mathbf{i}_{kp3}} - x_{p3}^n) \\
s_{18}(\mathbf{x}_{\mathbf{i}_{kp}^n}^n - \mathbf{x}_p^n) &= g_1(x_{\mathbf{i}_{kp1}} - x_{p1}^n)(x_{\mathbf{i}_{kp2}} - x_{p2}^n)(x_{\mathbf{i}_{kp3}} - x_{p3}^n)
\end{aligned}
$$

$$
\begin{aligned}
s_{19}(\mathbf{x}_{\mathbf{i}_{kp}^n}^n - \mathbf{x}_p^n) &= g_2(x_{\mathbf{i}_{kp2}} - x_{p2}^n)(x_{\mathbf{i}_{kp1}} - x_{p1}^n) \\
s_{20}(\mathbf{x}_{\mathbf{i}_{kp}^n}^n - \mathbf{x}_p^n) &= g_2(x_{\mathbf{i}_{kp2}} - x_{p2}^n)(x_{\mathbf{i}_{kp3}} - x_{p3}^n) \\
s_{21}(\mathbf{x}_{\mathbf{i}_{kp}^n}^n - \mathbf{x}_p^n) &= g_2(x_{\mathbf{i}_{kp2}} - x_{p2}^n)(x_{\mathbf{i}_{kp1}} - x_{p1}^n)(x_{\mathbf{i}_{kp3}} - x_{p3}^n)
\end{aligned}
$$

$$
\begin{aligned}
s_{22}(\mathbf{x}_{\mathbf{i}_{kp}^n}^n - \mathbf{x}_p^n) &= g_3(x_{\mathbf{i}_{kp3}} - x_{p3}^n)(x_{\mathbf{i}_{kp1}} - x_{p1}^n) \\
s_{23}(\mathbf{x}_{\mathbf{i}_{kp}^n}^n - \mathbf{x}_p^n) &= g_3(x_{\mathbf{i}_{kp3}} - x_{p3}^n)(x_{\mathbf{i}_{kp2}} - x_{p2}^n) \\
s_{24}(\mathbf{x}_{\mathbf{i}_{kp}^n}^n - \mathbf{x}_p^n) &= g_3(x_{\mathbf{i}_{kp3}} - x_{p3}^n)(x_{\mathbf{i}_{kp1}} - x_{p1}^n)(x_{\mathbf{i}_{kp2}} - x_{p2}^n)
\end{aligned}
$$

$$
\begin{aligned}
s_{25}(\mathbf{x}_{\mathbf{i}_{kp}^n}^n - \mathbf{x}_p^n) &= g_1(x_{\mathbf{i}_{kp1}} - x_{p1}^n)g_2(x_{\mathbf{i}_{kp2}} - x_{p2}^n)(x_{\mathbf{i}_{kp3}} - x_{p3}^n) \\
s_{26}(\mathbf{x}_{\mathbf{i}_{kp}^n}^n - \mathbf{x}_p^n) &= g_1(x_{\mathbf{i}_{kp1}} - x_{p1}^n)g_3(x_{\mathbf{i}_{kp3}} - x_{p3}^n)(x_{\mathbf{i}_{kp2}} - x_{p2}^n) \\
s_{27}(\mathbf{x}_{\mathbf{i}_{kp}^n}^n - \mathbf{x}_p^n) &= g_2(x_{\mathbf{i}_{kp2}} - x_{p2}^n)g_3(x_{\mathbf{i}_{kp3}} - x_{p3}^n)(x_{\mathbf{i}_{kp1}} - x_{p1}^n)
\end{aligned}
$$

The entries for $\mathcal{S}_{pr}^n \cdot \left(\mathbf{m}_p^n \mathcal{S}_{pr}^n\right)$, $r = 1, 2, \cdots, 27$ are:

$$
\begin{aligned}
&1, \\
&\tfrac{\Delta x^2}{4}, \tfrac{\Delta x^2}{4}, \tfrac{\Delta x^2}{4}, \\
&\tfrac{\Delta x^4}{16}, \tfrac{\Delta x^4}{16}, \tfrac{\Delta x^4}{16}, \tfrac{\Delta x^6}{64}, \\
&\tfrac{\left(\Delta x^2-4x^2\right)^2\left(3\Delta x^2-4x^2\right)}{16\Delta x^2}, \tfrac{\left(\Delta x^2-4y^2\right)^2\left(3\Delta x^2-4y^2\right)}{16\Delta x^2}, \tfrac{\left(\Delta x^2-4z^2\right)^2\left(3\Delta x^2-4z^2\right)}{16\Delta x^2}, \\
&\tfrac{\left(\Delta x^2-4x^2\right)^2\left(3\Delta x^2-4x^2\right)\left(\Delta x^2-4y^2\right)^2\left(3\Delta x^2-4y^2\right)}{256\Delta x^4}, \\
&\tfrac{\left(\Delta x^2-4x^2\right)^2\left(3\Delta x^2-4x^2\right)\left(\Delta x^2-4z^2\right)^2\left(3\Delta x^2-4z^2\right)}{256\Delta x^4}, \\
&\tfrac{\left(\Delta x^2-4y^2\right)^2\left(3\Delta x^2-4y^2\right)\left(\Delta x^2-4z^2\right)^2\left(3\Delta x^2-4z^2\right)}{256\Delta x^4}, \\
&\tfrac{\left(\Delta x^2-4x^2\right)^2\left(3\Delta x^2-4x^2\right)\left(\Delta x^2-4y^2\right)^2\left(3\Delta x^2-4y^2\right)\left(\Delta x^2-4z^2\right)^2\left(3\Delta x^2-4z^2\right)}{4096\Delta x^6}, \\
&\tfrac{1}{64}\left(\Delta x^2-4x^2\right)^2\left(3\Delta x^2-4x^2\right), \tfrac{1}{64}\left(\Delta x^2-4x^2\right)^2\left(3\Delta x^2-4x^2\right), \tfrac{1}{256}\left(3\Delta x^2-4x^2\right)\left(\Delta x^3-4\Delta x x^2\right)^2, \\
&\tfrac{1}{64}\left(\Delta x^2-4y^2\right)^2\left(3\Delta x^2-4y^2\right), \tfrac{1}{64}\left(\Delta x^2-4y^2\right)^2\left(3\Delta x^2-4y^2\right), \tfrac{1}{256}\left(3\Delta x^2-4y^2\right)\left(\Delta x^3-4\Delta x y^2\right)^2, \\
&\tfrac{1}{64}\left(\Delta x^2-4z^2\right)^2\left(3\Delta x^2-4z^2\right), \tfrac{1}{64}\left(\Delta x^2-4z^2\right)^2\left(3\Delta x^2-4z^2\right), \tfrac{1}{256}\left(3\Delta x^2-4z^2\right)\left(\Delta x^3-4\Delta x z^2\right)^2, \\
&\tfrac{\left(\Delta x^2-4x^2\right)^2\left(3\Delta x^2-4x^2\right)\left(\Delta x^2-4y^2\right)^2\left(3\Delta x^2-4y^2\right)}{1024\Delta x^2}, \\
&\tfrac{\left(\Delta x^2-4x^2\right)^2\left(3\Delta x^2-4x^2\right)\left(\Delta x^2-4z^2\right)^2\left(3\Delta x^2-4z^2\right)}{1024\Delta x^2}, \\
&\tfrac{\left(\Delta x^2-4y^2\right)^2\left(3\Delta x^2-4y^2\right)\left(\Delta x^2-4z^2\right)^2\left(3\Delta x^2-4z^2\right)}{1024\Delta x^2}
\end{aligned}
$$

2 Grid to Particle

From grid to particle, we wish to find $\mathbf{c}$ such that

$$
\begin{aligned}
\sum_{t=1}^{N_r} \mathcal{S}_{pr}^n \cdot \left(\mathbf{m}_p^n \mathcal{S}_{pt}^n\right) c_{pt\alpha}^{n+1} &= \mathcal{Q}_{pr\alpha}^n \cdot \left(\mathbf{M}_p^n \hat{\mathcal{V}}_p^{n+1}\right) \\
&= \sum_{k=1}^{(N_B+1)^d} m_{\mathbf{i}_{kp}^n p}^n s_r(\mathbf{x}_{\mathbf{i}_{kp}^n}^n - \mathbf{x}_p^n) \hat{v}_{\mathbf{i}_{kp}^n \alpha}^{n+1}.
\end{aligned}
$$

The basis we choose satisfy the property that $\mathcal{S}_{pr}^n \cdot \left(\mathbf{m}_p^n \mathcal{S}_{pt}^n\right) = 0$ for $r \neq t$. So we have

$$\begin{aligned}\mathcal{S}_{pr}^n \cdot \left(\mathbf{m}_p^n \mathcal{S}_{pr}^n\right) c_{pr\alpha}^{n+1} &= \mathcal{Q}_{pr\alpha}^n \cdot \left(\mathbf{M}_p^n \hat{\mathcal{V}}_p^{n+1}\right) \\ &= \sum_{k=1}^{(N_B+1)^d} m_{\mathbf{i}_{kp}^n p}^n s_r(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n) \hat{v}_{\mathbf{i}_{kp}^n \alpha}^{n+1}.\end{aligned}$$

For linear interpolation, the grid to particle transfer is similar to APIC. For quadratic interpolation, $\mathcal{S}_{pr}^n \cdot \left(\mathbf{m}_p^n \mathcal{S}_{pr}^n\right)$ can be zero for some r when $x_{p\alpha}^n = \pm\frac{h}{2}, \pm\frac{\sqrt{3}}{2}h$. However, we can still find a meaningful expression for $\mathbf{c}$.

In 2D, $m_{\mathbf{i}_{kp}^n p}^n = m_p N(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n) = m_p N_1(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n) N_2(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n)$, $\mathbf{c}$ can be computed from the formula below:

$$\begin{aligned}
c_{p1\alpha} &= \sum_{k=1}^{(N_B+1)^d} N(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n) \hat{v}_{\mathbf{i}_{kp}^n \alpha}^{n+1} \\
c_{p2\alpha} &= \frac{\sum_{k=1}^{(N_B+1)^d} N(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n)(x_{\mathbf{i}_{kp1}^n} - x_{p1}) \hat{v}_{\mathbf{i}_{kp}^n \alpha}^{n+1}}{\frac{\Delta x^2}{4}} \\
c_{p3\alpha} &= \frac{\sum_{k=1}^{(N_B+1)^d} N(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n)(x_{\mathbf{i}_{kp2}^n} - x_{p2}) \hat{v}_{\mathbf{i}_{kp}^n \alpha}^{n+1}}{\frac{\Delta x^2}{4}} \\
c_{p4\alpha} &= \frac{\sum_{k=1}^{(N_B+1)^d} N(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n)(x_{\mathbf{i}_{kp1}^n} - x_{p1})(x_{\mathbf{i}_{kp2}^n} - x_{p2}) \hat{v}_{\mathbf{i}_{kp}^n \alpha}^{n+1}}{\frac{\Delta x^4}{16}} \\
c_{p5\alpha} &= \frac{\sum_{k=1}^{(N_B+1)^d} N_2(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n)(-2)^{(i_{kp1}-1) \bmod 2} \hat{v}_{\mathbf{i}_{kp}^n \alpha}^{n+1}}{2\Delta x^2} \\
c_{p6\alpha} &= \frac{\sum_{k=1}^{(N_B+1)^d} N_1(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n)(-2)^{(i_{kp2}-1) \bmod 2} \hat{v}_{\mathbf{i}_{kp}^n \alpha}^{n+1}}{2\Delta x^2} \\
c_{p7\alpha} &= \frac{2\sum_{k=1}^{(N_B+1)^d} (x_{\mathbf{i}_{kp1}} - x_{p1}) N_2(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n)(-2)^{(i_{kp2}-1) \bmod 2} \hat{v}_{\mathbf{i}_{kp}^n \alpha}^{n+1}}{\Delta x^4} \\
c_{p8\alpha} &= \frac{2\sum_{k=1}^{(N_B+1)^d} (x_{\mathbf{i}_{kp2}} - x_{p2}) N_1(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n)(-2)^{(i_{kp1}-1) \bmod 2} \hat{v}_{\mathbf{i}_{kp}^n \alpha}^{n+1}}{\Delta x^4} \\
c_{p9\alpha} &= \sum_{k=1}^{(N_B+1)^d} \frac{1}{4\Delta x^4} (-2)^{(i_{kp1}-1) \bmod 2} (-2)^{(i_{kp2}-1) \bmod 2} \hat{v}_{\mathbf{i}_{kp}^n \alpha}^{n+1}
\end{aligned}$$

In 3D, $m_{\mathbf{i}_{kp}^n p}^n = m_p N(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n) = m_p N_1(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n) N_2(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n) N_3(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n)$, $\mathbf{c}$ can be computed

from the formula below:

$$c_{p1\alpha} = \sum_{k=1}^{(N_B+1)^d} N(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n)\hat{v}_{\mathbf{i}_{kp}^n\alpha}^{n+1}$$

$$c_{p2\alpha} = \frac{\sum_{k=1}^{(N_B+1)^d} N(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n)(x_{\mathbf{i}_{kp1}^n} - x_{p1})\hat{v}_{\mathbf{i}_{kp}^n\alpha}^{n+1}}{\frac{\Delta x^2}{4}}$$

$$c_{p3\alpha} = \frac{\sum_{k=1}^{(N_B+1)^d} N(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n)(x_{\mathbf{i}_{kp2}^n} - x_{p2})\hat{v}_{\mathbf{i}_{kp}^n\alpha}^{n+1}}{\frac{\Delta x^2}{4}}$$

$$c_{p4\alpha} = \frac{\sum_{k=1}^{(N_B+1)^d} N(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n)(x_{\mathbf{i}_{kp3}^n} - x_{p3})\hat{v}_{\mathbf{i}_{kp}^n\alpha}^{n+1}}{\frac{\Delta x^2}{4}}$$

$$c_{p5\alpha} = \frac{\sum_{k=1}^{(N_B+1)^d} N(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n)(x_{\mathbf{i}_{kp1}^n} - x_{p1})(x_{\mathbf{i}_{kp2}^n} - x_{p2})\hat{v}_{\mathbf{i}_{kp}^n\alpha}^{n+1}}{\frac{\Delta x^4}{16}}$$

$$c_{p6\alpha} = \frac{\sum_{k=1}^{(N_B+1)^d} N(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n)(x_{\mathbf{i}_{kp2}^n} - x_{p2})(x_{\mathbf{i}_{kp3}^n} - x_{p3})\hat{v}_{\mathbf{i}_{kp}^n\alpha}^{n+1}}{\frac{\Delta x^4}{16}}$$

$$c_{p7\alpha} = \frac{\sum_{k=1}^{(N_B+1)^d} N(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n)(x_{\mathbf{i}_{kp1}^n} - x_{p1})(x_{\mathbf{i}_{kp3}^n} - x_{p3})\hat{v}_{\mathbf{i}_{kp}^n\alpha}^{n+1}}{\frac{\Delta x^4}{16}}$$

$$c_{p8\alpha} = \frac{\sum_{k=1}^{(N_B+1)^d} N(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n)(x_{\mathbf{i}_{kp1}^n} - x_{p1})(x_{\mathbf{i}_{kp2}^n} - x_{p2})(x_{\mathbf{i}_{kp3}^n} - x_{p3})\hat{v}_{\mathbf{i}_{kp}^n\alpha}^{n+1}}{\frac{\Delta x^6}{64}}$$

$$c_{p9\alpha} = \frac{\sum_{k=1}^{(N_B+1)^d} N_2(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n) N_3(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n)(-2)^{(i_{kp1}-1) \bmod 2} \hat{v}_{\mathbf{i}_{kp}^n \alpha}^{n+1}}{2\Delta x^2}$$

$$c_{p10\alpha} = \frac{\sum_{k=1}^{(N_B+1)^d} N_1(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n) N_3(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n)(-2)^{(i_{kp2}-1) \bmod 2} \hat{v}_{\mathbf{i}_{kp}^n \alpha}^{n+1}}{2\Delta x^2}$$

$$c_{p11\alpha} = \frac{\sum_{k=1}^{(N_B+1)^d} N_1(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n) N_2(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n)(-2)^{(i_{kp3}-1) \bmod 2} \hat{v}_{\mathbf{i}_{kp}^n \alpha}^{n+1}}{2\Delta x^2}$$

$$c_{p12\alpha} = \frac{\sum_{k=1}^{(N_B+1)^d} N_3(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n)(-2)^{(i_{kp1}-1) \bmod 2}(-2)^{(i_{kp2}-1) \bmod 2} \hat{v}_{\mathbf{i}_{kp}^n \alpha}^{n+1}}{4\Delta x^4}$$

$$c_{p13\alpha} = \frac{\sum_{k=1}^{(N_B+1)^d} N_2(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n)(-2)^{(i_{kp1}-1) \bmod 2}(-2)^{(i_{kp3}-1) \bmod 2} \hat{v}_{\mathbf{i}_{kp}^n \alpha}^{n+1}}{4\Delta x^4}$$

$$c_{p14\alpha} = \frac{\sum_{k=1}^{(N_B+1)^d} N_1(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n)(-2)^{(i_{kp2}-1) \bmod 2}(-2)^{(i_{kp3}-1) \bmod 2} \hat{v}_{\mathbf{i}_{kp}^n \alpha}^{n+1}}{4\Delta x^4}$$

$$c_{p15\alpha} = \frac{\sum_{k=1}^{(N_B+1)^d} (-2)^{(i_{kp1}-1) \bmod 2}(-2)^{(i_{kp2}-1) \bmod 2}(-2)^{(i_{kp3}-1) \bmod 2} \hat{v}_{\mathbf{i}_{kp}^n \alpha}^{n+1}}{8\Delta x^6}$$

$$c_{p16\alpha} = \frac{\sum_{k=1}^{(N_B+1)^d} N_2(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n) N_3(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n)(x_{\mathbf{i}_{kp2}^n} - x_{p2})(-2)^{(i_{kp1}-1) \bmod 2} \hat{v}_{\mathbf{i}_{kp}^n \alpha}^{n+1}}{\frac{\Delta x^4}{2}}$$

$$c_{p17\alpha} = \frac{\sum_{k=1}^{(N_B+1)^d} N_2(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n) N_3(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n)(x_{\mathbf{i}_{kp3}^n} - x_{p3})(-2)^{(i_{kp1}-1) \bmod 2} \hat{v}_{\mathbf{i}_{kp}^n \alpha}^{n+1}}{\frac{\Delta x^4}{2}}$$

$$c_{p18\alpha} = \frac{\sum_{k=1}^{(N_B+1)^d} N_2(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n) N_3(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n)(x_{\mathbf{i}_{kp2}^n} - x_{p2})(x_{\mathbf{i}_{kp3}^n} - x_{p3})(-2)^{(i_{kp1}-1) \bmod 2} \hat{v}_{\mathbf{i}_{kp}^n \alpha}^{n+1}}{\frac{\Delta x^6}{8}}$$

$$c_{p19\alpha} = \frac{\sum_{k=1}^{(N_B+1)^d} N_1(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n) N_3(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n)(x_{\mathbf{i}_{kp1}^n} - x_{p1})(-2)^{(i_{kp2}-1) \bmod 2} \hat{v}_{\mathbf{i}_{kp}^n \alpha}^{n+1}}{\frac{\Delta x^4}{2}}$$

$$c_{p20\alpha} = \frac{\sum_{k=1}^{(N_B+1)^d} N_1(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n) N_3(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n)(x_{\mathbf{i}_{kp3}^n} - x_{p3})(-2)^{(i_{kp2}-1) \bmod 2} \hat{v}_{\mathbf{i}_{kp}^n \alpha}^{n+1}}{\frac{\Delta x^4}{2}}$$

$$c_{p21\alpha} = \frac{\sum_{k=1}^{(N_B+1)^d} N_1(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n) N_3(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n)(x_{\mathbf{i}_{kp1}^n} - x_{p1})(x_{\mathbf{i}_{kp3}^n} - x_{p3})(-2)^{(i_{kp2}-1) \bmod 2} \hat{v}_{\mathbf{i}_{kp}^n \alpha}^{n+1}}{\frac{\Delta x^6}{8}}$$

$$c_{p22\alpha} = \frac{\sum_{k=1}^{(N_B+1)^d} N_1(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n) N_2(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n)(x_{\mathbf{i}_{kp1}^n} - x_{p1})(-2)^{(i_{kp3}-1) \bmod 2} \hat{v}_{\mathbf{i}_{kp}^n \alpha}^{n+1}}{\frac{\Delta x^4}{2}}$$

$$c_{p23\alpha} = \frac{\sum_{k=1}^{(N_B+1)^d} N_1(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n) N_2(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n)(x_{\mathbf{i}_{kp2}^n} - x_{p2})(-2)^{(i_{kp3}-1) \bmod 2} \hat{v}_{\mathbf{i}_{kp}^n \alpha}^{n+1}}{\frac{\Delta x^4}{2}}$$

$$c_{p24\alpha} = \frac{\sum_{k=1}^{(N_B+1)^d} N_1(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n) N_2(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n)(x_{\mathbf{i}_{kp1}^n} - x_{p1})(x_{\mathbf{i}_{kp2}^n} - x_{p2})(-2)^{(i_{kp3}-1) \bmod 2} \hat{v}_{\mathbf{i}_{kp}^n \alpha}^{n+1}}{\frac{\Delta x^6}{8}}$$

$$c_{p25\alpha} = \frac{\sum_{k=1}^{(N_B+1)^d} N_3(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n)(x_{\mathbf{i}_{kp3}^n} - x_{p3})(-2)^{(i_{kp1}-1) \bmod 2}(-2)^{(i_{kp2}-1) \bmod 2} \hat{v}_{\mathbf{i}_{kp}^n \alpha}^{n+1}}{\Delta x^6}$$

$$c_{p26\alpha} = \frac{\sum_{k=1}^{(N_B+1)^d} N_2(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n)(x_{\mathbf{i}_{kp2}^n} - x_{p2})(-2)^{(i_{kp1}-1) \bmod 2}(-2)^{(i_{kp3}-1) \bmod 2} \hat{v}_{\mathbf{i}_{kp}^n \alpha}^{n+1}}{\Delta x^6}$$

$$c_{p27\alpha} = \frac{\sum_{k=1}^{(N_B+1)^d} N_1(\mathbf{x}_{\mathbf{i}_{kp}^n} - \mathbf{x}_p^n)(x_{\mathbf{i}_{kp1}^n} - x_{p1})(-2)^{(i_{kp2}-1) \bmod 2}(-2)^{(i_{kp3}-1) \bmod 2} \hat{v}_{\mathbf{i}_{kp}^n \alpha}^{n+1}}{\Delta x^6}$$

3 PolyPIC is loss less

In this section, we prove that if we use PolyPIC to transfer grid momentum to particle and then directly transfer back without advecting, we get the exact same grid momentum back. For simplicity, we prove it for the one particle case.

Given grid mass $\mathbf{M}_p^n$ and grid velocity $\mathcal{V}_p^n$, the $\mathbf{c}_p^{n+1}$ we find by using full-interpolation PolyPIC is given by

$$\mathbf{c}_p^{n+1} = ({\mathbf{Q}_P^n}^T \mathbf{M}_p^n \mathbf{Q}_p^n)^{-1} \mathbf{Q}_p^{nT} \mathbf{M}_p^n \mathcal{V}_p^n.$$

We want to show that the momentum $\hat{\mathbf{p}}$ we get from $\hat{\mathbf{p}} = \mathbf{M}_p^n \mathbf{Q}_p^n \mathbf{c}_p^{n+1}$ is equal to the original momentum on the grid $\mathbf{M}_p^n \mathbf{v}_i$. The key obversation is that $\mathbf{Q}_p^n$ is invertible: $(\mathbf{Q}_p^n)^T \mathbf{M}_p^n \mathbf{Q}_p^n$ is full rank diagonal, which means that $\mathbf{Q}_p^n$ is also full rank and therefore invertible.

$$\begin{aligned}
\hat{\mathbf{p}} &= \mathbf{M}_p^n \mathbf{Q}_p^n \mathbf{c}_p^{n+1} \\
&= (\mathbf{Q}_p^{n-T} \mathbf{Q}_p^{nT}) \mathbf{M}_p^n \mathbf{Q}_p^n \mathbf{c}_p^{n+1} \\
&= (\mathbf{Q}_p^{n-T} \mathbf{Q}_p^{nT}) \mathbf{M}_p^n \mathbf{Q}_p^n (\mathbf{Q}_p^{nT} \mathbf{M}_p^n \mathbf{Q}_p^n)^{-1} \mathbf{Q}_p^{nT} \mathbf{M}_p^n \mathcal{V}_p^n \\
&= (\mathbf{Q}_p^n)^{-T} (\mathbf{Q}_p^{nT} \mathbf{M}_p^n \mathbf{Q}_p^n)(\mathbf{Q}_p^{nT} \mathbf{M}_p^n \mathbf{Q}_p^n)^{-1} \mathbf{Q}_p^{nT} \mathbf{M}_p^n \mathcal{V}_p^n \\
&= \mathbf{M}_p^n \mathcal{V}_p^n
\end{aligned}$$

4 PolyPIC is linear and angular momentum conserving

The ith component of the local linear momentum associated with velocity $\mathcal{U}$ is $\mathbf{C}_i^T (\mathbf{Q}_p^n)^T \mathbf{M}_p^n \mathcal{U}$ and the j^{th} component of the angular velocity is $\mathbf{C}_{j+d}^T (\mathbf{Q}_p^n)^T \mathbf{M}_p^n \mathcal{U}$ where $C_{ki} = \delta_{ki}$ for $1 \le k \le d$ and

$$C_{k(j+d)} = \begin{cases} 1, & k = 4 \\ -1, & k = 5 \end{cases}$$

when $d = 2$ and

$$C_{k4} = \begin{cases} 1, & k = 5 \\ -1, & k = 7 \end{cases}$$

$$C_{k5} = \begin{cases} 1, & k = 6 \\ -1, & k = 10 \end{cases}$$

$$C_{k6} = \begin{cases} 1, & k = 9 \\ -1, & k = 11 \end{cases}$$

Thus local linear and angular momentum conservation (which implies global) follows from

$$\mathbf{C}_i^T (\mathbf{Q}_p^n)^T \mathbf{M}_p^n \mathcal{V}_p^{n+1} = \mathbf{C}_i^T (\mathbf{Q}_p^n)^T \mathbf{M}_p^n \mathbf{Q}_p^n \mathbf{c}_p^{n+1}$$

5 Grid momentum update

In the case of the incompressible Euler, we used a MAC grid discretization of the pressure projection to update the fluid velocity. We first compute an intermediate velocity field on the grid

$$\mathbf{v}_\mathbf{i}^* = \mathbf{v}_\mathbf{i}^n + \Delta t \mathbf{g},$$

where $\mathbf{g}$ is the gravitational acceleration. Then we set up a Poisson system for pressure p

$$\nabla^2 p = \frac{\rho}{\Delta t} \nabla \cdot \mathbf{v}_\mathbf{i}^*$$

to project grid velocities to be divergence free. Note that velocity is discretized at cell faces and pressure is discretized at cell centers. During the solve we enforce Dirichlet $p = 0$ boundary condition at free surface cells and zero Neumann boundary condition at collision object cells. Then we compute the updated velocity with

$$\hat{\mathbf{v}}_\mathbf{i}^{n+1} = \mathbf{v}_\mathbf{i}^* + \frac{\Delta t}{\rho} \nabla p.$$

In the case of elastoplastic solids and MPM the updates is from the elastic force.

$$\hat{\mathbf{v}}_\mathbf{i}^{n+1} = \mathbf{v}_\mathbf{i}^n + \frac{\Delta t}{m_\mathbf{i}^n} (\mathbf{f} + \mathbf{g})$$

where $\mathbf{f}$ is the elastic force. If explicit time integration is adopted, $\mathbf{f}_\mathbf{i}^n$ is given explicitly as

$$\mathbf{f}_\mathbf{i}^n = -\sum_p V_p^n \boldsymbol{\sigma}_p^n \nabla w_{\mathbf{i}p}^n,$$

where $V_p^n = V_p^0 J_p^n$ is the current volume of particle p, V_p^0 is its original volume, $J_p^n = \det(\mathbf{F}_p^n)$ and $\mathbf{F}_p^n$ is the current deformation gradient. Cauchy stress $\boldsymbol{\sigma}_p^n$ can be computed from $\mathbf{F}_p^n$ using any elastic consitutive model. For hyperelasticity with energy density function $\Psi(\mathbf{F})$, $\boldsymbol{\sigma}_p^n$ is given by

$$\boldsymbol{\sigma}_p^n = \frac{1}{J_p^n} \frac{\partial \Psi}{\partial \mathbf{F}} (\mathbf{F}_p^n) \mathbf{F}_p^{nT}.$$

For implicit time integration such as Backward Euler, we need to differentiate the force with respect to imaginarily deformed grid node positions $\hat{\mathbf{x}}_\mathbf{i} = \mathbf{x}_\mathbf{i}^n + \Delta t \hat{\mathbf{v}}_\mathbf{i}$. The force differential on an arbirary increment $\delta \mathbf{u}$ is given by

$$\delta f_{i\alpha} = \frac{\partial f_{i\alpha}}{\partial x_{j\lambda}} \delta u_{j\lambda} = -\sum_p V_p^0 \frac{\partial F_{\beta\zeta}^E}{\partial x_{i\alpha}} \frac{\partial^2 \psi}{\partial F_{\beta\zeta}^E \partial F_{\omega\sigma}^E} \frac{\partial F_{\omega\sigma}^E}{\partial x_{j\lambda}} \delta u_{j\lambda}.$$

Based on that we solve

$$\hat{\mathbf{v}}_{\mathbf{i}}^{n+1} = \mathbf{v}_{\mathbf{i}}^{n} + \frac{\Delta t}{m_{\mathbf{i}}^{n}}(\mathbf{f}(\mathbf{x}_{\mathbf{i}}^{n} + \Delta t \hat{\mathbf{v}}_{\mathbf{i}}^{n+1}) + \mathbf{g})$$

using Newton's method.

6 Mathematica code

In this section we present the mathematica code we use to generate the basis and corresponding formula.

6.1 Linear interpolation in 2d

```
(* 2D linear *)
ClearAll[ "Global*"]
(* Assume x is in [0,h],
N[x,1] is the weight of node at xi = 0,
N[x,2] is the weight of node at h.
*)
N1[ x_, i_] = Piecewise[{{1 - x, i == 1}, {x, i == 2}}];
NN[ x_, y_, ii_, jj_] = N1[x/h, ii] * N1[y/h, jj];
(* xi - xp *)
r = ConstantArray[0, {4, 2}];
Do[{id = 2 * ( i1 - 1) + j1;
nodex = ( i1 - 1) * h;
nodey = ( j1 - 1) * h;
r[[id]][[1]] = (nodex - particlex)/h;
r[[id]][[2]] = (nodey - particley)/h; }, { i1, 1, 2}, { j1, 1, 2}];
(* mass *)
M = ConstantArray[0, {4, 4}];
Do[{id = 2 * ( i1 - 1) + j1;
weight = NN[ particlex, particley, i1, j1];
M[[id]][[id]] = mass * weight; }, { i1, 1, 2}, { j1, 1, 2}];
(* basis *)
(* row index is node index
```

```
 col index is basis index
*)
Do[{id = 2 * ( i1 - 1) +  j1;
 weight =  NN[ particlex,  particley,  i1,  j1];
M[[id]][[id]] =  mass *  weight; }, { i1, 1, 2}, { j1, 1, 2}];
B1 = ConstantArray[0, {4, 4}];
Do[{id = ( i1 - 1) * 2 +  j1;
B1[[idr]][[id]]  = r[[idr]][[1]]^( i1 - 1) * r[[idr]][[2]]^( j1 - 1)}, { i1, 1, 2}, { j1, 1, 2}, {idr, 1, 4}];
Diagonal[M] // MatrixForm;
(* Verify the diagonal structure *)
MatrixForm[ Transpose[ B1].M. B1] // Simplify
```

6.2 Linear interpolation in 3d

```
(* 3D linear *)
ClearAll[ "Global*"]
(* Assume x is in[0, h],
N[x, 1] is the weight of node at xi = 0,
N[x, 2] is the weight of node ath.
*)
N1[ x_,  i_] = Piecewise[{{1 - x, i == 1}, {x, i == 2}}];
NN[ x_,  y_,  z_,  ii_,  jj_,  kk_] =  N1[x/h,  ii] *  N1[y/h,  jj] *  N1[z/h,  kk];
(* xi - xp *)
r = ConstantArray[0, {8, 3}];
Do[{id = 4 * ( i1 - 1) + 2 * ( j1 - 1) +  k1;
nodex = ( i1 - 1) * h;
nodey = ( j1 - 1) * h;
nodez = ( k1 - 1) * h;
r[[id]][[1]] = ( nodex -  particlex)/h;
r[[id]][[2]] = ( nodey -  particley)/h;
```

```
r[[id]][[3]] = ( nodez -  particlez)/h; }, { i1, 1, 2}, { j1, 1, 2}, { k1, 1, 2}];
 (* mass *)
M = ConstantArray[0, {8, 8}];
 Do[{id = 4 * ( i1 - 1) + 2 * ( j1 - 1) +  k1;
 weight =  NN[ particlex,  particley,  particlez,  i1,  j1,  k1];
M[[id]][[id]] =  mass *  weight; }, { i1, 1, 2}, { j1, 1, 2}, { k1, 1, 2}];
 (* basis *)
 (* row index is node index
  col index is basis index
 *)
 B1 =  ConstantArray[0, {4 * 2, 8}];
 Do[{id = ( i1 - 1) * 4 + ( j1 - 1) * 2 +  k1;
 B1[[idr]][[id]]  = r[[idr]][[1]]^( i1 - 1) * r[[idr]][[2]]^( j1 - 1) * r[[idr]][[3]]^( k1 - 1); },
     {i1, 1, 2}, { j1, 1, 2}, { k1, 1, 2}, {idr, 1, 8}];
 Diagonal[M] // MatrixForm;
 (* Verify the diagonal structure *)
 MatrixForm[ Transpose[ B1].M. B1] // Simplify
```

6.3 Quadratic interpolation in 2d

```
(* 2D quadratic *)
ClearAll["Global*"]
(* Assume x is in [-0.5h, 0.5h], we use quadratic interpolation.
N2[x, 1] is the weight of node at xi = -h,
 N2[x, 2] is the weight of node at xi = 0,
 N2[x, 3] is the weight of node at xi = h,
*)
N2[x_, i_] = Piecewise[{{1/2 * (1/2 - x)^2, i == 1}, {3/4 - x^2, i == 2}, {1/2 * (x + 1/2)^2, i == 3}}];
NN[x_, y_, ii_, jj_] = N2[x/h, ii] * N2[y/h, jj];
```

```
(* xi - xp *)
r = ConstantArray[0, {9, 2}];
Do[{id = 3 * (i1 - 1) + j1;
nodex = (i1 - 2) * h;
nodey = (j1 - 2) * h;
r[[id]][[1]] = (nodex - x);
r[[id]][[2]] = (nodey - y); }, {i1, 1, 3}, {j1, 1, 3}];
(* mass *)
M = ConstantArray[0, {9, 9}];
Do[{id = 3 * (i1 - 1) + j1;
weight = NN[x, y, i1, j1];
M[[id]][[id]] = mass * weight; }, {i1, 1, 3}, {j1, 1, 3}];
(* basis *)
(* row index is node index
 col index is basis index
*)
B = ConstantArray[0, {9, 9}];
Do[{id = (j1 - 1) * 3 + i1;
B[[idr]][[id]]  = r[[idr]][[1]]^(i1 - 1) * r[[idr]][[2]]^(j1 - 1);
}, {i1, 1, 3}, {j1, 1, 3}, {idr, 1, 9}];
(* Rearrange the basis so that the corresponding polynomials arein the order of
 1, x, y, xy, x^2, y^2, x^2 * y, x * y^2, x^2 * y^2.
*)
BS = B[[All, {1, 2, 4, 5, 3, 7, 6, 8, 9}]];
(* The first four basis are already othogonal.
Use Gram-Schmidt and we get the basis corresponding to x^2 and y^2.
*)
BS[[All, 5]] = BS[[All, 5]] - h^2/4 - x(h^2 - 4x^2)/h^2 * BS[[All, 2]];
BS[[All, 6]] = BS[[All, 6]] - h^2/4 - y(h^2 - 4y^2)/h^2 * BS[[All, 3]];
(* The rest are simply products of the previous ones *)
```

BS[[All, 7]] = BS[[All, 5]]BS[[All, 3]];

BS[[All, 8]] = BS[[All, 6]]BS[[All, 2]];

BS[[All, 9]] = BS[[All, 5]] BS[[All, 6]];

(* Verify the diagonal structure *)

BTMB = Transpose[BS].M.BS//Simplify;

(* Get the awesome formula to put in your code!*)

MB = M.BS//Simplify;

MB//MatrixForm;

Inverse[BTMB].Transpose[BS].M//MatrixForm//Simplify;

6.4 Quadratic interpolation in 3d

(* 3D quadratic *)

ClearAll["Global*"]

(* Assume$x$ is in$[-0.5h, 0.5h]$, we use quadratic interpolation.

N2[x, 1] is the weight of node at xi = $-h$,

N2[x, 2] is the weight of node at xi = 0,

N2[x, 3] is the weight of node at xi = h,

*)

N2[x_, i_] = Piecewise[{{1/2 * (1/2 - x)^2, i == 1}, {3/4 - x^2, i == 2}, {1/2 * (x + 1/2)^2, i == 3}}];

NN[x_, y_, z_, ii_, jj_, kk_] = N2[x/h, ii] * N2[y/h, jj] * N2[z/h, kk];

(* xi - xp *)

r = ConstantArray[0, {27, 3}];

Do[{id = 9 * (i1 - 1) + 3 * (j1 - 1) + k1;

nodex = (i1 - 2) * h;

nodey = (j1 - 2) * h;

nodez = (k1 - 2) * h;

r[[id]][[1]] = (nodex - x);

r[[id]][[2]] = (nodey - y);

r[[id]][[3]] = (nodez - z); }, { i1, 1, 3}, { j1, 1, 3}, { k1, 1, 3}];

```
(* mass *)
M = ConstantArray[0, {27, 27}];
Do[{id = 9 * ( i1 - 1) + 3 * ( j1 - 1) + k1;
weight = NN[x, y, z, i1, j1, k1];
M[[id]][[id]] = mass * weight; }, { i1, 1, 3}, { j1, 1, 3}, { k1, 1, 3}];
(* basis *)
(* row index is node index
 col index is basis index
*)
B = ConstantArray[0, {27, 27}];
Do[{id = ( i1 - 1) * 4 + ( j1 - 1) * 2 + k1;
B[[idr]][[id]] = r[[idr]][[1]]^( i1 - 1) * r[[idr]][[2]]^( j1 - 1) * r[[idr]][[3]]^( k1 - 1);
}, { i1, 1, 2}, { j1, 1, 2}, { k1, 1, 2}, {idr, 1, 27}];
(* The basis correponding to1, x, y, z are already othogonal.
Use gram - schmidt and we get the basis corresponding tox^2, y^2, andz^2.
*)
B[[All, 9]] = B[[All, 5]]B[[All, 5]] - h^2/4 - x(h^2 - 4x^2)/h^2 * B[[All, 5]];
B[[All, 10]] = B[[All, 3]]B[[All, 3]] - h^2/4 - y(h^2 - 4y^2)/h^2 * B[[All, 3]];
B[[All, 11]] = B[[All, 2]]B[[All, 2]] - h^2/4 - z(h^2 - 4z^2)/h^2 * B[[All, 2]];

(* The rest are simply products of the previous ones *)
B[[All, 12]] = B[[All, 2]]B[[All, 9]];
B[[All, 13]] = B[[All, 3]]B[[All, 9]];
B[[All, 14]] = B[[All, 2]]B[[All, 3]]B[[All, 9]];

B[[All, 15]] = B[[All, 2]]B[[All, 10]];
B[[All, 16]] = B[[All, 5]]B[[All, 10]];
B[[All, 17]] = B[[All, 2]]B[[All, 5]]B[[All, 10]];

B[[All, 18]] = B[[All, 3]]B[[All, 11]];
```

```
B[[All, 19]] = B[[All, 5]]B[[All, 11]];
B[[All, 20]] = B[[All, 3]]B[[All, 5]]B[[All, 11]];

B[[All, 21]] = B[[All, 9]]B[[All, 10]];
B[[All, 22]] = B[[All, 10]]B[[All, 11]];
B[[All, 23]] = B[[All, 9]]B[[All, 11]];

B[[All, 24]] = B[[All, 21]]B[[All, 2]];
B[[All, 25]] = B[[All, 22]]B[[All, 5]];
B[[All, 26]] = B[[All, 23]]B[[All, 3]];

B[[All, 27]] = B[[All, 9]]B[[All, 10]]B[[All, 11]];

(*  Rearrange the basis so that the corresponding polynomials are in the order of
1, x, y, z,  xy,  xz,  yz,  xyz, x^2, y^2, z^2, x^2 * y^2, x^2 * z^2, y^2 * z^2, x^2 * y^2 * z^2,
x^2 * y,  x^2 * z,  x^2 *  yz,  y^2 * x, y^2 * z, y^2 *  xz, z^2 * x, z^2 * y, z^2 *  xy,
x^2 * y^2 * z, x^2 * z^2 * y, y^2 * z^2 * x
*)
BS = B[[All, {1, 5, 3, 2, 7, 6, 4, 8, 9, 10, 11, 21, 23, 22, 27, 13, 12, 14, 16, 15, 17, 19, 18, 20, 24, 26, 25}]];

(* Verify the diagonal structure *)
BTMB =  Transpose[ BS].M. BS // Simplify;
BTMB // MatrixForm
(* Get the awesome formula to put in your code! *)
MB = M. BS // Simplify;
MB // MatrixForm;
BTMBinvBTM =  Inverse[ BTMB]. Transpose[ BS].M;
Transpose[ BTMBinvBTM] // MatrixForm // Simplify;
```