MPM for Snow - Implementation Report

Nathan Gurrin-Smith

I. Parameters

In general, I found that the parameters given in Stomakhin et al., 2013 to be good ones to use, despite being in 2D. I found that for the grid, a spacing of $h = 0.05$ was best. For some reason, for h less than this, say 0.01, the snow would frequently just deflate into itself into a solid line. My suspicion is that with such a small grid size, there are less particles per grid and they may interact less (and thus slip around each other). Additionally, I found giving the snow a weight of 0.05 and a density of 100 to work best, as suggested by Nygaard et al., 2021.

II. Semi-Implicit Update

For the semi-implicit update, we need to solve the following equation (Stomakhin et al., 2013):

$$\sum_{\mathbf{j}} \left(I\delta_{\mathbf{ij}} + \beta \Delta t^2 m_{\mathbf{i}}^{-1} \frac{\partial \Phi^n}{\partial \hat{\mathbf{x}}_{\mathbf{i}} \partial \hat{\mathbf{x}}_{\mathbf{j}}} \right) \mathbf{v}_{\mathbf{j}}^{n+1} = \mathbf{v}_{\mathbf{i}}^* \tag{1}$$

The main barrier to doing this is computing $\partial \Phi^n / \partial \hat{\mathbf{x}}_{\mathbf{i}} \partial \hat{\mathbf{x}}_{\mathbf{j}}$. To do this, we utilize Jiang et al., 2016 to note that,

$$\frac{\partial^2 \Phi}{\partial \hat{\mathbf{x}}_{\mathbf{i}\alpha} \partial \hat{\mathbf{x}}_{\mathbf{j}\tau}} = \sum_p V_p^0 \frac{\partial^2 \Psi}{\partial F_{\alpha\beta} \partial F_{\tau\sigma}} \left(\nabla w_{\mathbf{j}p}^n \right)_\omega \left(\nabla w_{\mathbf{i}p}^n \right)_\gamma \left(F_p^n \right)_{\omega\sigma} \left(F_p^n \right)_{\gamma\beta} \tag{2}$$

Everything in this equation is given, except for $\frac{\partial^2 \Psi}{\partial F_{\alpha\beta} \partial F_{\tau\sigma}}$, which we shall compute now. Following Stomakhin et al., January 18, 2013, we note that,

$$\frac{\partial^2 \Psi}{\partial \mathbf{F} \partial \mathbf{F}} : \delta \mathbf{F} = 2\mu \delta \mathbf{F} - 2\mu \delta \mathbf{R} + \lambda J \mathbf{F}^{-T} \left(J \mathbf{F}^{-T} : \delta \mathbf{F} \right) + \lambda (J-1) \delta \left(J \mathbf{F}^{-T} \right) \tag{3}$$

The goal here is to write the RHS in the form $X : \delta \mathbf{F}$ and we can then set $\frac{\partial^2 \Psi}{\partial \mathbf{F} \partial \mathbf{F}} = X$. For the first term we have,

$$\delta \mathbf{F} = \frac{\partial \mathbf{F}}{\partial \mathbf{F}} : \delta \mathbf{F} = \begin{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \\ \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \end{pmatrix} : \delta \mathbf{F} \tag{4}$$

The second term is more complicated. Following the guidance providedin Stomakhin et al., January 18, 2013, we seek to solve the following for $R^T \delta R$ (dropping bolds for now).

$$R^T \delta F - \delta F^T R = YS + SY, \quad Y = R^T \delta R \tag{5}$$

after which we can set $\delta R = RY$. First, note that Y is skew-symmetric since,

$$\delta \left(R^T R \right) = \delta I = 0$$
$$Y = R^T \delta R = -\delta R^T R = -\left(R^T \delta R \right)^T$$

Since Y is symmetric, its diagonal is zero and $Y_{21} = -Y_{12}$. Additionally, since S is symmetric and $(YS + SY)^T = S^T Y^T + Y^T S^T = -SY - YS$, we get that the equation is skew-symmetric. Thus, we only need to look at the top-right entry of the equation to solve it. Doing so,

$$\begin{aligned}
R^T\delta F &= \begin{pmatrix} R_{11} & R_{21} \\ R_{12} & R_{22} \end{pmatrix}\begin{pmatrix} \delta F_{11} & \delta F_{12} \\ \delta F_{21} & \delta F_{22} \end{pmatrix} = \begin{pmatrix} * & R_{11}\delta F_{12} + R_{21}\delta F_{22} \\ * & * \end{pmatrix} \\
\delta F^T R &= \begin{pmatrix} \delta F_{11} & \delta F_{21} \\ \delta F_{12} & \delta F_{22} \end{pmatrix}\begin{pmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{pmatrix} = \begin{pmatrix} * & R_{12}\delta F_{11} + R_{22}\delta F_{21} \\ * & * \end{pmatrix} \\
YS &= \begin{pmatrix} 0 & Y_{12} \\ -Y_{12} & 0 \end{pmatrix}\begin{pmatrix} S_{11} & S_{12} \\ S_{12} & S_{22} \end{pmatrix} = \begin{pmatrix} * & Y_{12}S_{22} \\ * & * \end{pmatrix} \\
SY &= \begin{pmatrix} S_{11} & S_{12} \\ S_{12} & S_{22} \end{pmatrix}\begin{pmatrix} 0 & Y_{12} \\ -Y_{12} & 0 \end{pmatrix} = \begin{pmatrix} * & S_{11}Y_{12} \\ * & * \end{pmatrix}
\end{aligned}$$

Then, constructing the equation yields,

$$\begin{aligned}
R_{11}\delta F_{12} + R_{21}\delta F_{22} - R_{12}\delta F_{11} - R_{22}\delta F_{21} &= (S_{11} + S_{22})Y_{12} \\
Y_{12} &= \frac{1}{\mathrm{Tr}(S)}\begin{pmatrix} -R_{12} & R_{11} \\ -R_{22} & R_{21} \end{pmatrix} : \delta\mathbf{F} = \frac{1}{\mathrm{Tr}(S)} X : \delta\mathbf{F}
\end{aligned}$$

Finally, we get the following form for $\delta\mathbf{R}$,

$$\delta\mathbf{R} = \mathbf{R}\left(\mathbf{R}^T\delta\mathbf{R}\right) = \frac{1}{\mathrm{Tr}(\mathbf{S})}\begin{pmatrix} -R_{12}\mathbf{X} : \delta\mathbf{F} & R_{11}\mathbf{X} : \delta\mathbf{F} \\ -R_{22}\mathbf{X} : \delta\mathbf{F} & R_{21}\mathbf{X} : \delta\mathbf{F} \end{pmatrix} \tag{6}$$

Which we can rewrite as,

$$\delta\mathbf{R} = \frac{1}{\mathrm{Tr}(S)}\begin{pmatrix} -R_{12}\mathbf{X} & R_{11}\mathbf{X} \\ -R_{22}\mathbf{X} & R_{21}\mathbf{X} \end{pmatrix} : \delta\mathbf{F} \tag{7}$$

At last, we note that $\delta\left(J\mathbf{F}^{-T}\right) = \frac{\partial}{\partial\mathbf{F}} J\mathbf{F}^{-T} : \partial\mathbf{F}$. We note,

$$J\mathbf{F}^{-T} = J\frac{1}{J}\begin{pmatrix} F_{22} & -F_{21} \\ -F_{12} & F_{11} \end{pmatrix} = \begin{pmatrix} F_{22} & -F_{21} \\ -F_{12} & F_{11} \end{pmatrix} \tag{8}$$

Thus,

$$\frac{\partial}{\partial\mathbf{F}} J\mathbf{F}^{-T} : \partial\mathbf{F} = \begin{pmatrix} \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} & \begin{pmatrix} 0 & 0 \\ -1 & 0 \end{pmatrix} \\ \begin{pmatrix} 0 & -1 \\ 0 & 0 \end{pmatrix} & \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \end{pmatrix} : \delta\mathbf{F} \tag{9}$$

Putting all of this together yields,

$$\begin{aligned}
\frac{\partial^2\Psi}{\partial\mathbf{F}\partial\mathbf{F}} : \delta\mathbf{F} &= 2\mu\begin{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \\ \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \end{pmatrix} : \delta\mathbf{F} - \frac{2\mu}{\mathrm{Tr}(\mathbf{S})}\begin{pmatrix} -R_{12}\mathbf{X} & R_{11}\mathbf{X} \\ -R_{22}\mathbf{X} & R_{21}\mathbf{X} \end{pmatrix} : \delta\mathbf{F} \\
&\quad + \lambda J\mathbf{F}^{-T}\left(J\mathbf{F}^{-T} : \delta\mathbf{F}\right) + \lambda(J-1)\begin{pmatrix} \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} & \begin{pmatrix} 0 & 0 \\ -1 & 0 \end{pmatrix} \\ \begin{pmatrix} 0 & -1 \\ 0 & 0 \end{pmatrix} & \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \end{pmatrix} : \delta\mathbf{F} \\
&= \Bigg(2\mu\begin{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \\ \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \end{pmatrix} - \frac{2\mu}{\mathrm{Tr}(\mathbf{S})}\begin{pmatrix} -R_{12}\mathbf{X} & R_{11}\mathbf{X} \\ -R_{22}\mathbf{X} & R_{21}\mathbf{X} \end{pmatrix} \\
&\quad + \lambda J\begin{pmatrix} F_{22}\mathbf{F}^{-T} & -F_{21}\mathbf{F}^{-T} \\ -F_{12}\mathbf{F}^{-T} & F_{11}\mathbf{F}^{-T} \end{pmatrix} + \lambda(J-1)\begin{pmatrix} \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} & \begin{pmatrix} 0 & 0 \\ -1 & 0 \end{pmatrix} \\ \begin{pmatrix} 0 & -1 \\ 0 & 0 \end{pmatrix} & \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \end{pmatrix}\Bigg) : \delta\mathbf{F}
\end{aligned}$$

Thus, the derivative is,

$$\frac{\partial^2 \Psi}{\partial \mathbf{F} \partial \mathbf{F}} = \begin{pmatrix} 2\mu \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + \frac{2\mu R_{12}}{\mathrm{Tr}(\mathbf{S})}\mathbf{X} + \lambda F_{22} J \mathbf{F}^{-T} + \lambda(J-1)\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} & 2\mu \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} - \frac{2\mu R_{11}}{\mathrm{Tr}(\mathbf{S})}\mathbf{X} - \lambda F_{21} J \mathbf{F}^{-T} + \lambda(J-1)\begin{pmatrix} 0 & 0 \\ -1 & 0 \end{pmatrix} \\ 2\mu \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} + \frac{2\mu R_{22}}{\mathrm{Tr}(\mathbf{S})}\mathbf{X} - \lambda F_{12} J \mathbf{F}^{-T} + \lambda(J-1)\begin{pmatrix} 0 & -1 \\ 0 & 0 \end{pmatrix} & 2\mu \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} - \frac{2\mu R_{21}}{\mathrm{Tr}(\mathbf{S})}\mathbf{X} + \lambda F_{11} J \mathbf{F}^{-T} + \lambda(J-1)\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \end{pmatrix} \tag{10}$$

We'll write this in index notation as A_{ijkl}, where ij determine which outer matrix we are in, and kl will determine which sub-matrix we are in. Subsituting this into Equation 2 yields,

$$\frac{\partial^2 \Phi}{\partial x_{\mathbf{i}\alpha} \partial x_{\mathbf{j}\tau}} = \sum_p V_p^0 A_{\tau\sigma\alpha\beta} \left(\nabla w_{\mathbf{j}p}^n \cdot F_{p\cdot\sigma}^n \right) \left(\nabla w_{\mathbf{i}p}^n \cdot F_{p\cdot\beta}^n \right) \tag{11}$$

Where here, using the normal conventions of index notation, σ and β are being summed over.

A note on computational efficiency: since Ψ is smooth, $\frac{\partial \Phi^n}{\partial \hat{\mathbf{x}}_\mathbf{i} \partial \hat{\mathbf{x}}_\mathbf{j}}$ is symmetric.

III. Efficiency Improvements

III.1. Pre-Computations

The following values are utilized for multiple steps, so they are pre-computed and stored:

- $J_{E_p} = \det \mathbf{F}_{E_p}$
- $J_{P_p} = \det \mathbf{F}_{P_p}$

III.2. Interpolation Checks

We can reduce computational cost by not doing computations if the interpolation value $w_{\mathbf{i}p} = 0$. This may seem only slightly beneficial, but the benefits are much greater with the following three observations:

Lemma 1. *If $w_{\mathbf{i}p} = 0$ then $\nabla w_{\mathbf{i}p} = \mathbf{0}$.*

Proof. Suppose $w_{\mathbf{ip}} = N\big((x-ih)/h\big)N\big((y-jh)/h\big) = 0$. Then, either $N\big((x-ih)/h\big) = 0$ or $N\big((y-jh)/h\big) = 0$. Without loss of generality, let $N\big((x-ih)/h\big) = 0$. Then, $\frac{N}{\partial z}\big((x-ih)/h\big) = 0$ almost everywhere. Thus,

$$\nabla w_{\mathbf{i}p_1} = \frac{1}{h} \underbrace{\frac{\partial N}{\partial z}\left(\frac{x-ih}{h}\right)}_{=0} N\left(\frac{y-jh}{h}\right) = 0$$

$$\nabla w_{\mathbf{i}p_2} = \frac{1}{h} \underbrace{N\left(\frac{x-ih}{h}\right)}_{=0} \frac{\partial N}{\partial z}\left(\frac{y-jh}{h}\right) = 0$$

□

Lemma 2. *If $m_\mathbf{i} = 0$ then $w_{\mathbf{i}p} = 0$ for all p and $\nabla w_{\mathbf{i}p} = \mathbf{0}$ for all p.*

Proof. Suppose $m_\mathbf{i} = \sum_p m_p w_{\mathbf{i}p} = 0$. Then, since $m_p > 0$ for all p and $w_{\mathbf{i}p} \geq 0$ for all $\mathbf{i}$, p, we must have $w_{\mathbf{i}p} = 0$ for all p. The second result $\nabla w_{\mathbf{i}p} = \mathbf{0}$ follows from lemma 1. □

Lemma 3. *If $m_\mathbf{i} = 0$ then we can ignore all computations involving grid node $\mathbf{i}$.*

Proof. Suppose that $m_\mathbf{i} = 0$ so that by Lemma 2 $w_{\mathbf{i}p} = 0$ for all p and $\nabla w_{\mathbf{i}p} = \mathbf{0}$ for all p. First, note that nowhere in the (explicit integration) algorithm does a property of one grid node affect the properties of another grid node. Thus, we only have to look at when the grid is transferred back to the particles in step 7 and step 8.

- Step 1: There are no particles here so we'll set $\mathbf{v}_\mathbf{i}^n = 0$.

- Only the grid mass is used here.
- Step 3: Here, $\mathbf{f_i} = -\sum_p V_p \sigma_p \nabla w_{\mathbf{i}p}$. If $m_{\mathbf{i}} = 0$ then we can skip the sum entirely and set $\mathbf{f_i} = \mathbf{0}$ since $\nabla w_{\mathbf{i}p} = \mathbf{0}$ for all p. If only $w_{\mathbf{i}p} = 0$, then we can skip the matrix product summand since $\nabla w_{\mathbf{i}p} = \mathbf{0}$.
- Step 4: Since $\mathbf{v_i} = \mathbf{0}$ and $\mathbf{f_i} = \mathbf{0}$, we set $\mathbf{v}^* = \mathbf{0}$ here.
- Step 5: Since we are only updating the grid velocities with collision effects, we skip the collisions for index $\mathbf{i}$.
- Step 7: The computation here is $\nabla \mathbf{v}_p^{n+1} = \sum_j \mathbf{v}_{\mathbf{j}}^{n+1} \left(\nabla w_{\mathbf{j}p}^n\right)^T$. Since $\nabla w_{\mathbf{i}p} = \mathbf{0}$, the value of $\mathbf{v}_{\mathbf{i}}^{n+1}$ has no effect.
- Step 8: The computations here are $\sum_{\mathbf{j}} \mathbf{v}_{\mathbf{j}}^{n+1} w_{\mathbf{j}p}^n$ and $\sum_{\mathbf{j}} \mathbf{v}_{\mathbf{j}}^n w_{\mathbf{j}p}^n$. Since $\mathbf{v}_{\mathbf{j}}^{n+1}$ and $\mathbf{v}_{\mathbf{j}}^n$ are being multiplied by $w_{\mathbf{i}p}^n = 0$, it doesn't matter what value they have.

□

III.3. Grid-to-Particle Velocity Transfer

The method used by the authors to transfer the new grid velocities to the new particle velocities was (with N grid nodes):

$$\begin{aligned}
v_p^{n+1} &= (1-\alpha) v_{\mathrm{PIC}_p}^{n+1} + \alpha v_{\mathrm{FLIP}_p^{n+1}} && \text{4 operations} \\
v_{\mathrm{PIC}_p}^{n+1} &= \sum_i v_i^{n+1} w_{ip}^n && 2N-1 \text{ operations} \\
v_{\mathrm{FLIP}_p^{n+1}} &= v_p^n + \sum_i \left(v_i^{n+1} - v_i^n\right) w_{ip}^n && 3N \text{ operations}
\end{aligned}$$

Thus, the total computational cost is $5N+3$ operations per particle. This can be simplified as follows,

$$\begin{aligned}
v_p^{n+1} &= (1-\alpha)\sum_i v_i^{n+1} w_{ip}^n + \alpha v_p^n + \alpha \sum_i v_i^{n+1} w_{ip}^n - \alpha \sum_i v_i^n w_{ip}^n \\
&= (1-\alpha+\alpha)\sum_i v_i^{n+1} w_{ip}^n + \alpha v_p^n - \alpha \sum_i v_i^n w_{ip}^n \\
&= \underbrace{\underbrace{\sum_i v_i^{n+1} w_{ip}^n}_{2N-1} + \underbrace{\alpha \underbrace{\left(v_p^n - \sum_i v_i^n w_{ip}^n\right)}_{2N}}_{1}}_{1}
\end{aligned}$$

This has a total cost of $4N+1$ operations per particle, a slight improvement!

References

Jiang, C., Schroeder, C., Teran, J., Stomakhin, A., & Selle, A. (2016). The material point method for simulating continuum materials. *ACM SIGGRAPH 2016 Courses*. https://doi.org/10.1145/2897826.2927348

Nygaard, I., melissaEdge, & Oborn, J. (2021). Snow. https://github.com/Azmisov/snow/

Stomakhin, A., Schroeder, C., Chai, L., Teran, J., & Selle, A. (2013). A material point method for snow simulation. *ACM Trans. Graph.*, *32*(4). https://doi.org/10.1145/2461912.2461948

Stomakhin, A., Schroeder, C., Chai, L., Teran, J., & Selle, A. (January 18, 2013). Material point method for snow simulation technical report. https://www.disneyanimation.com/publications/a-material-point-method-for-snow-simulation/