Effective time step restrictions for explicit MPM simulation

Contents

1 Simple time step restrictions

In this section, we derive the time step restrictions computed by the functions VELOCITY_DT, POSITION_DT, and DEFORMATION_GRADIENT_DT (lines 2, 6, and 7 in Algorithm 1, respectively).

1.1 Velocity

The velocity transferred from particle p to grid node i is given by $\mathbf{v}_{ip}^n = \mathbf{v}_p^n + \mathbf{B}_p^n \mathbf{D}_p^{-1}(\mathbf{x}_i^n - \mathbf{x}_p^n)$, where $\mathbf{B}_p = \mathbf{0}$ for non-APIC transfers. Let $d = 2, 3$ be the dimension, and let $o = 2, 3$ be the spline order (quadratic or cubic), so that $\mathbf{D}_p^{-1} = \frac{6-o}{\Delta x^2}\mathbf{I}$ [8]. An upper bound for $\|\mathbf{v}_{ip}^n\|$ over i is given by

$$
\begin{aligned}
\max_i \|\mathbf{v}_{ip}^n\| &= \max_i \|\mathbf{v}_p^n + \mathbf{B}_p^n \mathbf{D}_p^{-1}(\mathbf{x}_i^n - \mathbf{x}_p^n)\| \\
&\leq \max_p \left(\|\mathbf{v}_p\| + \frac{6-o}{\Delta x^2} \|\mathbf{B}_p^n\|_F \max_i \|\mathbf{x}_i^n - \mathbf{x}_p^n\| \right) \\
&\leq \max_p \left(\|\mathbf{v}_p\| + \frac{(6-o)(o+1)\sqrt{d}}{2\Delta x} \|\mathbf{B}_p^n\|_F \right) \\
&= \max_p \left(\|\mathbf{v}_p\| + \frac{6\sqrt{d}}{\Delta x} \|\mathbf{B}_p\|_F \right)
\end{aligned}
$$

Letting $s = \max_p \left(\|\mathbf{v}_p\| + \frac{6\sqrt{d}}{\Delta x} \|\mathbf{B}_p\|_F \right)$, the velocity time step restriction is then given by $\frac{\Delta x}{\Delta t} \geq s$, or

$$\Delta t = \nu_v \frac{\Delta x}{s}, \quad 0 < \nu_v \leq 1.$$

We choose $\nu_v = 1$ for all examples.

1.2 Particle displacement

We limit the time step so that the displacement of each particle is less than a grid cell. For each grid node i, explicit integration of forces on the grid gives

$$\tilde{\mathbf{v}}_i^{n+1} = \mathbf{v}_i^n + \frac{\Delta t}{m_i^n} \mathbf{f}_i.$$

The velocity and position of particle p is then updated as

$$\begin{aligned} \mathbf{v}_p^{n+1} &= \sum_i w_{ip}^n \tilde{\mathbf{v}}_i^{n+1} \\ \mathbf{x}_p^{n+1} &= \mathbf{x}_p^n + \Delta t \mathbf{v}_p^{n+1}, \end{aligned}$$

so that its displacement over the time step is given by

$$\mathbf{x}_p^{n+1} - \mathbf{x}_p^n = \Delta t \mathbf{v}_p^{n+1} = \Delta t \sum_i w_{ip}^n \tilde{\mathbf{v}}_i^{n+1} = \Delta t \sum_i w_{ip}^n \left(\mathbf{v}_i^n + \frac{\Delta t}{m_i^n} \mathbf{f}_i \right).$$

Thus, if we scale the time step to be $s\Delta t$, where $0 < s \leq 1$, the particle displacement is

$$\begin{aligned} \mathbf{x}_p^{n+1} - \mathbf{x}_p^n &= s\Delta t \sum_i w_{ip}^n \left(\mathbf{v}_i^n + \frac{s\Delta t}{m_i^n} \mathbf{f}_i \right) = s\Delta t \sum_i w_{ip}^n \mathbf{v}_i^n + s^2 \Delta t \sum_i w_{ip}^n \frac{\Delta t}{m_i^n} \mathbf{f}_i \\ &= s\Delta t \sum_i w_{ip}^n \mathbf{v}_i^n + s^2 \Delta t \sum_i w_{ip}^n (\tilde{\mathbf{v}}_i^{n+1} - \mathbf{v}_p^n). \end{aligned}$$

We choose s such that $\forall p$, $\|\mathbf{x}_p^{n+1} - \mathbf{x}_p^n\|_\infty \leq \nu_x \Delta x$, where $0 < \nu_x \leq 1$. Note that this limits the *actual* displacement of each particle to be less than a grid cell; the velocity-based time step restriction merely made a prediction to get an initial time step size. Computing s amounts to solving a quadratic equation for each component. As optimization, we keep a running best s as we go (initially $s = 1$) and first test to see whether the previous s satisfies the current constraint. This test saves us most of the square roots that would otherwise be required.

1.3 Deformation gradient

We impose a time step restriction to limit the change in the deformation gradient over the time step. The velocity gradient is transferred from the grid to particle p as

$$\nabla \mathbf{v}_p = \sum_i \tilde{\mathbf{v}}_i^{n+1} (\nabla w_{ip}^n)^T,$$

and the deformation gradient on the particle is updated as

$$\mathbf{F}_p^{n+1} = (\mathbf{I} + \Delta t \nabla \mathbf{v}_p) \mathbf{F}_p^n.$$

Scaling the time step to be $s\Delta t$, where $0 < s \leq 1$, gives the deformation gradient update

$$\mathbf{F}_p^{n+1} = (\mathbf{I} + s\Delta t \nabla \mathbf{v}_{p,a} + s^2 \Delta t^2 \nabla \mathbf{v}_{p,b}) \mathbf{F}_p^n,$$

where

$$\nabla \mathbf{v}_{p,a} = \sum_i \mathbf{v}_i^n (\nabla w_{ip}^n)^T,$$
$$\nabla \mathbf{v}_{p,b} = \sum_i (\tilde{\mathbf{v}}_i^{n+1} - \mathbf{v}_i^n)(\nabla w_{ip}^n)^T.$$

Here, we choose s so that $\|s\Delta t \nabla \mathbf{v}_{p,a} + s^2 \Delta t^2 \nabla \mathbf{v}_{p,b}\|_{\max} \leq \delta$, which again requires solving a quadratic equation componentwise. As with the position update, testing the previous s avoids most of the square roots. Note that the role of δ here is to limit the fraction of change in the deformation gradient. We use $\delta = 0.2$ for our examples.

2 Sound speed

The speed of waves in media play a critical role in many disciplines. The sound speed plays a fundamental role in compressible flows, especially in the study of explosions and shocks. Indeed, calculating the speed of sound in compressible flow is a fairly routine exercise. Outside fluid dynamics, we are only aware of results for isotropic linear elasticity at the rest configuration [3, 6]. Although limited in scope, these solutions provide a useful estimate for the speed of pressure waves (P-waves) and shear waves (S-waves) and are used for understanding earthquakes. Curiously, we are unaware of any more general treatment of the topic. Very similar treatments exist for Riemann solvers [2], where the wave speeds along fixed directions are required.

Only the fastest wave speed is needed for properly computing a CFL condition for explicit integrators. Thus, we are not interested in computing all of the wave speeds for a given place in a material (though we show at least in principle how to do so).

In this section, we will adapt ideas from the study of compressible flows to study the wave speeds for fully nonlinear hyperelastic constitutive models in configurations far from the rest state. This leads to an eigenvalue-like problem, which we do not know how to solve practically. We then show that, for isotropic (but still nonlinear) constitutive models, we can compute many of the solutions very efficiently. We also describe briefly the solutions to the mathematical problem. In practice, the fastest wave speed is virtually always one of the solutions we are able to compute efficiently. The resulting method provides a very efficient CFL estimate for SVD-based isotropic hyperelastic constitutive models. We show that our solution agrees with the known theoretical results at the rest configuration.

2.1 Wave speed for an elastic solid in 1D

For simplicity, we first solve the problem in 1D. We do this mostly for expositional reasons, as the notation is much simpler in 1D and the solution generally easier to follow. The 1D problem also looks more similar to the compressible flow case.

We begin be writing the equations of motion in Eulerian form. We have the usual equations for conservation of mass and momentum (but with general stress σ instead of pressure). We also have an Eulerian form of the MPM update rule for the deformation gradient F.

$$\begin{aligned} \rho_t + (\rho u)_x &= 0 \\ (\rho u)_t + (\rho u^2 - \sigma)_x &= 0 \\ F_t + uF_x - u_x F &= 0 \end{aligned}$$

We note that this formulation is quite different from the formulation followed in [2], which uses ρF instead of F as a primary variable, uses conservation of energy, and omits conservation of mass. Writing this in

terms of primary variables ρ, $\phi = \rho u$, and F gives

$$\begin{aligned} \rho_t + \phi_x &= 0 \\ \phi_t + (\rho^{-1}\phi^2 - \sigma(F))_x &= 0 \\ F_t + (\rho^{-1}\phi)F_x - (\rho^{-1}\phi)_x F &= 0 \end{aligned}$$

Expanding the derivatives to be on primary variables gives

$$\begin{aligned} \rho_t + \phi_x &= 0 \\ \phi_t - \rho^{-2}\phi^2\rho_x + 2\rho^{-1}\phi\phi_x - \sigma_F(F)F_x &= 0 \\ F_t + \rho^{-1}\phi F_x + \rho^{-2}\phi\rho_x F - \rho^{-1}\phi_x F &= 0 \end{aligned}$$

where $\sigma_F(F)$ is the derivative of stress with respect to the deformation gradient. In matrix form this is

$$U_t + GU_x = 0,$$

where

$$U = \begin{pmatrix} \rho \\ \phi \\ F \end{pmatrix}, \quad G = \begin{pmatrix} 0 & 1 & 0 \\ -\rho^{-2}\phi^2 & 2\rho^{-1}\phi & -\sigma_F(F) \\ \rho^{-2}\phi F & -\rho^{-1}F & \rho^{-1}\phi \end{pmatrix}$$

The scalar advection equation $w_t + cw_x = 0$ describes a quantity w that moves with velocity c. Similarly, if the matrix G is diagonalizable, the linearized system can be decoupled into three scalar advection equations as follows. Let G be diagonalizable, so that $G = P\Lambda P^{-1}$ for some diagonal matrix Λ. Substituting this factorization into the system above and multiplying by P^{-1} on the left, we get

$$P^{-1}U_t + \Lambda P^{-1}U_x = 0.$$

By linearizing about the current state, G and its factorization are treated as constant, and the system can be written as

$$(P^{-1}U)_t + \Lambda(P^{-1}U)_x = 0.$$

Since Λ is diagonal, the system has decoupled into three scalar equations for the three characteristic variables $P^{-1}U$. Each characteristic variable moves with a velocity equal to the corresponding diagonal entry of Λ. Curiously, for the matrix G given above, $G = P\Lambda P^{-1}$ with factors P and Λ that take the very simple form

$$P = \begin{pmatrix} 1 & 1 & 1 \\ u & u+c & u-c \\ 0 & -\frac{F}{\rho} & -\frac{F}{\rho} \end{pmatrix}, \quad \Lambda = \begin{pmatrix} u & & \\ & u+c & \\ & & u-c \end{pmatrix}, \quad c = \sqrt{\frac{\sigma_F F}{\rho}}.$$

Thus, we have found our sound speed c. In general, we are after the wave speeds and so would use $|u| + c$ to compute our CFL.

2.2 2D and 3D

With a general idea of how the derivation will proceed and what we are looking for, it is time to turn our attention to 2D and 3D. Let d denote the number of spatial dimensions. We adopt index tensor notation with summation convention and comma notation for partial derivatives, with a special adaptation for matrices. We use the notation $u_{i,r}$ to denote $\frac{\partial u_i}{\partial x_r}$ and $\sigma_{ir,(km)}$ to denote $\frac{\partial \sigma_{ir}}{\partial F_{km}}$. The parentheses distinguish the notation for the Hessian from the partial derivative with respect to the matrix F. The equations of motion are:

$$\begin{aligned} \rho_t + (\rho u_r)_{,r} &= 0 \\ (\rho u_i)_t + (\rho u_i u_r - \sigma_{ir})_{,r} &= 0 \\ (F_t)_{ij} + F_{ij,r}u_r - u_{i,r}F_{rj} &= 0 \end{aligned}$$

Writing this in terms of the primary variables ρ, $\phi_i = \rho u_i$, and F_{ij} gives

$$\begin{aligned}\rho_t + \phi_{r,r} &= 0\\ (\phi_i)_t + (\rho^{-1}\phi_i\phi_r - \sigma_{ir})_{,r} &= 0\\ (F_{ij})_t + (\rho^{-1}\phi_r)F_{ij,r} - (\rho^{-1}\phi_i)_{,r}F_{rj} &= 0\end{aligned}$$

Expanding the derivatives to be on primary variables gives

$$\begin{aligned}\rho_t + \phi_{r,r} &= 0\\ (\phi_i)_t - \rho^{-2}\phi_i\phi_r\rho_{,r} + \rho^{-1}\phi_{i,r}\phi_r + \rho^{-1}\phi_i\phi_{r,r} - \sigma_{ir,(km)}F_{km,r} &= 0\\ (F_{ij})_t + \rho^{-1}\phi_r F_{ij,r} - \rho^{-1}\phi_{i,r}F_{rj} + \rho^{-2}\phi_i F_{rj}\rho_{,r} &= 0,\end{aligned}$$

where $\sigma_{ir,r} = \sigma_{ir,(km)}F_{km,r}$ was used to obtain the second equation. Writing the equations in "matrix form", we get

$$\begin{pmatrix}\rho_t\\ (\phi_i)_t\\ (F_{ij})_t\end{pmatrix} + \begin{pmatrix}0 & \delta_{kr} & 0\\ -\rho^{-2}\phi_i\phi_r & \rho^{-1}\delta_{ik}\phi_r + \rho^{-1}\phi_i\delta_{kr} & -\sigma_{ir,(km)}\\ \rho^{-2}\phi_i F_{rj} & -\rho^{-1}\delta_{ik}F_{rj} & \rho^{-1}\phi_r\delta_{ik}\delta_{jm}\end{pmatrix}\begin{pmatrix}\rho_{,r}\\ \phi_{k,r}\\ F_{km,r}\end{pmatrix} = 0. \tag{1}$$

Denoting the matrix in Eq. (1) as G_r, we next consider G_r along a direction n_r, defining $N = G_r n_r$. Along this direction, the matrix becomes

$$N = \begin{pmatrix}0 & \delta_{kr}n_r & 0\\ -\rho^{-2}\phi_i\phi_r n_r & \rho^{-1}\delta_{ik}\phi_r n_r + \rho^{-1}\phi_i\delta_{kr}n_r & -\sigma_{ir,(km)}n_r\\ \rho^{-2}\phi_i F_{rj}n_r & -\rho^{-1}\delta_{ik}F_{rj}n_r & \rho^{-1}\phi_r\delta_{ik}\delta_{jm}n_r\end{pmatrix}.$$

Here we have used a bit of notation abuse, treating both ij and km as single indices taking on the values $1, \ldots, d^2$, which allows us to write N as a $(d^2 + d + 1)$-dimensional system. Substituting $\phi = \rho u$, we get

$$N = \begin{pmatrix}0 & n_k & 0\\ -u_i u_r n_r & \delta_{ik}u_r n_r + u_i n_k & -\sigma_{ir,(km)}n_r\\ \rho^{-1}u_i F_{rj}n_r & -\rho^{-1}\delta_{ik}F_{rj}n_r & u_r\delta_{ik}\delta_{jm}n_r\end{pmatrix}.$$

We make a general (but educated) guess at the form of an eigenvector based on the form of eigenvectors in the 1D case to get

$$\begin{pmatrix}0 & n_k & 0\\ -u_i u_r n_r & \delta_{ik}u_r n_r + u_i n_k & -\sigma_{ir,(km)}n_r\\ \rho^{-1}u_i F_{rj}n_r & -\rho^{-1}\delta_{ik}F_{rj}n_r & u_r\delta_{ik}\delta_{jm}n_r\end{pmatrix}\begin{pmatrix}1\\ u_k + v_k\\ M_{km}\end{pmatrix} - (n_r u_r + c)\begin{pmatrix}1\\ u_i + v_i\\ M_{ij}\end{pmatrix} = 0. \tag{2}$$

Since c, v_i, and M_{ij} are arbitrary, this is a general form for the eigenvalue, except for the assumption that the first component is nonzero, which we will consider later. This gives us three equations. Simplifying the first equation provides the definition for c,

$$c = n_r v_r. \tag{3}$$

The second and third equations relate M_{km} and v_i. Simplifying the second equation, we get

$$\begin{aligned}0 &= -u_i u_r n_r + \delta_{ik}u_r n_r(u_k + v_k) + u_i n_k(u_k + v_k) - \sigma_{ir,(km)}n_r M_{km} - n_r u_r(u_i + v_i) - c(u_i + v_i)\\ &= u_i n_r v_r - \sigma_{ir,(km)}n_r M_{km} - c(u_i + v_i)\\ &= -\sigma_{ir,(km)}n_r M_{km} - cv_i,\end{aligned}$$

where the last equality was obtained using Eq. (3). The second equation therefore gives

$$cv_i = -\sigma_{ir,(km)}n_r M_{km}. \tag{4}$$

Simplifying the third equation gives

$$\begin{aligned} 0 &= \rho^{-1} u_i F_{rj} n_r - \rho^{-1} \delta_{ik} F_{rj} n_r (u_k + v_k) + u_r \delta_{ik} \delta_{jm} n_r M_{km} - n_r u_r M_{ij} - c M_{ij} \\ &= -\rho^{-1} F_{rj} n_r v_i + u_r n_r M_{ij} - n_r u_r M_{ij} - c M_{ij}, \end{aligned}$$

so that

$$c M_{ij} = -\rho^{-1} F_{rj} n_r v_i. \tag{5}$$

Combining the results of Equations (4) and (5) gives

$$\begin{aligned} c^2 v_i &= -\sigma_{ir,(km)} n_r c M_{km} \\ &= \rho^{-1} \sigma_{ir,(km)} n_r F_{sm} n_s v_k \\ 0 &= (\sigma_{ir,(km)} n_r F_{sm} n_s - \rho c^2 \delta_{ik}) v_k. \end{aligned}$$

This gives us another eigenvalue problem,

$$(A_{ik} - \rho c^2 \delta_{ik}) v_k = 0, \tag{6}$$

where

$$A_{ik} = \sigma_{ir,(km)} n_r F_{sm} n_s.$$

There are two cases to consider in solving Eq. (6). In the first case, $v_k = 0$, and Eqs. (3) and (4) imply $c = 0$ and $n_r \sigma_{ir,(km)} M_{km} = 0$, respectively. The latter holds for any d^2-dimensional vector M_{km} which is in the nullspace of the $(d \times d^2)$-dimensional matrix $n_r \sigma_{ir,(km)}$, where d is number of spatial dimensions. Since $c = 0$, the associated local characteristic variables move at the bulk material velocity.

The second case is that v_k is an eigenvector of A_{ij}. In this case, let $\hat{\kappa} = \rho c^2$ be the associated eigenvalue, so that $c = \pm\sqrt{\frac{\hat{\kappa}}{\rho}}$. Since v_k is only known up to scale at this point, let $v_k = \alpha w_k$ with w_k a unit vector. Substituting this expression for v_k into Eq. (3), we obtain

$$c = \alpha n_r w_r,$$

which gives

$$\alpha = \frac{c}{n_r w_r} \tag{7}$$

Note that this assumes that $n_r w_r \neq 0$. Finally, M_{ij} can determined from Eqs. (5) and (7) to be

$$M_{ij} = -\frac{1}{\rho n_s w_s} F_{rj} n_r w_i. \tag{8}$$

Thus we see that every distinct eigenvector direction w_k for the system (6) such that $n_k w_k \neq 0$ leads to a unique eigenvector for the system (2).

The case $n_r w_r = 0$ must now be considered. By Eq. (3), this implies $c = 0$. Plugging this into Eq. (5), and using the fact that $v_i \neq 0$, we get

$$F_{rj} n_r = 0.$$

Thus, we see that in the case $n_r w_r = 0$, our deformation gradient must be singular. As this is nonphysical, we will not pursue this case further.

We now consider the case where the first component of the eigenvector of (2) is zero, giving the eigenvalue problem

$$\begin{pmatrix} 0 & n_k & 0 \\ -u_i u_r n_r & \delta_{ik} u_r n_r + u_i n_k & -\sigma_{ir,(km)} n_r \\ \rho^{-1} u_i F_{rj} n_r & -\rho^{-1} \delta_{ik} F_{rj} n_r & u_r \delta_{ik} \delta_{jm} n_r \end{pmatrix} \begin{pmatrix} 0 \\ u_k + v_k \\ M_{km} \end{pmatrix} - (n_r u_r + c) \begin{pmatrix} 0 \\ u_i + v_i \\ M_{ij} \end{pmatrix} = 0.$$

These equations simplify into

$$\begin{aligned} n_k(u_k + v_k) &= 0, \\ \sigma_{ir,(km)} n_r M_{km} &= -c(u_i + v_i), \\ cM_{ij} &= -\rho^{-1} F_{rj} n_r (u_i + v_i). \end{aligned}$$

Comparing these equations with the ones before, we see that this leads to the eigenvalue problem

$$(A_{ik} - \rho c^2 \delta_{ik})(u_k + v_k) = 0,$$

which will result in the same values of c as in Eq. (6).

2.3 Transforming to first Piola-Kirchhoff stress

The key to obtaining the characteristic speeds then reduces to one of finding the eigenvalues of A_{ik} in Eq. (6) for any choice of unit direction n_i . To derive a practical algorithm for bounding these eigenvalues, we first convert to the first Piola-Kirchhoff stress tensor P. The Cauchy stress is related to the first Piola-Kirchhoff stress as

$$\sigma_{ir} = \frac{1}{J} P_{in} F_{rn}.$$

Differentiating the Cauchy stress, we get

$$\begin{aligned} \sigma_{ir,(km)} &= -\frac{1}{J^2} P_{in} F_{rn} J_{,(km)} + \frac{1}{J} P_{in,(km)} F_{rn} + \frac{1}{J} P_{in} F_{rn,(km)} \\ &= -\frac{1}{J^2} P_{in} F_{rn} J (F^{-1})_{mk} + \frac{1}{J} P_{in,(km)} F_{rn} + \frac{1}{J} P_{in} \delta_{rk} \delta_{nm} \\ &= -\frac{1}{J} P_{in} F_{rn} (F^{-1})_{mk} + \frac{1}{J} P_{in,(km)} F_{rn} + \frac{1}{J} P_{im} \delta_{rk}. \end{aligned}$$

Multiplying by the deformation gradient,

$$\begin{aligned} \sigma_{ir,(km)} F_{sm} &= -\frac{1}{J} P_{in} F_{rn} (F^{-1})_{mk} F_{sm} + \frac{1}{J} P_{in,(km)} F_{rn} F_{sm} + \frac{1}{J} P_{im} \delta_{rk} F_{sm} \\ &= -\frac{1}{J} P_{in} F_{rn} \delta_{ks} + \frac{1}{J} P_{in,(km)} F_{rn} F_{sm} + \frac{1}{J} P_{im} \delta_{rk} F_{sm}. \end{aligned}$$

Using these results, A_{ik} can be expressed in terms of P as

$$\begin{aligned} A_{ik} = \sigma_{ir,(km)} F_{sm} n_r n_s &= -\frac{1}{J} P_{in} F_{rn} \delta_{ks} n_r n_s + \frac{1}{J} P_{in,(km)} F_{rn} F_{sm} n_r n_s + \frac{1}{J} P_{im} \delta_{rk} F_{sm} n_r n_s \\ &= -\frac{1}{J} P_{in} F_{rn} n_r n_k + \frac{1}{J} P_{in,(km)} F_{rn} F_{sm} n_r n_s + \frac{1}{J} P_{im} F_{sm} n_k n_s \\ &= -\frac{1}{J} P_{im} F_{sm} n_k n_s + \frac{1}{J} P_{in,(km)} F_{rn} F_{sm} n_r n_s + \frac{1}{J} P_{im} F_{sm} n_k n_s \\ &= \frac{1}{J} P_{in,(km)} F_{rn} F_{sm} n_r n_s. \end{aligned}$$

Expressed in this form, it is evident that A_{ik} is symmetric, since $P_{in,(km)}$ is the Hessian of the energy density. We are thus assured that the eigenvalues of A_{ik} will be real numbers.

Note that the problem that needs to be solved is not actually a simple $d \times d$ eigenvalue problem for A, since n_r also needs to be determined. Instead, the quantity that we need is

$$\kappa = \max_{\substack{u,n \\ \|u\|=\|n\|=1}} A_{ik} u_i u_k = \max_{\substack{u,n \\ \|u\|=\|n\|=1}} \frac{1}{J} P_{in,(km)} F_{rn} F_{sm} n_r n_s u_i u_k. \tag{9}$$

In general, we do not know how to solve problems of this type efficiently. The associated $d^2 \times d^2$ eigenvalue problem for $\frac{1}{J}P_{in,(km)}F_{rn}F_{sm}$ gives only an upper bound, since the domain of our optimization problem is more restrictive. For example, in the 3D isotropic case at the rest configuration, the bound obtained by solving the associated 9×9 eigenvalue problem is $6\mu + 9\lambda$, which is more than three times larger than the actual maximum $\kappa = 2\mu + \lambda$ obtained from the solution of Eq. (9).

2.4 Isotropy

To make further progress, we assume isotropy. In practice, this covers the significant majority of elasticity models in use in graphics and MPM. In the isotropic case, the derivative of P has a sparse representation in diagonal space [7, 12, 11], which we will exploit to find solutions of Eq. (9).

We first transform into a diagonalized representation using the singular value decomposition of the deformation gradient,

$$F_{ij} = U_{ik}\Sigma_{kn}V_{jn}.$$

Let $\sigma_i = \Sigma_{ii}$ be the singular values, and $\hat{\psi}(\sigma_i)$ be the energy density as a function of the singular values. The first Piola-Kirchhoff stress tensor diagonalizes in this space as $P_{ij} = U_{ik}\hat{P}_{kn}V_{jn}$, where $\hat{P}_{kn}$ is a diagonal matrix with $P_{ii} = \hat{\psi}_{,i} = \frac{\partial\hat{\psi}}{\partial\sigma_i}$. Differentiating P, we get

$$P_{in,(km)} = U_{ir}V_{ns}U_{kt}V_{mu}M_{rstu},$$

where M_{rstu} is a sparse tensor whose entries are described below [11]. Substituting this expression into our objective in Eq. (9), we have

$$\begin{aligned}
A_{ik}u_iu_k &= \frac{1}{J}P_{in,(km)}F_{rn}F_{sm}n_rn_su_iu_k \\
&= \frac{1}{J}(U_{ir}V_{ns}U_{kt}V_{mu}M_{rstu})(U_{jo}\Sigma_{op}V_{np})(U_{lv}\Sigma_{vw}V_{mw})n_jn_lu_iu_k \\
&= \frac{1}{J}U_{ir}U_{jo}U_{kt}U_{lv}M_{rstu}\Sigma_{os}\Sigma_{vu}u_in_ju_kn_l \\
&= \frac{1}{J}M_{rstu}\Sigma_{os}\Sigma_{vu}p_rq_op_tq_v,
\end{aligned}$$

where $p_r = U_{ir}u_i$ and $q_o = U_{jo}n_j$ are also unit vectors. Although this is essentially the same form we started with, we have replaced dense matrices and tensors with diagonal matrices and sparse tensors. In particular, $\frac{1}{J}M_{rstu}\Sigma_{os}\Sigma_{vu}$ has the same sparsity pattern as M_{rstu}, though more of its elements are distinct.

Sparsity. The sparsity pattern of M_{ijkl} can be summarized as follows [11]. Summation is not implied, and entries not explicitly given are zero.

$$M_{iikk} = \hat{\psi}_{,ik}, \qquad M_{ikik} = \frac{a_{ik} + b_{ik}}{2}, i \neq k, \qquad M_{ikki} = \frac{a_{ik} - b_{ik}}{2}, i \neq k,$$

where

$$a_{ik} = \frac{\hat{\psi}_{,i} - \hat{\psi}_{,k}}{\sigma_i - \sigma_k}, \qquad b_{ik} = \frac{\hat{\psi}_{,i} + \hat{\psi}_{,k}}{\sigma_i + \sigma_k}.$$

Observe that $M_{iikk} = M_{kkii}$, $M_{ikik} = M_{kiki}$, $M_{ikki} = M_{kiik}$, and generally $M_{ijkl} = M_{klij}$. With these properties, solving the 9×9 eigenvalue problem reduces to solving three 2×2 eigenvalue problems and one 3×3 eigenvalue problem, which is potentially tractable. However, as noted above, the bound obtained is not very tight, so we do not pursue this strategy.

System of equations. We note that, in this sparse form, some solutions are trivial. Using Lagrange multipliers, our constrained optimization problem in (9) becomes finding critical points of

$$L = \frac{1}{J} M_{rstu} \Sigma_{os} \Sigma_{vu} p_r q_o p_t q_v - \lambda(p_r p_r - 1) - \mu(q_r q_r - 1)$$

over p, q, λ, and μ. Differentiating L with respect to these variables and setting the result to 0, we get

$$0 = \frac{\partial L}{\partial p_i} = \frac{1}{J} M_{istu} \Sigma_{os} \Sigma_{vu} q_o p_t q_v + \frac{1}{J} M_{rsiu} \Sigma_{os} \Sigma_{vu} p_r q_o q_v - 2\lambda p_i,$$
$$0 = \frac{\partial L}{\partial q_i} = \frac{1}{J} M_{rstu} \Sigma_{is} \Sigma_{vu} p_r p_t q_v + \frac{1}{J} M_{rstu} \Sigma_{os} \Sigma_{iu} p_r q_o p_t - 2\mu q_i,$$

along with the unit norm constraints on p and q, so that the critical points satisfy

$$\lambda p_i = \frac{1}{J} M_{istu} \Sigma_{os} \Sigma_{vu} q_o p_t q_v, \tag{10}$$
$$\mu q_i = \frac{1}{J} M_{rstu} \Sigma_{is} \Sigma_{vu} p_r p_t q_v, \tag{11}$$
$$p_r p_r = 1, \tag{12}$$
$$q_r q_r = 1. \tag{13}$$

Multiplying Eq. (10) by p_i and using Eq. (12), and combining Eqs. (11) and (13) analogously, we find

$$\lambda = \mu = \hat{\kappa} = \frac{1}{J} M_{rstu} \Sigma_{os} \Sigma_{vu} p_r q_o p_t q_v.$$

We can summarize our system of equations as

$$C_{ro} = \frac{1}{J} M_{rstu} \Sigma_{os} \Sigma_{vu} p_t q_v, \quad \lambda p_i = C_{io} q_o, \quad \mu q_i = C_{ri} p_r, \quad \lambda = \mu = \hat{\kappa} = C_{ro} p_r q_o, \quad p_r p_r = 1, \quad q_r q_r = 1. \tag{14}$$

Trivial solutions. The system (14) has trivial solutions when p_r and q_o are axis vectors, which we denote as $p_r = \delta_{ra}$ and $q_o = \delta_{ob}$ for any choice a, b. If $a = b$, then

$$C_{ro} = \frac{1}{J} M_{rstu} \Sigma_{os} \Sigma_{vu} p_t q_v = \frac{1}{J} M_{rstu} \Sigma_{os} \Sigma_{vu} \delta_{ta} \delta_{va} = \frac{1}{J} M_{rsaa} \Sigma_{os} \Sigma_{aa} \qquad \text{(no summation on } a\text{)},$$

where we have used $u = v$ (Σ_{vu} is diagonal). Due to the sparsity pattern of M and the diagonal structure of Σ_{os}, we can see that $C_{ro} = 0$ for $r \neq o$. Thus, C_{ro} is diagonal with entries

$$C_{cc} = \frac{1}{J} M_{ccaa} \Sigma_{cc} \Sigma_{aa} \qquad \text{(no summation on } a, c\text{)}.$$

Plugging these into our equations (14) shows that they are all satisfied, and $\hat{\kappa} = \frac{1}{J} M_{aaaa} \Sigma_{aa} \Sigma_{aa}$ is our corresponding candidate. Otherwise, if $a \neq b$, then

$$C_{ro} = \frac{1}{J} M_{rstu} \Sigma_{os} \Sigma_{vu} p_t q_v = \frac{1}{J} M_{rstu} \Sigma_{os} \Sigma_{vu} \delta_{ta} \delta_{vb} = \frac{1}{J} M_{rsau} \Sigma_{os} \Sigma_{bu} = \frac{1}{J} M_{rsab} \Sigma_{os} \Sigma_{bb} \quad \text{(no summation on } a \text{ or } b\text{)}.$$

In this case, only C_{ab} and C_{ba} are nonzero, and are given by

$$C_{ab} = \frac{1}{J} M_{abab} \Sigma_{bb} \Sigma_{bb}, \qquad C_{ba} = \frac{1}{J} M_{baab} \Sigma_{aa} \Sigma_{bb}$$

The equations (14) are again satisfied, and $\hat{\kappa} = C_{ab} = \frac{1}{J} M_{abab} \Sigma_{bb} \Sigma_{bb}$.

Sound speed CFL condition. With these trivial solutions, our sound-speed-based CFL computation for MPM can be summarized as

$$\kappa = \frac{1}{J}\max_{a,b} M_{abab}\Sigma_{bb}\Sigma_{bb}, \qquad c = \sqrt{\frac{\kappa}{\rho}}, \qquad \Delta t = \frac{\Delta x}{c}. \tag{15}$$

The smallest time step computed for any particle (scaled by a factor less than one, typically 0.9) is one of the time step restrictions we use to limit our time step sizes. Observe that $J\rho = \rho_0$ is the initial particle density and does not change with time. Note also that we use the sound speed c directly, rather than the eigenvalues of N given by $n_r u_r + c$, as we do not need to account for the advective component $n_r u_r$, which is treated in a Lagrangian manner in MPM.

In practice, we've found these trivial solutions to be sufficient in computing our sound speed CFL. Our investigation of the full problem indicates that maximum eigenvalues occur at the trivial solutions in the vast majority of cases, as we discuss further below.

Mathematical form of full solution in 2D. The diagonal space problem in 2D is analytically solvable as the solution of a quadratic polynomial. For completeness, the solution can be computed using

$$a = (M_{0110} + M_{1100})\sigma_0\sigma_1, \quad b = M_{0000}\sigma_0^2 - M_{0101}\sigma_1^2, \quad d = (M_{0000} - M_{0101})\sigma_0^2, \quad e = (M_{0101} - M_{1111})\sigma_1^2,$$

$$f = de + a^2, \qquad g = be + a^2, \qquad h = bd - a^2, \qquad k = f - e(b - e), \qquad m = f + d(b - d),$$

$$-fhx^4 + 2(fh - fg - gh)x^2 + km = 0, \quad y = \frac{gx^2 - k}{a(d + e)x}, \quad c = M_{0000}\sigma_0^2 + \frac{2ayx - bx^2 - dy^2 - b - e}{(x^2 + 1)(y^2 + 1)}.$$

Here, x^2 is obtained by solving a quadratic equation. We may obtain 0, 2, or 4 solutions x. Each solution leads to a wave speed c. Since x and $-x$ produce the same wave speed c, there are only two candidates to check. Even though c can be computed from x^2 alone (x is not actually required), the wave speed is not meaningful unless $x^2 \geq 0$. We used Gröbner bases to solve the optimization problem analytically and to show that all of the solutions are obtained either as trivial solutions or by solving the quadratic above.

We implemented this analytic solution along with the trivial solutions during our testing and compared them with approximate numerical solutions determined by brute force to ensure that we were finding all possible solutions. We observed that for random inputs, the quadratic solution occurred occasionally. However, for inputs generated from physical simulation, the quadratic solution occurred only three times out of approximately a million sound speed computations. Given the added expense of the quadratic solution and the extreme improbability of encountering it in practice, we removed the quadratic solution from our code and use only the trivial solutions, which are simple to compute and provide the correct solution nearly always.

Mathematical form of full solution in 3D. The situation in 3D is significantly more complicated. The 3D problem can be restricted to 2D by fixing a component to zero. (That is, $u_a = n_a = 0$ for some a.) This results in the trivial solutions and the quadratic solutions.

Extensive Gröbner basis analysis reveals that, in addition to these 2D solutions, there seems to exist a fully general type of solution possible in 3D. This solution requires the solution to a degree-18 polynomial. The mathematical complexity of the system of polynomial equations in 3D has defied all of our attempts at a full analytic solution, and we have not succeeded in calculating the coefficients of the degree-18 polynomial. Rather, we are forced to infer its existence by substituting in integers for the inputs and solving the system of polynomials over finite fields.

In testing with random inputs, we never observed a maximum wave speed that could not be computed from a trivial solutions or a quadratic solution. As such, the fully general solutions are of little more than academic interest.

Agreement with isotropic linear elasticity. We show that our derivation agrees with existing results for isotropic linear elasticity. In this case, with Lamé parameters μ, λ, and strain ϵ, the stress is given by

$$\begin{aligned}\sigma &= 2\mu\epsilon + \lambda \mathrm{tr}(\epsilon) I, \\ &= \mu(F + F^T - 2I) + \lambda(\mathrm{tr}(F) - d)I,\end{aligned}$$

so that

$$\begin{aligned}\sigma_{ij} &= \mu(F_{ij} + F_{ji} - 2\delta_{ij}) + \lambda(F_{kk} - d)\delta_{ij}, \\ \sigma_{ij,(km)} &= \mu\delta_{ik}\delta_{jm} + \mu\delta_{jk}\delta_{im} + \lambda\delta_{ij}\delta_{km}.\end{aligned}$$

Substituting this into the matrix A_{ik}, we get

$$\begin{aligned}A_{ik} &= \sigma_{ij,(km)} n_j F_{sm} n_s \\ &= \mu\delta_{ik} n_j F_{sj} n_s + \mu n_k F_{si} n_s + \lambda n_i F_{sk} n_s,\end{aligned}$$

or in matrix and vector notation,

$$A = \mu(n^T F n) I + \mu F^T n n^T + \lambda n n^T F.$$

At the rest configuration, $F = I$, and $A = \mu I + \mu n n^T + \lambda n n^T$ has eigenvectors n and w, for any vector w orthogonal to n, so that

$$\begin{aligned}An &= (2\mu + \lambda) n, \\ Aw &= \mu w.\end{aligned}$$

This corresponds to a P-wave speed of $\sqrt{\frac{2\mu+\lambda}{\rho}}$ and an S-wave speed of $\sqrt{\frac{\mu}{\rho}}$, which agrees with published values [1]. These results can also obtained from the diagonal space results in Eq. (15) by using the corotated constitutive model at the rest configuration, where $\Sigma_{aa} = 1$, $J = 1$, $M_{aaaa} = 2\mu + \lambda$, and $M_{abab} = \mu$.

Agreement with compressible flow. We next show that our derivation agrees with existing results for compressible flow. The stress tensor for an inviscid fluid is given by

$$\sigma = -pI,$$

where $p = p(J)$ is pressure. The derivative of stress is

$$\sigma_{ij,(rs)} = -\delta_{ij} p_J(J) J (F^{-1})_{sr}.$$

Substituting $\sigma_{ij,(rs)}$ into A gives

$$A_{ir} = \sigma_{ij,(rs)} n_j F_{ks} n_k = -\delta_{ij} p_J(J) J (F^{-1})_{sr} n_j F_{ks} n_k = -\delta_{ij} p_J(J) J \delta_{kr} n_k n_j = -p_J(J) J n_i n_r,$$

or in matrix form $A = -p_J(J) J n n^T$. The maximum eigenvalue is $-p_J(J)J$. Therefore, the sound speed is $c = \sqrt{\frac{-p_J(J)J}{\rho}}$. This form of the sound speed may seem unfamiliar. This is because p is normally expressed as a function of density rather than deformation, so that its derivative with respect to J is

$$\frac{\partial}{\partial J} p(\rho) = \frac{\partial}{\partial J} p(\rho) = \frac{\partial}{\partial J} p\Big(\frac{\rho_0}{J}\Big) = \Big(-\frac{\rho_0}{J^2}\Big) p_\rho\Big(\frac{\rho_0}{J}\Big) = \Big(-\frac{\rho}{J}\Big) p_\rho(\rho).$$

Substituting this expression into the sound speed c, we get

$$c = \sqrt{\frac{-p_J(J)J}{\rho}} = \sqrt{\frac{\partial p}{\partial \rho}},$$

which is a familiar form for the sound speed in compressible flow.

3 Single particle stability

In an MPM simulation, neighboring particles have a stabilizing influence on particle evolution, and thus the ejection and isolation of a particle (for example, due to splashing water) may lead to simulation instability. While this observation is not new [8, 9], here we identify for the first time an instability caused by a feedback loop between $\mathbf{F}$ (or J_p for fluids) and forces. We show that this instability imposes restrictions on both the choice of basis function and the time step.

3.1 Fluids and pressures

Consider a fluid simulation, where the force is a pressure defined in terms of the deformation gradient through an energy density $\psi(J)$. Consider a particle p that becomes isolated from interactions with other particles on the grid. We assume PIC transfers for simplicity. During one time step of evolution, transferring data from the particle to the grid gives

$$\begin{aligned} m_i^n &= w_{ip}^n m_p, \\ m_i^n \mathbf{v}_i^n &= w_{ip}^n m_p \mathbf{v}_p^n, \\ \mathbf{v}_i^n &= \mathbf{v}_p^n, \end{aligned}$$

where the last equation results from the isolation of the particle. Explicitly integrating forces on the grid, we get

$$\begin{aligned} \mathbf{f}_i &= -V_p^0 J_p^n \psi'(J) \nabla w_{ip}^n, \\ \tilde{\mathbf{v}}_i^{n+1} &= \mathbf{v}_i^n + \frac{\Delta t}{m_i^n} \mathbf{f}_i, \end{aligned}$$

and the particle update then gives

$$\begin{aligned} \nabla \cdot \mathbf{v}_p &= \sum_i \tilde{\mathbf{v}}_i^{n+1} \cdot \nabla w_{ip}^n, \\ J_p^{n+1} &= (1 + \Delta t \nabla \cdot \mathbf{v}_p) J_p^n. \end{aligned}$$

Putting these together,

$$\begin{aligned} J_p^{n+1} &= \left(1 + \Delta t \sum_i \left(\mathbf{v}_p^n - \frac{\Delta t V_p^0 J_p^n \psi'}{w_{ip}^n m_p} \nabla w_{ip}^n\right) \cdot \nabla w_{ip}^n\right) J_p^n \\ &= \left(1 - \frac{\Delta t^2 V_p^0 J_p^n \psi'}{m_p} \sum_i \frac{\nabla w_{ip}^n \cdot \nabla w_{ip}^n}{w_{ip}^n}\right) J_p^n, \end{aligned}$$

where we have used $\sum_i \nabla w_{ip}^n = \mathbf{0}$. This contains (the trace of) the matrix

$$\mathbf{H} = \sum_i \frac{\nabla w_{ip}^n (\nabla w_{ip}^n)^T}{w_{ip}^n}.$$

This matrix contains two types of entries, which occur on and off the diagonal. We assume $w_{ip}^n = N(\mathbf{x}_i - \mathbf{x}_p)$ is a tensor product basis, so that in 3D $N(\mathbf{x}_{ijk}) = N(x_i)N(y_j)N(z_k)$. Note that off-diagonal entries of $\mathbf{H}$

are zero. For example,

$$\begin{aligned}\mathbf{H}_{12} &= \sum_{ijk} \frac{(N'(x_i)N(y_j)N(z_k))(N(x_i)N'(y_j)N(z_k))}{N(x_i)N(y_j)N(z_k)} \\ &= \sum_{ijk} N'(x_i)N'(y_j)N(z_k) \\ &= \left(\sum_i N'(x_i)\right)\left(\sum_j N'(y_j)\right)\left(\sum_k N(z_k)\right) \\ &= (0)(0)(1).\end{aligned}$$

The first diagonal entry is

$$\begin{aligned}\mathbf{H}_{11} &= \sum_{ijk} \frac{(N'(x_i)N(y_j)N(z_k))(N'(x_i)N(y_j)N(z_k))}{N(x_i)N(y_j)N(z_k)} \\ &= \sum_{ijk} \frac{N'(x_i)^2N(y_j)N(z_k)}{N(x_i)} \\ &= \left(\sum_i \frac{N'(x_i)^2}{N(x_i)}\right)\left(\sum_j N(y_j)\right)\left(\sum_k N(z_k)\right) \\ &= \sum_i \frac{N'(x_i)^2}{N(x_i)}.\end{aligned}$$

The other diagonal entries are similar. Observe that if $\mathbf{H}_{11}$ is unbounded, then $\mathrm{tr}(\mathbf{H})$ will also be unbounded, which means J_p^{n+1} will be unbounded. The resulting simulation will be unstable and may explode spontaneously if any particle becomes isolated. This boundedness is a property of the basis functions employed. This quantity is not bounded for linear splines. The interpolation must decay to zero quadratically or faster for stability. Note that the function $N(x)$ depends on the grid size. Let h be the grid size, and define $\hat{N}(\hat{x})$ by $N(x) = \hat{N}(x/h) = \hat{N}(\hat{x})$, where $\hat{N}(\hat{x})$ is not a function of grid size. Then

$$H_{11} = \sum_i \frac{N'(x_i)^2}{N(x_i)} = \frac{1}{h^2}\sum_i \frac{\hat{N}'(\hat{x}_i)^2}{\hat{N}(\hat{x}_i)}$$

Note that H_{11} should be a multiple of h^{-2}, which will generally depend on the location of a particle within a cell. We need it to be bounded. For quadratic splines,

$$\frac{4}{h^2} \le H_{11} \le \frac{6}{h^2}.$$

For cubic splines,

$$\frac{3}{h^2} \le H_{11} < \frac{3.14}{h^2}.$$

Thus, we can bound the matrix entry by $H_{11} \le \frac{K}{h^2}$ with $K = 6$ for quadratic splines and $K = 3.14$ for cubic splines. Note that the same bound applies to H_{22} and H_{33}. For our immediate purposes, $\mathrm{tr}(\mathbf{H}) \le \frac{Kd}{h^2}$ where $d = 2, 3$ is the dimension. The ratio of change is bounded by

$$J_p^{n+1} = \left(1 - \frac{\Delta t^2 V_p^0 K d J_p^n \psi'}{h^2 m_p}\right) J_p^n = (1 - r J_p^n \psi') J_p^n,$$

where

$$r = \frac{\Delta t^2 V_p^0 K d}{h^2 m_p}. \tag{16}$$

We can make r smaller by choosing a smaller time step; finding a suitable r leads to a time step restriction. We assume the pressure force is restorative, so that $J_p^n - 1$ and $\psi'(J_p^n)$ have the same sign. In this way, the value of J_p increases if it is less than 1 and decreases if it is larger than 1.

Simple bounds. For sufficiently small r, the solution should simply decay to $J \to 1$. We will choose r so that J does not overshoot 1. Reaching 1 exactly requires

$$(1 - r J_p^n \psi') J_p^n = 1.$$

Solving for r gives

$$r = \frac{J_p^n - 1}{(J_p^n)^2 \psi'}. \tag{17}$$

On the other hand, $r = 0$ leaves J_p^n unchanged. Values in between leave J_p^n in between these extremes, which is the desired behavior (decay without overshoot). Solving for Δt in Eq. (16), we get

$$\begin{aligned} \Delta t &= h \sqrt{\frac{m_p r}{V_p^0 K d}} \\ &= \Delta x \sqrt{\frac{\rho_p^0 r}{K d}}, \end{aligned}$$

where $h = \Delta x$. Substituting the value of r from Eq. (17) gives

$$\Delta t = \frac{\Delta x}{J_p^n} \sqrt{\frac{\rho_p^0 (J_p^n - 1)}{K \psi' d}},$$

where we have used $\rho_p^0 = m_p / V_p^0$, and V_p^0 is the initial volume of particle p. Note that Δt remains well-defined near the rest configuration since by L'Hôpital's rule,

$$\begin{aligned} \lim_{J_p^n \to 1} \Delta t &= \lim_{J_p^n \to 1} \frac{\Delta x}{J_p^n} \sqrt{\frac{\rho_p^0 (J_p^n - 1)}{K \psi' d}} \\ &= \Delta x \sqrt{\frac{\rho_p^0}{K \psi'' d}}. \end{aligned}$$

Less restrictive bounds. In practice, this value of Δt can be improved somewhat. To ease up the restriction on Δt, we must allow ourselves to overshoot $J = 1$, alternating between $J < 1$ and $J > 1$. Doing this safely requires us to have an idea of how our forces will behave at the other side of our overshoot. For our purposes, we assume that the force's growth is bounded by

$$0 \geq \psi'(J) \geq \lambda (J - 1) J^{-2}, \quad 0 < J \leq 1, \tag{18}$$

$$0 \leq \psi'(J) \leq \lambda (J - 1), \qquad J \geq 1, \tag{19}$$

for some $\lambda > 0$. This growth is fast enough to accommodate constitutive models $\psi(J)$ that contain common terms like $(J-1)^2$, $\ln J$, $\ln^2 J$, or J^{-1}. The value λ must be computed given knowledge of the constitutive model. The analysis that follows can be repeated with different bounds or an actual constitutive model if desired.

We select r in order to ensure that

$$J_2 = (1 - rJ\psi')J \tag{20}$$

is bounded in a way that prevents growth over time. We can do this be ensuring that $|\ln(J_2)| \leq |\ln(J)|$, which is equivalent to requiring J_2 to be between J and J^{-1}. Across many time steps, we will have $|\ln(J)| \geq |\ln(J_2)| \geq |\ln(J_3)| \geq |\ln(J_4)| \geq \cdots \geq 0 = |\ln(1)|$, which prevents divergence.

We select r in two separate cases. First, we consider the case $J \leq 1$, where we want to ensure $J \leq J_2 \leq J^{-1}$. From (18) and (20) we have the bounds $J_2 \geq J$ and $J_2 = (1 - rJ\psi')J \leq (1 - rJ\lambda(J-1)J^{-2})J$. Requiring

$$r \leq \frac{2}{\lambda(2-J)^2}, \qquad 0 < J \leq 1,$$

ensures $(1 - rJ\lambda(J-1)J^{-2})J \leq J^{-1}$ since

$$J^{-1} - (1 - rJ\lambda(J-1)J^{-2})J \geq J^{-1} - \left(1 - \frac{2}{\lambda(2-J)^2}J\lambda(J-1)J^{-2}\right)J = \frac{(5-(1-J)^2)(1-J)^2}{J(2-J)^2} \geq 0$$

when $0 < J \leq 1$.

For the case $J \geq 1$, we want $J^{-1} \leq J_2 \leq J$. From (19) and (20) we have the bounds $J_2 \leq J$ and $J_2 = (1 - rJ\psi')J \geq (1 - rJ\lambda(J-1))J$. The choice

$$r \leq \frac{J+1}{\lambda J^3}, \qquad J \geq 1$$

ensures $(1 - rJ\lambda(J-1))J \geq J^{-1}$ since

$$(1 - rJ\lambda(J-1))J - J^{-1} \geq \left(1 - \frac{J+1}{\lambda J^3}J\lambda(J-1)\right)J - J^{-1} = 0.$$

We can summarize this eased time step restriction as

$$\Delta t = \frac{\Delta x}{2-J}\sqrt{\frac{2\rho_p^0}{K\lambda d}}, \qquad 0 < J \leq 1,$$
$$\Delta t = \frac{\Delta x}{J}\sqrt{\frac{\rho_p^0(J+1)}{JK\lambda d}}, \qquad J \geq 1.$$

Near $J = 1$, this improves the time step size by a factor of $\sqrt{2}$ allowing us to take time steps that are 40% larger; we use this bound in all of our fluid tests. Performing the analysis directly on individual constitutive models can produce better bounds. Since $J \approx 1$ nearly all of the time for fluids, nearly all of the performance benefits are obtained by getting favorable time step bounds in the limit $J \to 1$.

4 Reflection boundary conditions

We enforce boundary conditions at domain walls through reflection [5]. The motivation behind these boundary conditions is what one sees when looking at a mirror. When one touches a mirror, the mirror exerts a force back on one's hand that is equal and opposite. However, looking in the mirror it appears as though there is a second hand that mirrors the movements of the real hand and pushes back on it with equal and opposite force. The observation is that the two physical situations behave alike. The force a hand experiences from the mirror is the same as the force it experiences from a reflected hand. Because of the hybrid nature of MPM, the interaction between two hands is automatic. This gives us a straightforward and efficient way to apply boundary conditions at domain walls. We extend this idea to handle both separation and friction.

Reflection notation. We will need to be able to denote reflections of various quantities. We denote reflections of positions, velocities, grid indices, and particles indices as $\hat{r}(\mathbf{x})$, $r(\mathbf{v})$, $r(i)$, and $r(p)$, respectively. We use a hat for the function that reflects positions to distinguish it from the very different function that reflects velocities. All reflections are involutions (they are their own inverse). Thus, for example, $r(r(p)) = p$ and $r(r(\mathbf{v}_p^n)) = \mathbf{v}_p^n$. We use p for particle indices; here $r(p)$ is the index of a (fictitious) new particle that is conceptually constructed as a reflected copy of the original particle p. The position of the reflected particle is just the reflected position $\mathbf{x}_{r(p)}^n = \hat{r}(\mathbf{x}_p) = \mathbf{R}\mathbf{x}_p + 2\mathbf{s}$, where $\mathbf{R}$ is a diagonal matrix with $\mathbf{R}_{aa} = \pm 1$, with -1 in the reflection direction and 1 otherwise, and $\mathbf{s}_a$ is nonzero only in the reflection direction. Note that $\mathbf{R}\mathbf{s} = -\mathbf{s}$. Note that $\mathbf{s}$ is on the plane of reflection, since $\hat{r}(\mathbf{s}) = \mathbf{s}$. Because we are always reflecting about grid domain boundaries, reflecting the locations of grid nodes (or grid cells, depending on how MPM was implemented) always produces the location of another grid node (or cell). Thus, we have $\mathbf{x}_{r(i)}^n = \hat{r}(\mathbf{x}_i^n)$. Reflected particles have the same mass as the original ($m_{r(p)} = m_p$). Reflection preserves weights, since

$$w_{r(i)r(p)}^n = N(\mathbf{x}_{r(i)}^n - \mathbf{x}_{r(p)}^n) = N(\mathbf{R}(\mathbf{x}_i^n - \mathbf{x}_p)) = N(\mathbf{x}_i^n - \mathbf{x}_p^n) = w_{ip}^n,$$

which follows from reflection symmetry of the interpolation kernel $N(\mathbf{x})$. Similarly, $w_{r(i)p}^n = w_{ir(p)}^n$.

Boundary conditions. We have a few useful choices for the way in which velocities are reflected, which we denote $\mathbf{v}_{r(p)}^n = r(\mathbf{v}_p^n) = \mathbf{A}\mathbf{v}_p^n + 2\mathbf{b}$. The involution property implies $\mathbf{A}\mathbf{b} = -\mathbf{b}$. The different options correspond to different types of boundary conditions. The choice $\mathbf{A} = \mathbf{I}, \mathbf{b} = \mathbf{0}$ behaves as a free surface boundary condition (in the computational fluid dynamics sense). The choice $\mathbf{A} = -\mathbf{I}$ enforces the no-slip boundary condition $\mathbf{v} = \mathbf{b}$. The other meaningful choice is $\mathbf{A} = \mathbf{R}$, which enforces a slip boundary condition. The normal velocity is enforced ($\mathbf{n} \cdot \mathbf{v} = \mathbf{n} \cdot \mathbf{b}$), but tangential velocity is unconstrained.

Transfers with reflection. In the absence of boundary conditions, the MPM transfers of mass and momentum from particles to grid are (assuming PIC transfers for simplicity)

$$m_i^n = \sum_p w_{ip}^n m_p, \qquad m_i^n \mathbf{v}_i^n = \sum_p w_{ip}^n m_p \mathbf{v}_p^n.$$

Next, lets consider a *modified* system where we have both our regular particles p as well as a reflected copy $r(p)$. The grid mass for our modified system is

$$\hat{m}_i^n = \sum_p w_{ip}^n m_p + \sum_p w_{ir(p)}^n m_{r(p)} = \sum_p w_{ip}^n m_p + \sum_p w_{r(i)p}^n m_p = m_i^n + m_{r(i)}^n.$$

That is, we simply add in the mass from the reflected grid node. Note that $\hat{m}_i^n = \hat{m}_{r(i)}^n$; the modified grid mass is symmetric with respect to the reflection that we used to generate it. The momentum for the modified scheme is

$$\begin{aligned}
\hat{m}_i^n \hat{\mathbf{v}}_i^n &= \sum_p w_{ip}^n m_p \mathbf{v}_p^n + \sum_p w_{ir(p)}^n m_{r(p)} \mathbf{v}_{r(p)}^n \\
&= m_i^n \mathbf{v}_i^n + \sum_p w_{ir(p)}^n m_{r(p)} (\mathbf{A}\mathbf{v}_p^n + 2\mathbf{b}) \\
&= m_i^n \mathbf{v}_i^n + \mathbf{A} \sum_p w_{r(i)p}^n m_p \mathbf{v}_p^n + 2\mathbf{b} \sum_p w_{r(i)p}^n m_p \\
&= m_i^n \mathbf{v}_i^n + \mathbf{A} m_{r(i)}^n \mathbf{v}_{r(i)}^n + 2\mathbf{b} m_{r(i)}^n \\
&= m_i^n \mathbf{v}_i^n + m_{r(i)}^n r(\mathbf{v}_{r(i)}^n).
\end{aligned}$$

Here, we simply add in the momentum from the reflected grid node, but with reflection. The modified mass $\hat{m}_i^n$ and momentum $\hat{m}_i^n \hat{\mathbf{v}}_i^n$ can be computed from m_i^n and $m_i^n \mathbf{v}_i^n$ as a simple postprocess. These corrections

only need to be applied near the boundary, since $m^n_{r(i)}$ will be nonzero only in a narrow ghost region around the interface. We never construct actual reflected particles. Noting

$$\begin{aligned}\hat{m}^n_i\hat{\mathbf{v}}^n_i - \mathbf{A}\hat{m}^n_{r(i)}\hat{\mathbf{v}}^n_{r(i)} &= (m^n_i\mathbf{v}^n_i + \mathbf{A}m^n_{r(i)}\mathbf{v}^n_{r(i)} + 2\mathbf{b}m^n_{r(i)}) - \mathbf{A}(m^n_{r(i)}\mathbf{v}^n_{r(i)} + \mathbf{A}m^n_i\mathbf{v}^n_i + 2\mathbf{b}m^n_i)\\ &= 2\mathbf{b}m^n_{r(i)} - 2\mathbf{A}\mathbf{b}m^n_i = 2\mathbf{b}m^n_{r(i)} + 2\mathbf{b}m^n_i = 2\mathbf{b}\hat{m}^n_i\end{aligned}$$

and dividing by $\hat{m}^n_i = \hat{m}^n_{r(i)}$, we have $\hat{\mathbf{v}}^n_i = \mathbf{A}\hat{\mathbf{v}}^n_{r(i)} + 2\mathbf{b} = r(\hat{\mathbf{v}}^n_{r(i)})$. As with mass, the velocity is invariant under the reflection used to generate it. In this way, we get valid masses and velocities inside the domain that respect our desired boundary conditions *and* valid extrapolated values in the ghost region as well.

Forces with reflection. In the absence of boundary conditions, the velocities at grid nodes are updated by applying forces giving

$$\mathbf{v}^{n+1}_i = \mathbf{v}^n_i + \Delta t(m^n_i)^{-1}\mathbf{f}^n_i.$$

In our modified system, we would receive forces $\mathbf{f}^n_i$ from our real particles and reflected forces $\mathbf{A}\mathbf{f}^n_{r(i)}$ from our (fictitious) reflected particles. The forces for our modified system are $\hat{\mathbf{f}}^n_i = \mathbf{f}^n_i + \mathbf{A}\mathbf{f}^n_{r(i)}$, so that

$$\hat{\mathbf{v}}^{n+1}_i = \hat{\mathbf{v}}^n_i + \Delta t(\hat{m}^n_i)^{-1}\hat{\mathbf{f}}^n_i.$$

Reflecting the forces and applying them with the modified mass preserves the symmetry of the velocity field: $\hat{\mathbf{v}}^{n+1}_i = \mathbf{A}\hat{\mathbf{v}}^{n+1}_{r(i)} + 2\mathbf{b} = r(\hat{\mathbf{v}}^{n+1}_{r(i)})$. Once again, the appropriate reflected force can be computed with a simple grid-based correction near the boundaries.

Corners. At the corners of the domain, one must enforce boundary conditions for more than one domain wall. This process is straightforward. Reflections should be performed in the ghost region (even when both i and $r(i)$ are in the ghost region). The domain walls may be reflected independently; the final result does not depend on the order.

Momentum conservation. For no-slip boundary conditions, the velocity at the interface is fixed to $\hat{\mathbf{v}}^n_i = \hat{\mathbf{v}}^{n+1}_i = \mathbf{b}$, which follows from the symmetry of the modified velocity field. For the free boundary condition, $\mathbf{A} = \mathbf{I}$, $\mathbf{b} = \mathbf{0}$, and

$$\hat{m}^n_i\hat{\mathbf{v}}^n_i = m^n_i\mathbf{v}^n_i + \mathbf{A}m^n_{r(i)}\mathbf{v}^n_{r(i)} + 2\mathbf{b}m^n_{r(i)} = m^n_i\mathbf{v}^n_i + m^n_{r(i)}\mathbf{v}^n_{r(i)} = \hat{m}^n_{r(i)}\hat{\mathbf{v}}^n_{r(i)} \implies \hat{\mathbf{v}}^n_i = \hat{\mathbf{v}}^n_{r(i)}$$

The total mass and momentum of the modified system is not the same as the original system, since we have additional particles. If we instead use the velocity from the modified system but keep our original masses, we find

$$m^n_i\hat{\mathbf{v}}^n_i + m^n_{r(i)}\hat{\mathbf{v}}^n_{r(i)} = m^n_i\hat{\mathbf{v}}^n_i + m^n_{r(i)}\hat{\mathbf{v}}^n_i = (m^n_i + m^n_{r(i)})\hat{\mathbf{v}}^n_i = \hat{m}^n_i\hat{\mathbf{v}}^n_i = m^n_i\mathbf{v}^n_i + m^n_{r(i)}\mathbf{v}^n_{r(i)}.$$

Total momentum is thus conserved by this boundary condition treatment. Since we did not change the grid mass, it is conserved as well. The slip case simply uses the no-slip case for the normal direction and the free-surface case for the tangential directions. It thus conserves the tangential momentum components and enforces the normal velocity component.

Reflection as a velocity correction. The boundary condition we get by introducing fictitious reflected particles has three undesirable properties: it does not conserve mass, it does not conserve momentum in the free-surface case, and it will cause problems during the proposed friction treatment. Instead, we use the reflection analogy only to compute a corrected velocity:

$$\hat{\mathbf{v}}^{n+1}_i = \frac{m^n_i\mathbf{v}^{n+1}_i + m^n_{r(i)}r(\mathbf{v}^{n+1}_{r(i)})}{m^n_i + m^n_{r(i)}} = \alpha^n_i\mathbf{v}^{n+1}_i + \alpha^n_{r(i)}r(\mathbf{v}^{n+1}_{r(i)}), \qquad \alpha^n_i = \frac{m^n_i}{m^n_i + m^n_{r(i)}} = 1 - \alpha^n_{r(i)}. \tag{21}$$

Note that, near corners, the mass and momentum contributions from all reflected ghost nodes must be included (4 nodes if near two walls, 8 nodes if near three walls). We also note that it is not necessary to apply a correction after the particle-to-grid transfer and then a second correction after forces. Instead, we can delay the correction until after forces have been applied to the velocity field but before grid data is transferred back to particles.

Note that our modified velocity in Eq. (21) takes the familiar form of a sticking collision between two particles of masses m_1, m_2, with velocities before collision of v_1, v_2, and velocities after collision given by $v_1' = v_2' = \frac{m_1}{m_1+m_2}v_1 + \frac{m_2}{m_1+m_2}v_2$. In our case, the different types of boundary conditions are achieved by the appropriate choice of $r(\mathbf{v})$. Directly at the wall $m_i = m_{r(i)}$, so the modified velocity will be an equally weighted average of $\mathbf{v}_i$ and $r(\mathbf{v}_{r(i)})$. As we move further from the wall, the relative weight of $\mathbf{v}_i$ increases smoothly. This leads to an exact enforcement of the boundary condition at the wall, and a continuous treatment of the boundary conditions in small band near the wall.

Friction. In the case of slip boundary conditions ($\mathbf{A} = \mathbf{R}$), it makes sense to consider friction. Based on the amount of normal force applied when enforcing the boundary condition, we wish to apply a Coulomb friction impulse to reduce the magnitude of the tangential velocity component. An obvious way to do this is to simply apply Coulomb friction to the grid nodes that receive normal forces. This approach does not work; objects experience a friction force, but this force does not converge to the correct value. In practice, a sliding object experiences less friction than it should. Instead, we approach this from the perspective of momentum.

As a result of enforcing the slip boundary condition by reflecting the velocity, the total tangential momentum was conserved but the normal component of total momentum changed. The difference in the normal component is

$$\begin{aligned}
\Delta p_n &= (m_i^n \hat{\mathbf{v}}_i^{n+1} + m_{r(i)}^n \hat{\mathbf{v}}_{r(i)}^{n+1}) \cdot \mathbf{n} - (m_i^n \mathbf{v}_i^{n+1} + m_{r(i)}^n \mathbf{v}_{r(i)}^{n+1}) \cdot \mathbf{n} \\
&= m_i^n (\hat{\mathbf{v}}_i^{n+1} - \mathbf{v}_i^{n+1}) \cdot \mathbf{n} + m_{r(i)}^n (\hat{\mathbf{v}}_{r(i)}^{n+1} - \mathbf{v}_{r(i)}^{n+1}) \cdot \mathbf{n} \\
&= m_i^n (\alpha_i^n \mathbf{v}_i^{n+1} + \alpha_{r(i)}^n r(\mathbf{v}_{r(i)}^{n+1}) - \mathbf{v}_i^{n+1}) \cdot \mathbf{n} + m_{r(i)}^n (\alpha_{r(i)}^n \mathbf{v}_{r(i)}^{n+1} + \alpha_i^n r(\mathbf{v}_i^{n+1}) - \mathbf{v}_{r(i)}^{n+1}) \cdot \mathbf{n} \\
&= m_i^n \alpha_{r(i)}^n (r(\mathbf{v}_{r(i)}^{n+1}) - \mathbf{v}_i^{n+1}) \cdot \mathbf{n} + m_{r(i)}^n \alpha_i^n (r(\mathbf{v}_i^{n+1}) - \mathbf{v}_{r(i)}^{n+1}) \cdot \mathbf{n} \\
&= m_i^n \alpha_{r(i)}^n (r(\mathbf{v}_{r(i)}^{n+1}) - \mathbf{v}_i^{n+1} + r(\mathbf{v}_i^{n+1}) - \mathbf{v}_{r(i)}^{n+1}) \cdot \mathbf{n} \qquad \text{Note: } m_i^n \alpha_{r(i)}^n = m_{r(i)}^n \alpha_i^n \\
&= m_i^n \alpha_{r(i)}^n (\mathbf{A}\mathbf{v}_{r(i)}^{n+1} + 2\mathbf{b} - \mathbf{v}_i^{n+1} + \mathbf{A}\mathbf{v}_i^{n+1} - \mathbf{v}_{r(i)}^{n+1} + 2\mathbf{b}) \cdot \mathbf{n} \\
&= 2m_i^n \alpha_{r(i)}^n (2\mathbf{b} - \mathbf{v}_i^{n+1} - \mathbf{v}_{r(i)}^{n+1}) \cdot \mathbf{n} \qquad (\text{using } \mathbf{n}^T\mathbf{A} = -\mathbf{n}^T).
\end{aligned}$$

We assume that $\mathbf{n}$ points towards the interior of the MPM domain. Then $\Delta p_n > 0$ corresponds to the wall pushing on the material, and $\Delta p_n < 0$ corresponds to the wall pulling the object towards it. For now, we assume that $\Delta p_n \geq 0$. Decomposing into normal and tangential components and scaling the tangential component,

$$\hat{\mathbf{v}}_{in}^{n+1} = \mathbf{n}\mathbf{n}^T \hat{\mathbf{v}}_i^{n+1}, \qquad \hat{\mathbf{v}}_{it}^{n+1} = (\mathbf{I} - \mathbf{n}\mathbf{n}^T)\hat{\mathbf{v}}_i^{n+1}, \qquad \hat{\mathbf{v}}_{i\mu}^{n+1} = \hat{\mathbf{v}}_{in}^{n+1} + s\hat{\mathbf{v}}_{it}^{n+1}, \qquad 0 \leq s \leq 1.$$

We note that $\hat{\mathbf{v}}_{it}^{n+1} = \hat{\mathbf{v}}_{r(i)t}^{n+1}$ and $\hat{\mathbf{v}}_{in}^{n+1} = 2\mathbf{b} - \hat{\mathbf{v}}_{r(i)n}^{n+1}$. In the case of full sticking, $s = 0$, and the change in tangential momentum due to friction is $\Delta p_t = \|m_i^n \hat{\mathbf{v}}_{it}^{n+1} + m_{r(i)}^n \hat{\mathbf{v}}_{r(i)t}^{n+1}\| = \hat{m}_i^n \|\hat{\mathbf{v}}_{it}^{n+1}\|$. In the case of sliding friction, for a given value of $s > 0$, the tangential change in momentum due to friction is $(1-s)\Delta p_t$. The Coulomb friction inequality gives $(1-s)\Delta p_t \leq \mu \Delta p_n$, so $s = \max(1 - \mu\frac{\Delta p_n}{\Delta p_t}, 0)$. With this, $\hat{\mathbf{v}}_{i\mu}^{n+1} = \hat{\mathbf{v}}_{in}^{n+1} + s\hat{\mathbf{v}}_{it}^{n+1}$ is the corrected velocity, with the effects of friction included. The reflected velocity is $\mathbf{v}_{r(i)\mu}^{n+1} = r(\hat{\mathbf{v}}_{i\mu}^{n+1}) = 2\mathbf{b} - \hat{\mathbf{v}}_{r(i)n}^{n+1} + s\hat{\mathbf{v}}_{it}^{n+1}$. This reflected value is precisely the same as would be obtained by repeating the entire derivation replacing i with $r(i)$.

Separation. All that remains is a proper treatment of the separation condition $\Delta p_n < 0$. A relatively obvious choice would be to revert to the free-surface boundary condition in this case, but this leads to a

velocity discontinuity in the normal direction. Disabling the boundary condition during separation leads to velocity discontinuity in the tangential direction. Instead we leave the normal velocity component untouched but use the reflected tangential components, $\mathbf{v}_{i,sep}^{n+1} = \mathbf{n}\mathbf{n}^T\mathbf{v}_i^{n+1} + \hat{\mathbf{v}}_{it}^{n+1}$. At $\Delta\mathbf{p}_n = 0$, we have $s = 1$, $(2\mathbf{b} - \mathbf{v}_i^{n+1} - \mathbf{v}_{r(i)}^{n+1}) \cdot \mathbf{n} = 0$, and

$$\begin{aligned}
\hat{\mathbf{v}}_{i\mu}^{n+1} - \mathbf{v}_{i,sep}^{n+1} &= (\hat{\mathbf{v}}_{in}^{n+1} + s\hat{\mathbf{v}}_{it}^{n+1}) - (\mathbf{n}\mathbf{n}^T\mathbf{v}_i^{n+1} + \hat{\mathbf{v}}_{it}^{n+1}) \\
&= \mathbf{n}\mathbf{n}^T\hat{\mathbf{v}}_i^{n+1} - \mathbf{n}\mathbf{n}^T\mathbf{v}_i^{n+1} \\
&= \mathbf{n}\mathbf{n}^T(\alpha_i^n\mathbf{v}_i^{n+1} + \alpha_{r(i)}^n(\mathbf{A}\mathbf{v}_{r(i)}^{n+1} + 2\mathbf{b}) - \mathbf{v}_i^{n+1}) \\
&= \alpha_{r(i)}^n\mathbf{n}\mathbf{n}^T(\mathbf{A}\mathbf{v}_{r(i)}^{n+1} + 2\mathbf{b} - \mathbf{v}_i^{n+1}) \\
&= \alpha_{r(i)}^n\mathbf{n}\mathbf{n}^T(2\mathbf{b} - \mathbf{v}_i^{n+1} - \mathbf{v}_{r(i)}^{n+1}) = \mathbf{0}.
\end{aligned}$$

This choice leads to velocity continuity and is also fully momentum conserving.

Summary. At the end of the grid update, we apply boundary conditions that do not include friction or separation using

$$\hat{\mathbf{v}}_i^{n+1} = \alpha_i^n\mathbf{v}_i^{n+1} + \alpha_{r(i)}^n r(\mathbf{v}_{r(i)}^{n+1}), \qquad \hat{\mathbf{v}}_{r(i)}^{n+1} = r(\hat{\mathbf{v}}_i^{n+1}), \qquad \alpha_i^n = \frac{m_i^n}{m_i^n + m_{r(i)}^n} = 1 - \alpha_{r(i)}^n.$$

If slip boundary conditions are being enforced with friction and separation, we compute the separation condition $c = (2\mathbf{b} - \mathbf{v}_i^{n+1} - \mathbf{v}_{r(i)}^{n+1}) \cdot \mathbf{n}$. If $c \geq 0$, we apply friction to the modified velocity

$$\hat{\mathbf{v}}_{i\mu}^{n+1} = \hat{\mathbf{v}}_{in}^{n+1} + s\hat{\mathbf{v}}_{it}^{n+1}, \quad \hat{\mathbf{v}}_{in}^{n+1} = \mathbf{n}\mathbf{n}^T\hat{\mathbf{v}}_i^{n+1}, \quad \hat{\mathbf{v}}_{it}^{n+1} = (\mathbf{I} - \mathbf{n}\mathbf{n}^T)\hat{\mathbf{v}}_i^{n+1}, \quad s = \max\left(1 - \frac{2\alpha_i^n\alpha_{r(i)}^n\mu c}{\|\hat{\mathbf{v}}_{it}^{n+1}\|}, 0\right),$$

with $\hat{\mathbf{v}}_{r(i)\mu}^{n+1} = r(\hat{\mathbf{v}}_{i\mu}^{n+1})$. Otherwise, we used the unmodified normal velocities and the modified tangential velocities,

$$\mathbf{v}_{i,sep}^{n+1} = \mathbf{n}\mathbf{n}^T\mathbf{v}_i^{n+1} + \hat{\mathbf{v}}_{it}^{n+1}, \qquad \mathbf{v}_{r(i),sep}^{n+1} = \mathbf{n}\mathbf{n}^T\mathbf{v}_{r(i)}^{n+1} + \hat{\mathbf{v}}_{it}^{n+1}.$$

5 Sources

Particle seeding at inlets must be done carefully to avoid visible discontinuities between different batches of seeded particles. We seed particles at the end of each time step, at which point the correct time step size Δt is available. We assume that all seeded particles entered the domain by passing through a plane (defined by a point $\mathbf{p}$ and normal $\mathbf{n}$) at some time $t^s = t^n + \Delta t - \Delta t_p$ (where $0 \leq \Delta t_p < \Delta t$) and at some location $\mathbf{z}_p$ on the seeding plane ($(\mathbf{z}_p - \mathbf{p}) \cdot \mathbf{n} = 0$). We assume that all particles pass through the seeding plane at constant source velocity $\mathbf{v}_s$ and evolve for the rest of the time step under a constant acceleration field $\mathbf{a}$ (typically gravity). Since its creation, the particle has accelerated to velocity $\mathbf{v}_p^{n+1} = \mathbf{v}_s + \Delta t_p\mathbf{a}$ and moved to position $\mathbf{x}_p^{n+1} = \mathbf{z}_p + \Delta t_p\mathbf{v}_s + \frac{1}{2}\Delta t_p^2\mathbf{a}$, which provide the initial position and velocity for the seeded particle. For APIC, we also need either the velocity gradient $\mathbf{C}_p^{n+1} = \frac{\partial \mathbf{u}}{\partial \mathbf{x}}$ or its analogue $\mathbf{B}_p^{n+1} = \mathbf{C}_p^{n+1}\mathbf{D}_p^n$. For this we need the spatial derivatives of the Eulerian velocity field $\mathbf{u}(\mathbf{x}, t)$, which is related to the Lagrangian velocity $\mathbf{v}_p(t)$ by $\mathbf{u}(\mathbf{x}_p(t), t) = \mathbf{v}_p(t)$. In order to calculate the velocity that a particle would have at a given location, we must determine where the particle was seeded and how long it has been evolving.

$$\begin{aligned}
\mathbf{x} &= \mathbf{z}_p + \Delta t_p\mathbf{v}_s + \frac{1}{2}\Delta t_p^2\mathbf{a} \\
\mathbf{x} - \mathbf{p} &= \mathbf{z}_p - \mathbf{p} + \Delta t_p\mathbf{v}_s + \frac{1}{2}\Delta t_p^2\mathbf{a} \\
(\mathbf{x} - \mathbf{p}) \cdot \mathbf{n} &= \Delta t_p\mathbf{v}_s \cdot \mathbf{n} + \frac{1}{2}\Delta t_p^2\mathbf{a} \cdot \mathbf{n}
\end{aligned}$$

This is a quadratic equation that can be solved for Δt_p. We note that we don't need to solve this equation; it is enough to implicitly differentiate it.

$$\begin{aligned}\frac{\partial}{\partial \mathbf{x}}((\mathbf{x}-\mathbf{p})\cdot\mathbf{n}) &= \frac{\partial}{\partial \mathbf{x}}\left(\Delta t_p \mathbf{v}_s\cdot\mathbf{n} + \frac{1}{2}\Delta t_p^2 \mathbf{a}\cdot\mathbf{n}\right)\\ \mathbf{n} &= (\mathbf{v}_s\cdot\mathbf{n} + \Delta t_p \mathbf{a}\cdot\mathbf{n})\frac{\partial \Delta t_p}{\partial \mathbf{x}}\\ \frac{\partial \Delta t_p}{\partial \mathbf{x}} &= \frac{\mathbf{n}}{(\mathbf{v}_s + \Delta t_p \mathbf{a})\cdot\mathbf{n}}\end{aligned}$$

Since $\mathbf{v}_p = \mathbf{v}_s + \Delta t_p \mathbf{a}$ depends on the query location $\mathbf{x}$ only through Δt_p (the seeding location $\mathbf{z}_p$ does not matter),

$$\begin{aligned}\mathbf{u}(\mathbf{x},t) &= \mathbf{v}_s + \Delta t_p \mathbf{a}\\ \frac{\partial \mathbf{u}}{\partial \mathbf{x}}(\mathbf{x},t) &= \mathbf{a}\left(\frac{\partial \Delta t_p}{\partial \mathbf{x}}\right)^T = \frac{\mathbf{a}\mathbf{n}^T}{(\mathbf{v}_s + \Delta t_p \mathbf{a})\cdot\mathbf{n}}\\ \mathbf{C}_p^{n+1} = \frac{\partial \mathbf{u}}{\partial \mathbf{x}}(\mathbf{x}_p^{n+1}, t^s + \Delta t_p) &= \frac{\mathbf{a}\mathbf{n}^T}{\mathbf{v}_p^{n+1}\cdot\mathbf{n}}.\end{aligned}$$

What remains is to seed particles uniformly and determine Δt_p and $\mathbf{z}_p$. We perform particle seeding in a reference volume, where we transform the seeding plane to the yz plane. The bounding box for the portion of this transformed plane where seeding must occur is $[y_0, y_1] \times [z_0, z_1]$; seeds inside this bounding box but not the seeding area are rejected after sampling. We use Poisson disk sampling [4, 10] to compute seeds $(x, y, z) \in [0, \Delta t \mathbf{v}_s \cdot \mathbf{n}] \times [y_0, y_1] \times [z_0, z_1]$, from which $\Delta t_p = \frac{x}{\mathbf{v}_s \cdot \mathbf{n}}$. $\mathbf{z}_p$ is obtained by transforming the point $(0, y, z)$ back into world space. Note that we have taken advantage of the fact that $\mathbf{a}$ does not affect particles until *after* they have been seeded and that Poisson disk sample distributions are insensitive to sheer. Note that seeding should *not* be performed directly on the box $(\Delta t_p, y, z) \in [0, \Delta t] \times [y_0, y_1] \times [z_0, z_1]$ due to the mixing of units. We retain a layer of seeds from the previous time step to use as starting seeds for the current time step; this ensures seamless continuity in the seeding distribution between time steps.

References

[1] S-wave. `https://en.wikipedia.org/wiki/S-wave`. Accessed: 2020-01-18.

[2] Philip T Barton, Dimitris Drikakis, Evgeniy Romenski, and Vladimir A Titarev. Exact and approximate solutions of riemann problems in non-linear elasticity. *Journal of Computational Physics*, 228(18):7046–7068, 2009.

[3] Craig R Bina and George R Helffrich. Calculation of elastic properties from thermodynamic equation of state principles. *Annual Review of Earth and Planetary Sciences*, 20(1):527–552, 1992.

[4] R. Bridson. Fast poisson disk sampling in arbitrary dimensions. In *SIGGRAPH sketches*, page 22, 2007.

[5] O. Ding, T. Shinar, and C. Schroeder. Affine particle in cell method for mac grids and fluid simulation. *Journal of Computational Physics*, 2019.

[6] Yu Fang, Yuanming Hu, Shi-Min Hu, and Chenfanfu Jiang. A temporally adaptive material point method with regional time stepping. In *Computer graphics forum*, volume 37, pages 195–204. Wiley Online Library, 2018.

[7] G. Irving, J. Teran, and R. Fedkiw. Invertible finite elements for robust simulation of large deformation. In *Proc. Symp. Comp. Anim.*, pages 131–140, 2004.

[8] C. Jiang, C. Schroeder, A. Selle, J. Teran, and A. Stomakhin. The affine particle-in-cell method. *ACM Trans Graph*, 34(4):51:1–51:10, 2015.

[9] C. Jiang, C. Schroeder, and J. Teran. An angular momentum conserving affine-particle-in-cell method. *J Comp Phys*, 338:137–164, 2017.

[10] G. Klár, T. Gast, A. Pradhana, C. Fu, C. Schroeder, C. Jiang, and T. Teran. Drucker-prager elasto-plasticity for sand animation. *ACM Transactions on Graphics (SIGGRAPH 2016).*, 2016.

[11] C. Schroeder. Practical course on computing derivatives in code. In *ACM SIGGRAPH 2019 Courses*, SIGGRAPH 19, New York, NY, USA, 2019. Association for Computing Machinery.

[12] A. Stomakhin, R. Howes, C. Schroeder, and J.M. Teran. Energetically consistent invertible elasticity. In *Proc. Symp. Comp. Anim.*, pages 25–32, 2012.