A Grid-Based Approach to Simulating Hair

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Clara Kim

Spring, 2022

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Advisor

Luther Tychonievich, Department of Computer Science

A Grid-Based Approach to Simulating Hair

Clara Kim
The University of Virginia
cjk8ad@virginia.edu

Abstract

We present a 2D simulation of hair strands in a fluid using a hybrid of a tridiagonal solver for inextensible hair and a material point method for fluid dynamics. For the fluid solver we used linear interpolation to transfer between particles and the grid. We implemented both particle-in-cell (PIC) and fluid-implicit-particle (FLIP) method; PIC was observed to better keep particles distributed across the grid than FLIP, which contained a large amount of noise and particle dispersion. To prevent strands from passing through other strands, we experimented with particle-based and grid-based approaches, most without success. We conclude that a background fluid simulation for implicit collision handling is not sufficient alone for solving strand interactions. We show the final integrated strand simulation which runs in real-time, but does not prevent undesired strand-strand crossing over from occurring.

CCS Concepts: • **Computing methodologies** → **Computer graphics**; *Animation*; Physical simulation.

Keywords: hair simulation, fluids, PIC, FLIP, MPM

1 Introduction

Hair continues to be the focus of numerous publications in computer graphics due to its importance to characterization in animation as well as its physical complexity, a survey of which can be found in [32]. This complexity stems from its structure as an aggregate of thousands of thin strands and the dynamic movement it is often expected to undergo.

Interactions between hair strands is further complicated by the structural properties of individual strands, which affect the dynamics of the overall structure [32]. Strands have irregular surfaces at a microscopic level, which causes friction between strands and causes them to move in clumps with neighboring strands. This observation has led to the formation of a range of different approaches to simulating hair, with some methods treating hair as a continuous medium, others utilizing a significantly smaller number of guide hair strands to capture large scale dynamics while interpolating for remaining strands, and others opting to model individual strands [21, 32].

While methods that are simplifications of the individual strand model are generally more efficient, finer details of hair interactions tend to be lost [21, 32]. This paper outlines an efficient grid-based approach to simulating hair strand dynamics. We represent strands as a series of connected particles, and preserve distance between particles to enforce inextensibility. An important feature in making strand-based hair dynamics look plausible is ensuring that hair strands do not pass through each other in visually dissonant ways. However, for particle-based hair strand systems, explicitly checking for strand segment collisions can be inefficient and difficult to control at a large scale [1, 21]. We integrate our strand model with a background fluid simulation to implicitly preserve volume and prevent strands from passing through each other.

2 Related Work

2.1 Hair Simulation

Hair dynamics simulation have been approached with several different methods, a survey of which can be found at [32].

Rosenblum et al. [27], Anjyo et al. [1] and Selle et al. [28] used strand representations to animate individual hair strands. Early approaches allowed for efficient single strand representations, but ignored hair-hair collisions for simplicity [1, 27].

Some models leverage the observation that neighboring hair strands tend to move similarly to each other by using a smaller number of guide strands to simulate movement of the overall hair mass and interpolating strands between these guides [7, 8, 19, 33]. Plante et al. (2001) [24] followed a similar principle for simulating interactions inside long hair by clustering hair strands into wisps comprised of a deformable envelope surrounding a central skeleton hair which dictates movement of the wisp.

Continuum models for hair simulation handle interactions between hair strands implicitly [12, 13]. Hadap and Magnenat-Thalmann [13] introduced the continuum model technique and animated hair interactions with Lagrangian fluid dynamics. To model individual hair strands, they used chains of rigid segments. Bando et al. [2] extended on this general idea, modeling the hair as a set of loosely coupled particles sampled to represent the hair volume. Petrovic et al. [23] uses a grid-based Eulerian fluid model to simulate and direct aggregate movement of hair volumes.

Our method is closely related to that of McAdams et al. [21] who modeled the collisions of many thin straight hairs using a mass/spring Lagrangian hair model, a FLIP based fluid solver to handle interactions and volume preservation of the overall hair volume, and Lagrangian edge-edge collision detection to solve for individual strand collisions from Bridson et al. [6]. Using this technique McAdams et al. simulated overall hair dynamics efficiently while also keeping the details at the strand level of a purely Lagrangian approach.

We similarly use a fluid solver to handle volume preservation of a Lagrangian strand system, but use a tridiagonal matrix formulation of distance constraints between particles rather than a mass/spring model and do not perform edge-edge collision detection, instead relying on the fluid solver to handle collisions and volume preservation implicitly.

2.2 PIC/FLIP Fluid Simulation

Historically, both Lagrangian and Eulerian [9, 12, 22, 25, 32] techniques have been used to simulate continuum materials. Lagrangian techniques represent dynamics with the interaction of particles. Eulerian techniques use a background fluid grid to perform pressure and viscosity updates to simulate flow. Hybrid Lagrangian/Eulerian material point methods (MPM), first introduced by Sulsky et al. [31], use Lagrangian particle representations with a Eulerian fluid grid. MPMs perform pressure and viscosity updates on the grid and advection on the Lagrangian particles [18]. To combine these techniques, MPMs involve a step that transfers information from the particles to the fluid grid cells and back.

Two methods for this transfer step are the Particle-in-cell (PIC) and fluid-implicit-particle (FLIP) methods [18]. PIC was first proposed by [15, 16]. The PIC method suffers from energy dissipation, causing the fluid to appear viscous. Brackbill and Ruppel [5] proposed the FLIP method to address this dissipation at the cost of stability.

Foster and Metaxas first used PIC techniques for graphics fluids simulation [11]. Zhu and Bridson [34] proposed a linear combination of PIC and FLIP to animate sand and liquid dynamics, which has been further extended by Bridson et al. for numerous other applications [3, 4, 10]. Stomakhin et al. [30] used MPM to simulate snow.

Jiang et al. [18] proposed the affine particle-in-cell (APIC) method to address the dissipation in PIC and the instability present in FLIP. The APIC technique produced simulations that avoided the instability and noise present in similar FLIP simulations as well as the viscous quality that appears in PIC and blended FLIP/PIC simulations [18]. The APIC method for particle-grid transfers has since been adopted for the simulation of a number of other phenomena [17, 20, 26].

We integrate a PIC and FLIP fluid solver with our strand simulation. While APIC improves upon both the PIC and FLIP techniques, the methods follow similar general concepts and PIC/FLIP are simpler to implement [18]. Integrating APIC into our method presented here is left to future work.

3 Method

We combine an inextensible Lagrangian strand simulation with a material point method fluid solver to simulate hair strand interactions. We enforce inextensibility in the strands using a tridiagonal matrix formulation as proposed by Han and Harada[14]. The linear system is formulated from distance constraints between adjacent points in the strand representations, and solved efficiently with a direct solver. We embed the strand solver in a PIC fluid solver to implicitly handle strand volume preservation and prevent strand crossing.

The structure of the main render loop is as follows:

1: **for** timestep t **do**
2: apply forces (particles)
3: fluid (grid)
4: hybrid collision handling
5: tridiagonal hair solver
6: draw particles & strands
7: **end for**

The following sections detail these steps.

3.1 Notation

Several variables are used throughout this section. To differentiate between calculations performed on the Lagrangian particles and Eulerian grid, subscripts p and q refer to particles and subscripts i and j refer to grid indices. The superscript is used to distinguish between quantities at the tth time step and quantities at the next $(t+1)$th, time step. Notation is summarized in table 1. Some notation is borrowed from Jiang et al. [17] and Han and Harada [14]. The relevant parts of their notation is included in table 1 with the source indicated.

3.2 Forces

At each time step t we apply position-based forces by updating particle velocities. The update for a single particle is as below.

$$\tilde{\mathbf{v}}_p^{t+1} = \mathbf{x}_p^t - \mathbf{x}_p^{t-1} + \mathbf{g} \tag{1}$$

Where $\mathbf{x}_p^t$ is the particle's position at time step t, $\mathbf{x}^{t-1}$ is the particle's position at the previous time step, $\tilde{\mathbf{v}}_p^{t+1}$ is the particle's intermediate velocity, and $\mathbf{g}$ is the force added at each step. To simulate gravitational force, the vertical component of $\mathbf{g}$ is negative and is the only nonzero element.

Other position-based forces can be updated using similar formulations. For example, a wind-line force could be defined with a magnitude proportional to the length of hair perpendicular to the wind direction.

3.3 Fluid

For PIC and FLIP, we used Lagrangian particles on a uniform Eulerian fluid grid. In this subsection we reproduce Jiang et al.'s expressions for PIC and FLIP for the reader's convenience [18]. We use Jiang et al.'s notation, a part of which can be found in Figure 1, but we use t instead of n to denote time steps.

3.3.1 PIC. Based on the fluid grid we create a staggered grid, such that the cell centers of the staggered grid falls

Symbol	Location	Type	Meaning	Source
t	g	s	timestep	
$\mathbf{g}$	p	v, c	gravitational force	
$\mathbf{f}$	p	v, c	other particle-based force	
m_p	p	s	mass	[18]
$\mathbf{x}_p^t$	p	v	position	[18]
$\mathbf{v}_p^t$	p	v	velocity	[18]
m_i^n	n	s	mass	[18]
$\mathbf{x}_i^t$	n	v	position	[18]
$\mathbf{v}_i^t$	n	v	velocity	[18]
$\tilde{\mathbf{v}}_p^{t+1}$	p	v	intermediate velocity	
$\tilde{\mathbf{v}}_i^{t+1}$	n	v	intermediate velocity	[18]
w_{ip}^t	n+p	s	weight for particle+cell pair	[18]
$N(\mathbf{x})$	g	s	interpolation kernel	[18]
$\mathbf{x}_{[i]}$	n	s	position component	
$P_{[i]}$	n	s	grid cell pressure	
$\mathbf{P}$	n	v	pressure vector	
$d_{[i]}$	n	s	grid cell divergence	
$\mathbf{d}$	n	v	divergence vector	
R	n	s, c	number of cells per row	
N	n	s, c	number of cells in grid	
n	p	s, c	number of particles per strand	
$C_{[i]}$	p	s	strand segment constraint	[14]
$\mathbf{C}$	p	v	constraint vector	[14]
$r_{[i]}$	p	s	strand segment rest length	[14]
$\mathbf{n}$	p	v	constraint gradient	[14]
$\Delta\lambda'$	p	v	increment of Lagrange multipliers	[14]
s	p	s	target distance between particles	

Table 1. Summary of notation used in this paper. Locations are p (particle), n (grid), or g (global; no location in space). Types are s (scalar) or v (vector). Constants are denoted with c in the Type column. Sources from which notation is replicated from are listed in the source column. No source indicates notation created for this paper.

on the grid cell edges of the fluid grid as in Figure 1. This allows for velocity across fluid grid cell edges to be found by determining velocity relative to staggered grid cell centers.

We store mass m_p, position $\mathbf{x}_p^t$, and velocity $\mathbf{v}_p^t$ on each particle p. At the beginning of each time step particle masses and momentums are transferred to the four nearest staggered grid cells for each of the staggered grids as below:

$$m_i^t = \sum_p w_{ip}^t m_p \tag{2}$$

$$m_i^t \mathbf{v}_i^t = \sum_p w_{ip}^t m_p \mathbf{v}_p^t \tag{3}$$

where the interpolations weights w_{ip}^t are given as follows:

$$w_{ip}^t = N(\mathbf{x}_p^t - \mathbf{x}_i^t) \tag{4}$$

Where $\mathbf{x}_i$ is the position of the center of a staggered grid cell. A visual representation of a particle p with position $\mathbf{x}_p^t$ and some nearby staggered grid cell center $\mathbf{x}_i^t$ is shown in Figure 2. In a 2-dimensional implementation the interpolation

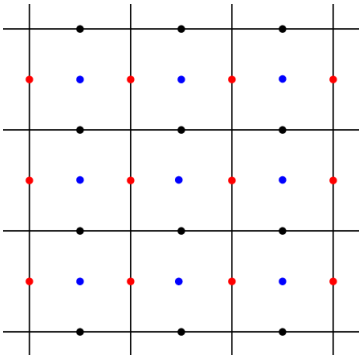

Figure 1. A staggered grid based on a 3-by-3 grid (cell boundaries shown with solid lines). Fluid cell centers are shown by blue points. Cell centers of the horizontally staggered grid are shown by red points. Cell centers of the vertically staggered grid are shown by black points.

kernel $N(\mathbf{x})$ is given by:

$$N(\mathbf{x}) = (1 - \mathbf{x}_{[1]})(1 - \mathbf{x}_{[2]}) \tag{5}$$

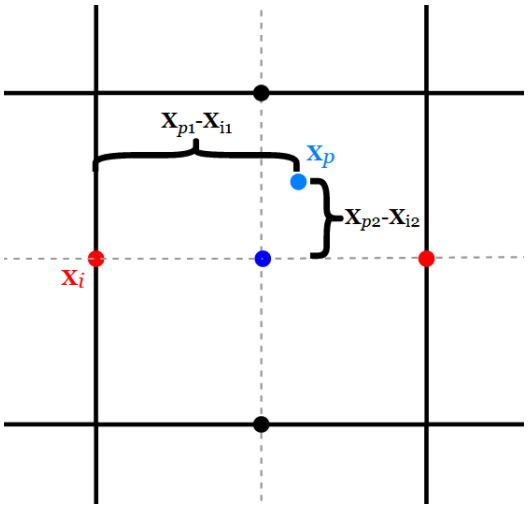

Figure 2. A sample particle $\mathbf{X}_p$ and the four nearest staggered grid cell centers. Components of the interpolation kernel $N(\mathbf{x})$ are shown for one of these cell centers. Here, the first component $\mathbf{x}_1$ of the position vectors are the horizontal component, and the second component $\mathbf{x}_2$ are the vertical.

We calculate velocities for each staggered grid cell using the mass and momentum found with equations 2 and 3.

$$\mathbf{v}_i^t = \frac{m_i^t \mathbf{v}_i^t}{m_i^t} = \frac{\sum_p w_{ip}^t m_p \mathbf{v}_p^t}{\sum_p w_{ip}^t m_p} \tag{6}$$

If no particle contributes mass and momentum to a cell, the numerator and denominator above are both 0; we treat such cells as having a velocity of 0.

Divergences are calculated for each fluid grid cell using the velocities in the staggered grid cells. Divergences are initially calculated separately for each dimension using the respective staggered grid velocities. The horizontal divergence of a fluid grid cell is calculated as the difference in velocity across its two vertical sides. Similarly the vertical divergence of a cell is calculated as the difference in velocity across its two horizontal sides.

For each fluid grid cell we calculate total divergence as the summation of both the horizontal and vertical divergence in that cell. We use these divergences to solve for pressures in grid cells by building a linear system.

We construct the linear system as follows. For a numbering of cells as is shown in Figure 3 we formulate an equation using the pressures $P_{[i]}$ and the divergence of the center cell, which we denote here as $d_{[k]}$. For the system to be

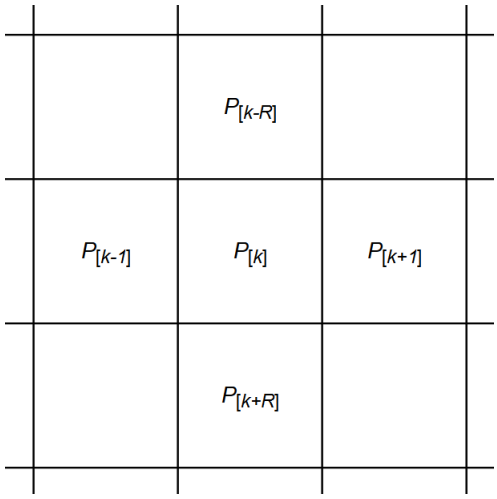

Figure 3. Grid cells with pressure $P_{[i]}$ for cell indexed by $[i]$. R is the number of grid cells in a single row.

divergence free, the following must be true:

$$\begin{aligned} d_{[k]} = (P_{[k]} - P_{[k-N]}) + (P_{[k]} - P_{[k-1]}) \\ + (P_{[k]} - P_{[k+1]}) + (P_{[k]} - P_{[k+N]}) \end{aligned} \tag{7}$$

where N is the number of cells in a single row.

Formulating similar equations for all N cells in the grid, we can create a linear system $A\mathbf{P} = \mathbf{d}$ with a sparse $N \times N$ matrix A. $\mathbf{d}$ is a N-length column vector containing all the cell divergences in the grid. One row of the A matrix contains at most 5 nonzero elements. A single row of A and the system is built from the example discussed above:

$$\begin{bmatrix} \dots & -1 & \dots & -1 & 4 & -1 & \dots & -1 & \dots \end{bmatrix} \begin{bmatrix} \vdots \\ P_{[k-R]} \\ \vdots \\ P_{[k-1]} \\ P_{[k]} \\ P_{[k+1]} \\ \vdots \\ P_{[k+R]} \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ d_{[k]} \\ \vdots \end{bmatrix} \tag{8}$$

Where only non-zero elements of the sample row from A are shown. Once the ordering of cells is established, the sparse matrix A is constant. Only the divergence vector $\mathbf{d}$ changes at each time step. As A is a sparse matrix with maximum 5 nonzero elements per row, we can save space by using a matrix storage technique that forgoes storing the full $N \times N$ matrix A.

We use the conjugate gradient method [29] to solve this system for the pressures at each grid cell. We use pressures at each cell to calculate $\tilde{\mathbf{v}}_i^{n+1}$ for cells in the staggered grids. Cell

centers on the staggered grids correspond to cell edges on the fluid grid. $\tilde{\mathbf{v}}_i^{n+1}$ is calculated as the difference in pressures over adjacent cells in the fluid grid. For the grid staggered in x we use horizontally adjacent cells, and for the grid staggered in y we use vertically adjacent cells.

We finally interpolate new velocities back to the particles using the intermediate velocities and weights.

$$\mathbf{v}_p^{n+1} = \sum_i w_{ip}^n \tilde{\mathbf{v}}_i^{n+1} \tag{9}$$

We move particles using a set time step Δt:

$$\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \Delta t \mathbf{v}_p^{n+1} \tag{10}$$

3.3.2 FLIP. FLIP follows the same technique as PIC, with the only difference being at the final step interpolating the new velocities to the particles. In FLIP, particle velocities are incremented by changes in cell velocities rather than directly interpolated from the new cell velocities as in PIC [18]. The final interpolation step for FLIP is shown below.

$$\mathbf{v}_p^{n+1} = \mathbf{v}_p^n = \sum_i w_{ip}^n (\tilde{\mathbf{v}}_i^{n+1} \mathbf{v}_i^n) \tag{11}$$

3.4 Hybrid Collision Handling

We tested two approaches for handling collisions, with one at the grid level and the other at the particle level.

3.4.1 Grid-based Approach. For the grid based approach to handling collisions, we calculate and store particle densities on each grid cell at every time step. A cap for the density per cell is set. If the cap is exceeded for a grid cell, the divergence of the grid cell is increased proportionally to the amount the density exceeded the set cap. This forces particles apart if many are clustered in a small area together.

3.4.2 Particle-based Approach. To handle collisions using particles, at each time step we do two passes over all the particles in the system.

On the first pass, we find the nearest neighboring particle for each particle in the system. This is done efficiently by only searching over particles stored in neighboring grid cells. We store the amount of force that the particle is exerting on the neighboring particle. This force is based on proximity of a pair of particles, with closer particles exerting greater force on each other, and is non-zero only when particles are within some target distance s from each other. The force is formulated as the vector difference between the actual offset of the particles and the target offset, as below:

$$s \frac{\mathbf{x}_{p_o}^t - \mathbf{x}_{p_c}^t}{\|\mathbf{x}_{p_o}^t - \mathbf{x}_{p_c}^t\|} - (\mathbf{x}_{p_o}^t - \mathbf{x}_{p_c}^t) \tag{12}$$

Where $\mathbf{x}_{p_c}^t$ is the position of the current particle, $\mathbf{x}_{p_o}^t$ is the position of the closest neighbor particle, and s is the target distance. This force is added to the total offsetting force acting on $\mathbf{x}_{p_o}^t$ only when it points in the same direction as the offset $\mathbf{x}_{p_o}^t - \mathbf{x}_{p_n}^t$, which indicates that the particles need to move further apart from each other to reach the target offset distance. Additionally, each particle stores the number of neighbor particles that is exerting forces on it.

On the second pass, we take the average amount of force exerted on each particle by its neighbors. We use this average force to move particles by setting particle positions offset by this force.

3.5 Tridiagonal Hair Solver

The following formulation for a tridiagonal strand solver has been adapted from Han and Harada's [14] formulation. We adapt most of their notation, but use $\mathbf{x}_p$ to denote particle position rather than $\mathbf{P}$ for consistency and convenience for the reader.

To enforce inextensibililty in strands, we take vertex positions $\mathbf{x}_p$ and finding the position corrections $\Delta\mathbf{x}_p$ needed to ensure that distance constraints between adjacent particles in a strand remain met after forces are applied. We do this by solving a system of linear equations. The matrix free formulation of this system as derived by Han and Harada is reproduced below for the reader's convenience [14].

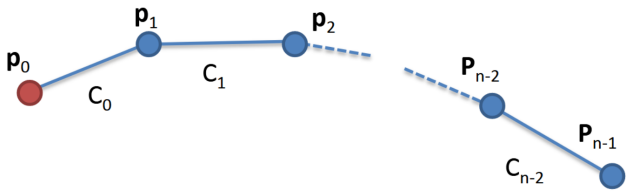

Figure 4. Chain structure for a strand, diagram from Han and Harada [14]. The first particle in the structure p_0 is given infinite mass.

We represent each strand as a chain structure of serially indexed vertices as in Figure 4, so that a strand is comprised of n particles $p_0, p_1, ...p_{n-1}$ (with positions $\mathbf{x}_{p_0}^t, \mathbf{x}_{p_1}^t, ...\mathbf{x}_{p_{n-1}}^t$) and distance constraints $C_0, C_1, ...C_{n-2}$ between pairs of adjacent vertices. We define a constraint vector $\mathbf{C}$ where $C_i = \|\mathbf{x}_{p_i}^t - \mathbf{x}_{p_{i+1}}^t\| - r_i$. r_i is the rest length between the particle positions $\mathbf{x}_{p_i}^t$ and $\mathbf{x}_{p_{i+1}}^t$. The constraint gradient is then defined as:

$$\frac{\delta C_i}{\delta \mathbf{x}_{p_i}^t} = -\frac{\delta C_i}{\delta \mathbf{x}_{p_{i+1}}^t} = \frac{\mathbf{x}_{p_i}^t - \mathbf{x}_{p_{i+1}}^t}{\|\mathbf{x}_{p_i}^t - \mathbf{x}_{p_{i+1}}^t\|} = \mathbf{n}_i \tag{13}$$

With these defined, a constraint equation is written as below to solve for $\Delta\lambda'$:

$$-\mathbf{n}_{i-1}\mathbf{n}_i \Delta\lambda'_{i-1} + 2\Delta\lambda'_i - \mathbf{n}_i\mathbf{n}_{i+1}\Delta\lambda'_{i+1} = \|\mathbf{x}_{p_i} - \mathbf{x}_{p_{i+1}}\| - r_i = C_i \tag{14}$$

This linear equation forms a symmetric and tridiagonal matrix, where the first, second, and third terms form the subdiagonal, diagonal, and superdiagonal, respectively. For

simulating hair one end of each strand is fixed. This is reflected in our definition. The first vertex is fixed by defining that particle to have infinite mass and the other particles are left free. The first constraint equation drops the first term and is then $\Delta\lambda'_0 - \mathbf{n}_0\mathbf{n}_1\Delta\lambda'_1 = \|\mathbf{x}_{p_0} - \mathbf{x}_{p_1}\| - r_0$. The last constraint equation drops the third term and is $-\mathbf{n}_{n-3}\mathbf{n}_{n-3}\Delta\lambda'_{n-3} + 2\Delta\lambda'_{n-3} = \|\mathbf{x}_{p_{n-2}} - \mathbf{x}_{p_{n-1}}\| - r_{n-2}$.

The matrix has a constant diagonal. It is also symmetric, and the superdiagonal and subdiagonal elements are calculated as the dot product of what is essentially two adjacent normalized edge vectors. To solve the system, we use the Thomas tridiagonal matrix algorithm to directly solve the system in a forward and backward sweep [14].

We find the position corrections as below after solving the linear system.

$$\Delta\mathbf{x}_{p_i} = \mathbf{n}_{i-1}\Delta\lambda'_{i-1} - \mathbf{n}_i\Delta\lambda'_i \tag{15}$$

4 Results

4.1 Tridiagonal Hair Solver

In Figure 5 we show a simulation of three strands each comprised of ten particles using only the Han and Harada formulation [14]. The simulation runs at 60fps (176 frames rendered in 3.12798 seconds, for 0.0177726 seconds per frame). Strands are initialized horizontally. Particle-based gravitational force is added to the particles at each time step as described by equation 1. The strands swing back and forth while preserving length as expected and described in [14], eventually stabilizing due to numerical damping with vertical hanging strands. No strand-strand collision handling is included.

4.2 Fluid Solver

In Figure 6 we show a comparison of the PIC and FLIP fluid simulations using a 30-by-30 fluid grid configuration with an initial density of four particles per cell (900 cells and 3,600 particles). The PIC simulation runs at roughly 47.5fps (501 frames rendered in 10.5531 seconds, for 0.0210641 seconds per frame). The FLIP simulation runs at roughly 48.0fps (501 frames rendered in 10.4219 seconds, for 0.0208022 seconds per frame). Particles initially in the upper grid cells (red) are given negative (leftward) horizontal forces and particles in the bottom cells (black) are given positive (rightward) horizontal forces. Both models oscillate around the center; PIC eventually stabilizes due to numerical damping; FLIP mostly stabilizes, but particles retain some oscillatory motion for thousands of frames because damping in FLIP applies only to cells, not individual particles. The PIC fluid displays more stable behavior, with the boundary between the red and black particles mostly preserved throughout the simulation. The FLIP fluid is more energetic and displays localized swirling patterns nearer to the boundary. Additionally, unlike the PIC fluid, the FLIP fluid allows for dispersion, with particles moving out of frame and requiring a separate particle containment step to prevent volume loss.

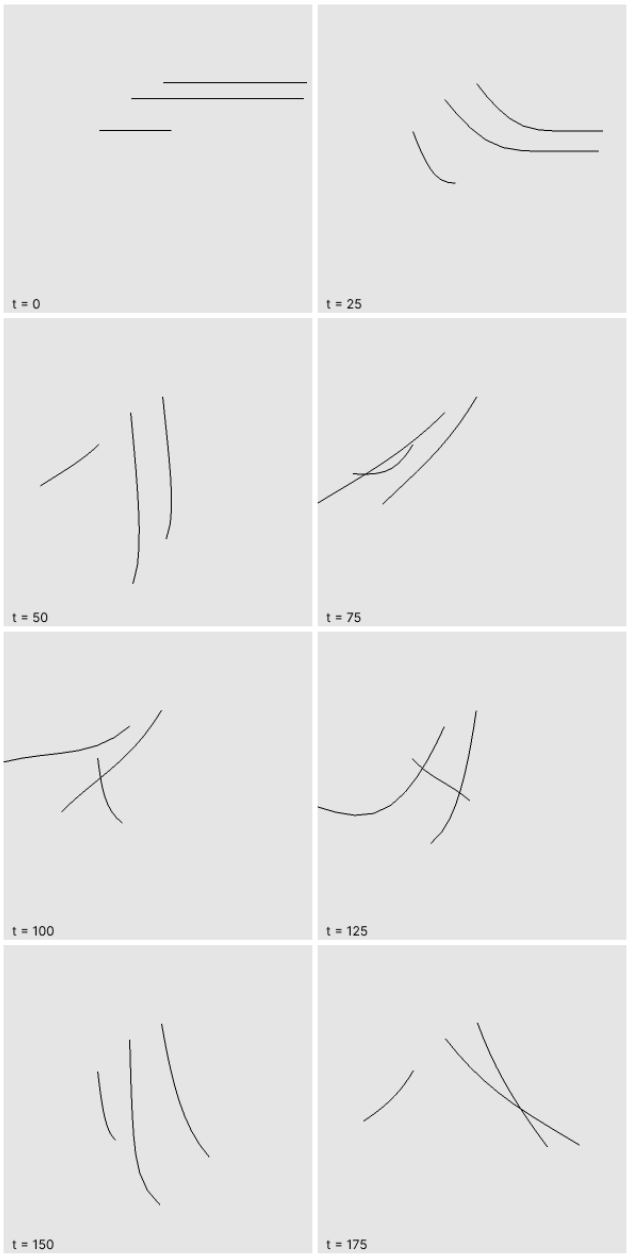

Figure 5. Motion of hairs in our implementation of the inextensible strands solver proposed by Han and Harada [14]. Strands begin at horizontal position at time $t = 0$ (top left); downward gravitational force is placed on each particle in subsequent time steps; snapshots are provided every 25 frames, as noted in the bottom-left label of each image.

4.3 Integrated Model

Figures 7 and 8 show inextensible strands integrated into a fluid solver. In each, 12 strands, each made up of 20 particles and immersed in fluid, begin at a horizontal position parallel to each other at time $t = 0$. The fluid grid is 50-by-50 cells and

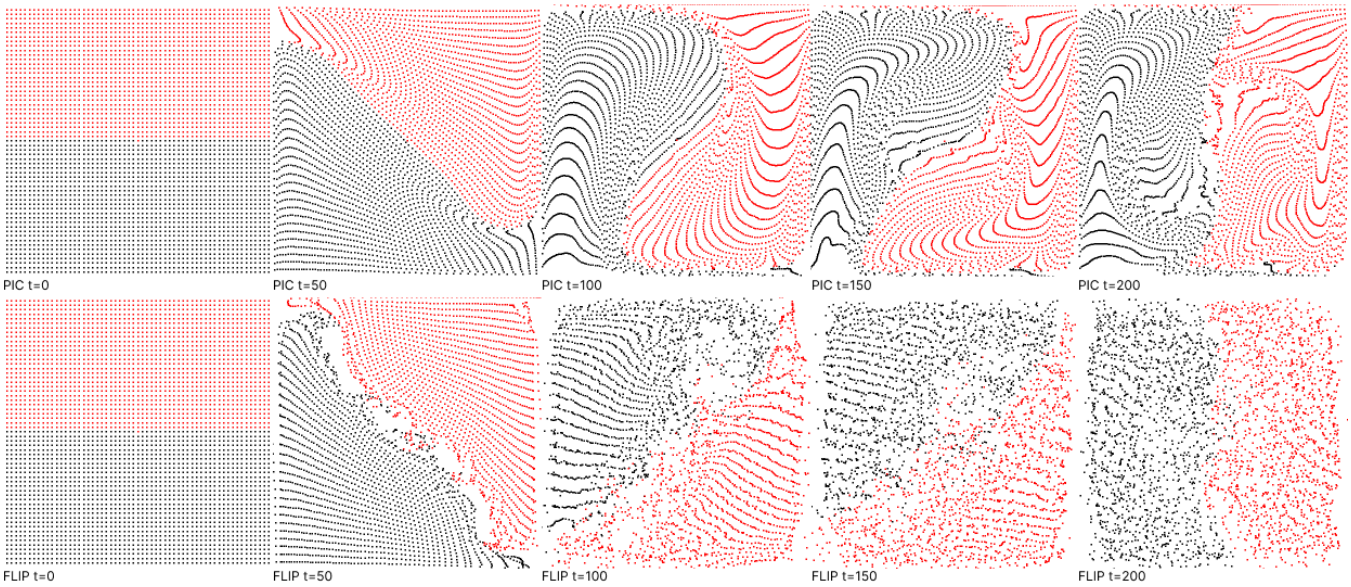

Figure 6. PIC and FLIP simulations. Particles in the bottom half (black) are given negative horizontal force. Particles in the top (red) are given positive horizontal force. Snapshots provided every 50 frames, as noted in the bottom-left label of each image.

initialized with a density of two particles per cell. Position-based gravitational force as described by equation 1 is applied to the particles in the strands.

Figure 7 shows the inextensible strands embedded in a FLIP fluid solver. The simulation runs at roughly 12.5fps (801 frames rendered in 64.1931 seconds, for 0.0801411 seconds per frame). Initially strand separations are prevented by the fluid forces, but eventually the gravitational and inextensibility solvers update their positions enough that the grid can no longer keep them separated and they pass through each other.

Figure 8 shows the strands embedded in the PIC fluid solver. The simulation runs at approximately 13.2fps (801 frames rendered in 60.5403 seconds, for 0.075581 seconds per frame). The strands behave similarly to those in the FLIP integrated model, with the grid delaying but not preventing strand overlap. The PIC-based model differs from the FLIP-based model primarily in the behavior of the surrounding fluid particles. Fluid particles in the PIC-based model exhibit clumping and striating patterns, which might be attributed to discontinuities in the velocity gradient caused by the linear interpolation kernel (equation 5).

Figure 9 shows another simulation of strands embedded in the PIC fluid solver, with strands anchored at the bottom and initialized in a vertical orientation. Volume is preserved. Again, strand overlap and fluid clumping is observed.

5 Conclusion and Future Work

We successfully integrated an inextensible strand simulation using a tridiagonal matrix formulation with a fluid simulation; the resulting simulation was able to run at interactive speeds. However, the combination of hair and fluid was not sufficient to prevent hair overlap.

Future explorations of a tridiagonal hair formulation paired with a fluid simulation may benefit from

- Adding explicit collision handling such as the edge-edge detection in [21].
- Recently the APIC method has gained popularity over PIC and FLIP as it avoids the issues of dissipation present in PIC and the instability found in FLIP [17, 18, 20, 26]. In future extensions of this research we would like to investigate the use the APIC method for the background fluid simulation as an alternative to the current PIC and FLIP-based model.
- Using a higher-order interpolation kernel $N(\mathbf{x})$. Linear interpolation has discontinuities in the first derivative at cell boundaries, which might explain the particle banding observed in some of our simulation results (Figure 6 and 8). Some previous papers have recommended cubic interpolation [18].
- Adding multiple fluids (such as air-water boundaries) and different boundary conditions between the fluid, the borders of the simulation, and the strands.
- Extending to 3D. In principle this would be as simple as replacing our 2D vectors with 3D vectors as none of our code depends on the 2D nature directly.
- Simulating larger numbers of hairs, hairs with moving anchors, and forces exerted by hairs on their anchor point.

We also hope this paper will assist others beginning to work with PIC, FLIP, and particle-based hair simulation.

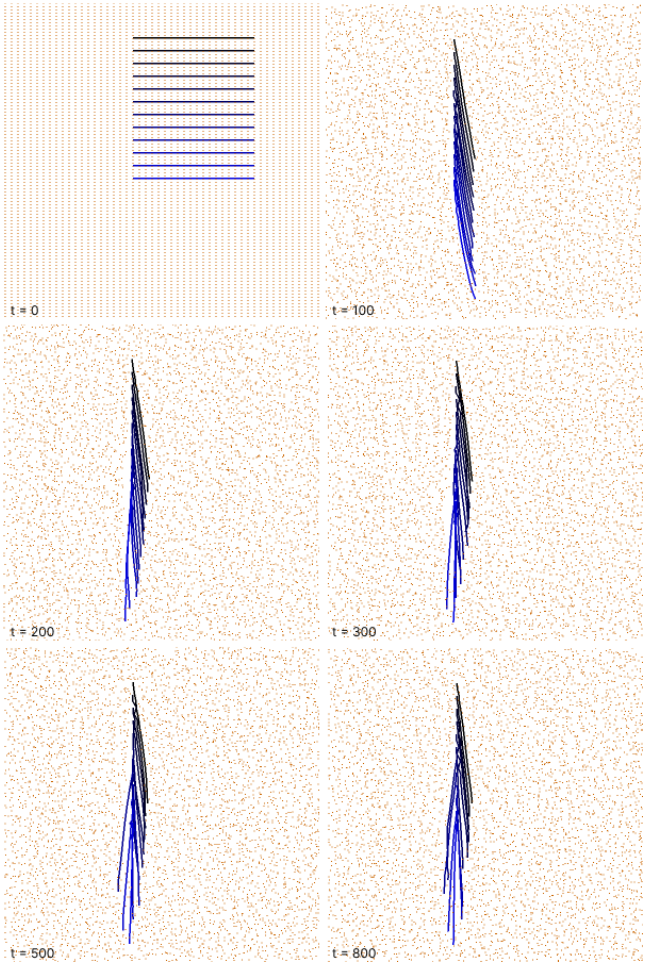

Figure 7. Integrated strand and FLIP fluid simulation. 12 strands, each comprised of 20 particles, are started at horizontal position at time $t = 0$ (top left). Position-based gravitational force is placed only on particles in each strand.

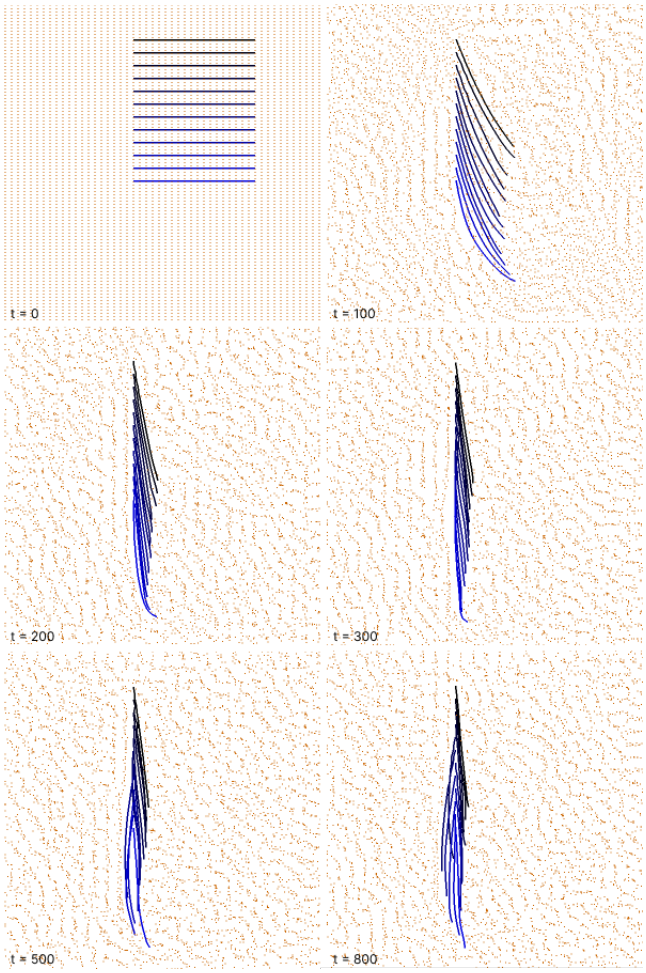

Figure 8. Integrated strand and PIC fluid simulation. 12 strands, each comprised of 20 particles, are started at horizontal position at time $t = 0$ (top left). Position-based gravitational force is placed only on particles in each strand.

6 Acknowledgements

We would like to thank Professor Luther Tychonievich, whose continuous support and guidance made this project possible.

References

[1] Ken-ichi Anjyo, Yoshiaki Usami, and Tsuneya Kurihara. 1992. A Simple Method for Extracting the Natural Beauty of Hair. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '92)*. Association for Computing Machinery, New York, NY, USA, 111–120. https://doi.org/10.1145/133994.134021

[2] Yosuke Bando, Bing-Yu Chen, and Tomoyuki Nishita. 2003. Animating Hair with Loosely Connected Particles. *Computer Graphics Forum* 22, 3 (2003), 411–418. https://doi.org/10.1111/1467-8659.00688 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/1467-8659.00688

[3] Christopher Batty and Robert Bridson. 2008. Accurate Viscous Free Surfaces for Buckling, Coiling, and Rotating Liquids. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Dublin, Ireland) *(SCA '08)*. Eurographics Association, Goslar, DEU, 219–228.

[4] Landon Boyd and Robert Bridson. 2012. MultiFLIP for Energetic Two-Phase Fluid Simulation. *ACM Trans. Graph.* 31, 2, Article 16 (apr 2012), 12 pages. https://doi.org/10.1145/2159516.2159522

[5] J.U. Brackbill and H.M. Ruppel. 1986. FLIP: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *J. Comput. Phys.* 65, 2 (1986), 314–343. https://doi.org/10.1016/0021-9991(86)90211-1

[6] Robert Bridson, Ronald Fedkiw, and John Anderson. 2002. Robust Treatment of Collisions, Contact and Friction for Cloth Animation. *ACM Trans. Graph.* 21, 3 (jul 2002), 594–603. https://doi.org/10.1145/566654.566623

[7] Johnny T. Chang, Jingyi Jin, and Yizhou Yu. 2002. A Practical Model for Hair Mutual Interactions. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (San Antonio, Texas) *(SCA '02)*. Association for Computing Machinery, New York, NY, USA, 73–80. https://doi.org/10.1145/545261.545273

[8] Agnes Daldegan, Nadia Magnenat Thalmann, Tsuneya Kurihara, and Daniel Thalmann. 1993. An Integrated System for Modeling, Animating and Rendering Hair. In *COMPUTER GRAPHICS FORUM*. 211–221.

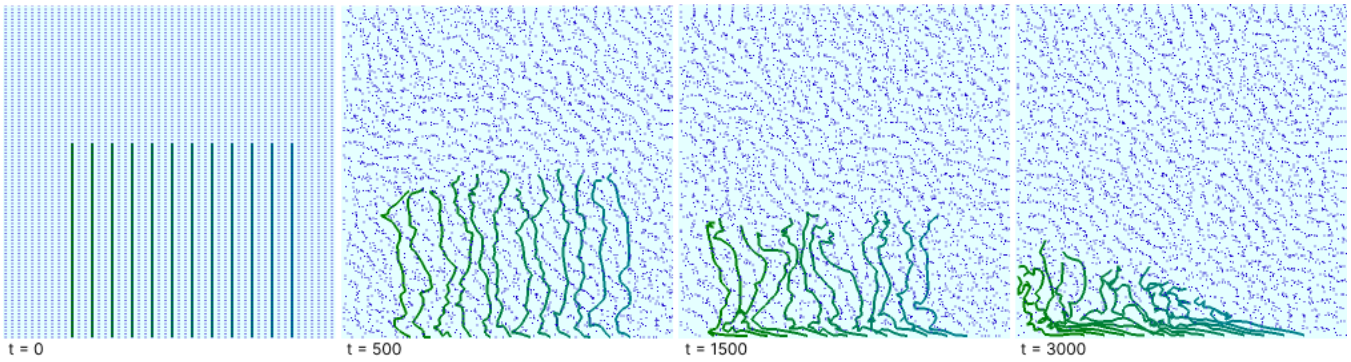

Figure 9. Integrated strand and PIC fluid simulation. 12 strands, each comprised of 30 particles, are started at vertical position at time $t = 0$ (left). Position-based gravitational force is placed only on particles in each strand. The simulation runs at roughly 10.3fps (3001 frames rendered in 290.899 seconds, or 0.0969341 seconds per frame).

[9] Mathieu Desbrun and Marie-Paule Gascuel. 1996. Smoothed Particles: A new paradigm for animating highly deformable bodies. In *Computer Animation and Simulation '96*, Ronan Boulic and Gerard Hégron (Eds.). Springer Vienna, Vienna, 61–76.

[10] Essex Edwards and Robert Bridson. 2014. Detailed Water with Coarse Grids: Combining Surface Meshes and Adaptive Discontinuous Galerkin. *ACM Trans. Graph.* 33, 4, Article 136 (jul 2014), 9 pages. https://doi.org/10.1145/2601097.2601167

[11] Nick Foster and Dimitri Metaxas. 1996. Realistic Animation of Liquids. In *Proceedings of the Conference on Graphics Interface '96* (Toronto, Ontario, Canada) *(GI '96)*. Canadian Information Processing Society, CAN, 204–212.

[12] Sunil Hadap, Marie-Paule Cani, Ming Lin, Tae-Yong Kim, Florence Bertails, Steve Marschner, Kelly Ward, and Zoran Kačić-Alesić. 2007. Strands and Hair: Modeling, Animation, and Rendering. In *ACM SIGGRAPH 2007 Courses* (San Diego, California) *(SIGGRAPH '07)*. Association for Computing Machinery, New York, NY, USA, 1–150. https://doi.org/10.1145/1281500.1281689

[13] Sunil Hadap and Nadia Magnenat-Thalmann. 2001. Modeling Dynamic Hair as a Continuum. *Computer Graphics Forum* 20, 3 (2001), 329–338. https://doi.org/10.1111/1467-8659.00525 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/1467-8659.00525

[14] Dongsoo Han and Takahiro Harada. 2013. Tridiagonal Matrix Formulation for Inextensible Hair Strand Simulation. In *Workshop on Virtual Reality Interaction and Physical Simulation*, Jan Bender, Jeremie Dequidt, Christian Duriez, and Gabriel Zachmann (Eds.). The Eurographics Association. https://doi.org/10.2312/PE.vriphys.vriphys13.011-016

[15] Francis H. Harlow. 1962. The particle-in-cell method for numerical solution of problems in fluid dynamics.

[16] Francis H. Harlow and J. Eddie Welch. 1965. Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface. *The Physics of Fluids* 8, 12 (1965), 2182–2189. https://doi.org/10.1063/1.1761178 arXiv:https://aip.scitation.org/doi/pdf/10.1063/1.1761178

[17] Chenfanfu Jiang, Theodore Gast, and Joseph Teran. 2017. Anisotropic Elastoplasticity for Cloth, Knit and Hair Frictional Contact. *ACM Trans. Graph.* 36, 4, Article 152 (jul 2017), 14 pages. https://doi.org/10.1145/3072959.3073623

[18] Chenfanfu Jiang, Craig Schroeder, Andrew Selle, Joseph Teran, and Alexey Stomakhin. 2015. The Affine Particle-in-Cell Method. *ACM Trans. Graph.* 34, 4, Article 51 (jul 2015), 10 pages. https://doi.org/10.1145/2766996

[19] Tsuneya Kurihara Ken-Ichi, Ken ichi Anjyo, and Daniel Thalmann. 1993. Hair Animation with Collision Detection. In *In Models and Techniques in Computer Animation*. Springer-Verlag, 128–138.

[20] Gergely Klár, Theodore Gast, Andre Pradhana, Chuyuan Fu, Craig Schroeder, Chenfanfu Jiang, and Joseph Teran. 2016. Drucker-Prager Elastoplasticity for Sand Animation. *ACM Trans. Graph.* 35, 4, Article 103 (jul 2016), 12 pages. https://doi.org/10.1145/2897824.2925906

[21] Aleka McAdams, Andrew Selle, Kelly Ward, Eftychios Sifakis, and Joseph Teran. 2009. Detail Preserving Continuum Simulation of Straight Hair. In *ACM SIGGRAPH 2009 Papers* (New Orleans, Louisiana) *(SIGGRAPH '09)*. Association for Computing Machinery, New York, NY, USA, Article 62, 6 pages. https://doi.org/10.1145/1576246.1531368

[22] Matthias Müller, David Charypar, and Markus Gross. 2003. Particle-Based Fluid Simulation for Interactive Applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (San Diego, California) *(SCA '03)*. Eurographics Association, Goslar, DEU, 154–159.

[23] Lena Petrovic, Markus Henne, and John Anderson Pixar. 2006. Volumetric Methods for Simulation and Rendering of Hair.

[24] Eric Plante, Marie-Paule Cani, and Pierre Poulin. 2001. A Layered Wisp Model for Simulating Interactions inside Long Hair. In *Eurographics Workshop on Computer Animation and Simulation (EGCAS)*, Daniel Thalmann and Nadia Magnenat-Thalmann (Eds.). EUROGRAPHICS, Springer, Manchester, United Kingdom, 139 – 148. https://hal.inria.fr/inria-00537500

[25] Simon Premžoe, Tolga Tasdizen, James Bigler, Aaron Lefohn, and Ross T. Whitaker. 2003. Particle-Based Simulation of Fluids. *Computer Graphics Forum* 22, 3 (2003), 401–410. https://doi.org/10.1111/1467-8659.00687 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/1467-8659.00687

[26] Daniel Ram, Theodore Gast, Chenfanfu Jiang, Craig Schroeder, Alexey Stomakhin, Joseph Teran, and Pirouz Kavehpour. 2015. A Material Point Method for Viscoelastic Fluids, Foams and Sponges. (2015), 157–163. https://doi.org/10.1145/2786784.2786798

[27] Robert E. Rosenblum, Wayne E. Carlson, and Edwin Tripp III. 1991. Simulating the structure and dynamics of human hair: Modelling, rendering and animation. *The Journal of Visualization and Computer Animation* 2, 4 (1991), 141–148. https://doi.org/10.1002/vis.4340020410 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/vis.4340020410

[28] Andrew Selle, Michael Lentine, and Ronald Fedkiw. 2008. A Mass Spring Model for Hair Simulation. *ACM Trans. Graph.* 27, 3 (aug 2008), 1–11. https://doi.org/10.1145/1360612.1360663

[29] Jonathan R Shewchuk. 1994. *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain.* Technical Report. USA.

[30] Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, and Andrew Selle. 2013. A Material Point Method for Snow Simulation. *ACM Trans. Graph.* 32, 4, Article 102 (jul 2013), 10 pages. https://doi.org/10.1145/2461912.2461948

[31] Deborah Sulsky, Shi-Jian Zhou, and Howard L. Schreyer. 1995. Application of a particle-in-cell method to solid mechanics. *Computer Physics Communications* 87, 1 (1995), 236–252. https://doi.org/10.1016/0010-4655(94)00170-7 Particle Simulation Methods.

[32] Kelly Ward, Florence Bertails, Tae-Yong Kim, Stephen R. Marschner, Marie-Paule Cani, and Ming C. Lin. 2007. A Survey on Hair Modeling: Styling, Simulation, and Rendering. *IEEE Transactions on Visualization and Computer Graphics* 13, 2 (mar 2007), 213–234. https://doi.org/10.1109/TVCG.2007.30

[33] Xue Dong Yang, Zhan Xu, Jun Yang, and Tao Wang. 2000. The Cluster Hair Model. *Graphical Models* 62, 2 (2000), 85–103. https://doi.org/10.1006/gmod.1999.0518

[34] Yongning Zhu and Robert Bridson. 2005. Animating Sand as a Fluid. *ACM Trans. Graph.* 24, 3 (jul 2005), 965–972. https://doi.org/10.1145/1073204.1073298