Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS (2018)
F. Jaillet, G. Zachmann, K. Erleben, and S. Andrews (Editors)

Laplacian Damping for Projective Dynamics

Jing Li[1] Tiantian Liu[2] Ladislav Kavan[1]

[1]University of Utah, United States
[2]University of Pennsylvania, United States

Abstract

Damping is an important ingredient in physics-based simulation of deformable objects. Recent work introduced new fast simulation methods such as Position Based Dynamics and Projective Dynamics. Explicit velocity damping methods currently used in conjunction with Position Based Dynamics or Projective Dynamics are simple and fast, but have some limitations. They may damp global motion or non-physically transport velocities throughout the simulated object. More advanced damping models do not have these limitations, but are slow to evaluate, defeating the benefits of fast solvers such as Projective Dynamics. We present a new type of damping model specifically designed for Projective Dynamics, which provides the quality of advanced damping models while adding only minimal computing overhead. The key idea is to define damping forces using Projective Dynamics' Laplacian matrix. In a number of simulation examples we show that this damping model works very well in practice. When used with a modified Projective Dynamics solver that uses a non-dissipative implicit midpoint integrator, our damping method provides fully user-controllable damping, allowing the user to quickly produce visually pleasing and vivid animations.

CCS Concepts

• ***Computing methodologies*** → ***Physical simulation;***

1. Introduction

Dissipation of mechanical energy is an important phenomenon in nature. In computer animation, damping is used to enhance stability of non-dissipative time integration schemes or to add aesthetic control to the results of simulations. Explicit damping methods produce fast damping effects, but may introduce unwanted artifacts. Explicit damping can be as simple as reducing each velocity in the system by a fixed ratio. This method is known as "ether drag" [SSF13]. A limitation of the ether drag is that it damps all velocities, even global rigid body motions. In order to mitigate this problem, Position Based Dynamics introduced a damping method ("PBD damping") that modifies the vertex velocities without changing the global motion, i.e. linear and angular momenta [MHHR07]. Even though this method successfully preserves the linear and angular momenta, it can non-physically transport velocities throughout the object and make it appear unnaturally rigid, because the underlying idea is to factor out linear and angular velocities as in a rigid body simulation. Implicit damping methods, on the other hand, can produce more visually satisfying results, but with extra computational cost.

Projective Dynamics [LBOK13,BML*14,LBK17] is a fast time integration method for elastic body simulations. Due to performance considerations, only explicit damping methods were considered in Projective Dynamics so far. In this paper, we propose a fast *implicit* damping method that can be naturally integrated into the framework of Projective Dynamics with negligible extra computational cost.

Our method is inspired by Rayleigh damping, which is extensively used in computer graphics [CAP17] [LKSH08] [GMD13] [FM17]. One feature of Rayleigh damping is that it is effective at removing high-frequency vibrations because the stiffness matrix acts as a high-pass filter [Wil02]. We use the Laplacian matrix to approximate this stiffness matrix because the Laplacian matrix is also a high-pass filter [Zha04] [Tau95] The key advantage of using the Laplacian matrix is that it is constant – independent of the current state of the simulated system – unlike the stiffness matrix. Therefore, Laplacian Damping can be efficiently incorporated in the Projective Dynamics framework with minimal computing overheads.

The original Projective Dynamics method is derived from backward Euler, also known as implicit Euler. Backward Euler is a strongly dissipative integrator, containing significant "built-in" numerical damping which is not user-controllable. We can combine Projective Dynamics with different time integrators, such as implicit midpoint. Implicit midpoint is non-dissipative and therefore can produce nice vivid motion, however, it is not always stable for large time steps. Our new damping method can be combined with implicit midpoint in the Projective Dynamics framework, producing stable motion that is as vivid motion as the user desires.

In summary, the main contributions of our new damping model are:

DOI: 10.2312/vriphys.20181065

- Our method does not exhibit the artifacts of fast explicit damping methods such as ether drag or PBD damping [MHHR07].
- Our method produces similar visual results to implicit damping methods such as lagged Rayleigh damping [GS14] or variational damping [KYT*06] while being much faster.
- Our method is easy to implement and fully compatible with the Projective Dynamics framework.
- Our method adds only minimal computing overheads on top of the underlying Projective Dynamics simulator.

2. Background and Related Work

Damping is commonly observed in nature. For purposes of numerical simulations, simple damping model, such as viscous damping, can be often assumed. Force corresponding to viscous damping can be expressed as:

$$\mathbf{f}_d = -\mathbf{D}\mathbf{v} \tag{1}$$

where $\mathbf{D}$ is a positive semi-definite damping matrix, $\mathbf{v}$ is the velocity of all particles in the simulated system, and $\mathbf{f}_d$ is the resulting damping force. $\mathbf{D}$ can be arbitrary, even time-dependent. The simplest choice of $\mathbf{D}$ is a constant scaled identity matrix: $\mathbf{D} := k_d\mathbf{I}$ where k_d is a constant damping coefficient [TPBF87]. Unfortunately, this damping model will also dissipate rigid body motions, i.e., global translation and rotation. This issue can be resolved by using a strain rate dependent damping matrix.

A special class of viscous damping methods is called Caughey damping [CO65]. It is designed for controlling the strength of damping for different vibrating components with different frequencies separately. The damping force in the Caughey damping model is still proportional to the velocity as described in Eq. 1, but the damping matrix becomes:

$$\mathbf{D}_{Caughey} := \mathbf{M}\sum_{i=1}^{p}\left(\alpha_i\left[\mathbf{M}^{-1}\mathbf{K}\right]^{i-1}\right) \tag{2}$$

where $\mathbf{M}$ is the mass matrix of the system, typically diagonally lumped; $\mathbf{K}$ is the stiffness matrix, i.e., the Hessian matrix of the elastic potential energy (or, equivalently, Jacobian of the elastic forces). In Eq. 2, α_i controls the strength of damping for vibration modes with different frequencies and p is the number of vibration modes. When only modes with the smallest vibration frequencies are considered, i.e. $p = 2$, the Caughey damping model reduces to its linear case known as Rayleigh damping [Ray96]:

$$\mathbf{D}_{Rayleigh} := \alpha_1\mathbf{M} + \alpha_2\mathbf{K} \tag{3}$$

Recently, Xu el al. [XB17] proposed an example based method that can converge $\mathbf{D}_{Rayleigh}$ towards $\mathbf{D}_{Caughey}$ by adding user-defined examples. This is essentially an artist-friendly way to approximate a Caughey damping matrix.

A common way to implement damping is by integrating damping forces in a separate numerical integration step, which can be either explicit or implicit [BMF05]. Ether drag [SSF13] is the most basic explicit damping method, which corresponds to multiplying all velocities by a constant (potentially spatially varying). Although this is very fast, ether drag shares a similar problem as [TPBF87] - even rigid body motions are damped, reducing both linear and angular momentum of a system. In order to fix this issue, Müller et al. [MHHR07] used a momentum-preserving explicit damping in Position Based Dynamics (PBD), which is almost as fast as ether drag. However, as we demonstrate in Figure 2, the results can be implausible in some cases due to non-physical transport of velocities through the simulated object. In cases where visual plausibility is more important than performance, implicit damping methods can be used. Bridson et al. [BMF05] combined a half step of implicit damping and another half step of explicit elastic force integration, producing highly realistic cloth simulation results. Kharevych et al. [KYT*06] proposed a new variational damping method, also incorporated a separate implicit time integration step.

Time Integration is one of the core ingredients of physically based simulation. Given the current state consisting of positions $\mathbf{x}_n$ and velocities $\mathbf{v}_n$, we need to predict the next state $\mathbf{x}_{n+1}$, $\mathbf{v}_{n+1}$ using Newton's second law of motion. Different time integration methods use different discrete estimates (quadratures) of the forces. Without damping, many implicit time integration schemes can be generally expressed as the following optimization problem:

$$g(\mathbf{x}) = \frac{1}{2h^2}||\mathbf{x}-\mathbf{y}||_M^2 + c_1E(c_2\mathbf{x}+\mathbf{z}) \tag{4}$$

this is a generalization of Eq.3 in [LBK17], where h is the time step size, $||\mathbf{x}||_{\mathbf{M}}^2 = \mathbf{x}^{\mathrm{T}}\mathbf{M}\mathbf{x}$ stands for the norm of $\mathbf{x}$ weighted by the mass matrix, $E(\mathbf{x})$ is the elastic energy evaluated at position $\mathbf{x}$, and the scalars c_1, c_2 and the vectors $\mathbf{y}$, $\mathbf{z}$ are time-integrator-dependent constants. For example, in backward Euler, $c_1 = c_2 = 1$, $\mathbf{y} = \mathbf{x}_n + h\mathbf{v}_n$ and $\mathbf{z} = 0$; in implicit midpoint, $c_1 = 1$, $c_2 = 0.5$, $\mathbf{y} = \mathbf{x}_n + h\mathbf{v}_n$ and $\mathbf{z} = \frac{\mathbf{x}_n}{2}$; in BDF-2, $c_1 = \frac{4}{9}, c_2 = 1$, $\mathbf{y} = \frac{4\mathbf{x}_n - \mathbf{x}_{n-1}}{3} + h\frac{8\mathbf{v}_n - 2\mathbf{v}_{n-1}}{9}$ and $\mathbf{z} = 0$. The goal of the time integration is to minimize $g(\mathbf{x})$ to find the next state position $\mathbf{x}_{n+1} = argmin_{\mathbf{x}}g(\mathbf{x})$, then to update the next state velocity $\mathbf{v}_{n+1}$ explicitly according to the integration rules (a local minimum is sufficient). This variational integration formulation usually provides a more stable numerical solution compared to nonlinear root finding [KYT*06, MTGG11]. This is because the minimization problem could at least find a local minimum, which could be a reasonable solution, while the root-finding could simply fail.

The optimization formulation was investigated by Gast et al. [GS14], who used a "lagged" Rayleigh damping matrix. Specifically, their damping force is defined as:

$$\mathbf{f}_d = -\mathbf{D}(\mathbf{x}_n)\mathbf{v}_{n+1} = -(\alpha_1\mathbf{M} + \alpha_2\mathbf{K}(\mathbf{x}_n))\mathbf{v}_{n+1} \tag{5}$$

$$\nabla E_d(\mathbf{x}) = -\mathbf{f}_d = \mathbf{D}(\mathbf{x}_n)\mathbf{v}_{n+1} \tag{6}$$

With this formulation, it is possible to derive corresponding viscous damping "energy" E_d which can be appended to Eq. 4 and handled just like the elastic potential E (the main difference is that this "damping potential" depends on the current state $\mathbf{x}_n$, hence this is not a classical potential function). The elastic energy is often non-convex and therefore $\mathbf{D}$ may be indefinite and the damping model can thus erroneously *increase* velocities. In order to ensure that the damping energy will be reducing velocities as expected, a "definiteness fix" process to guarantee semi-definiteness of $\mathbf{D}(\mathbf{x}_n)$ is often recommended [GS14]. After the indefiniteness is eliminated, this damping energy can be defined by anti-differentiating Eq. 6 on the

position of the vertices x:

$$E_d(\mathbf{x}) = \int_{\mathbf{x}} \mathbf{D}(\mathbf{x}_n)\mathbf{v}_{n+1}(\mathbf{x})d\mathbf{x} = \frac{c_2 h}{2}||\mathbf{v}_{n+1}(\mathbf{x})||^2_{\mathbf{D}(\mathbf{x}_n)} \quad (7)$$

where c_2 is the same as in Eq. 4, and $\mathbf{v}_{n+1}(\mathbf{x})$ represents the velocity update rule for a chosen time integrator.

Projective Dynamics (PD) [BML*14] is one of the acceleration schemes to solve Eq. 4, assuming a certain structure of the elastic energy:

$$E_{PD}(\mathbf{x}) = \sum_j w_j E_j(\mathbf{x}) \quad (8)$$

where w_j is the weight of the j-th element, encoding the size of the element and the stiffness of the material for that element. E_j is a special type of elastic energy:

$$E_j(\mathbf{x}) = \min_{\mathbf{p}_j \in \mathcal{M}_j} \tilde{E}_j(\mathbf{x},\mathbf{p}) \quad \tilde{E}_j(\mathbf{x},\mathbf{p}) = \frac{1}{2}||\mathbf{G}_j\mathbf{x} - \mathbf{S}_j\mathbf{p}||^2_F \quad (9)$$

where $||.||_F$ is the Frobenius norm, $\mathcal{M}_j$ is a constraint manifold, $\mathbf{p}$ is a stacked projection variable, $\mathbf{S}_j$ selects a certain $\mathbf{p}_j = \mathbf{S}_j\mathbf{p}$ being restricted to manifold $\mathcal{M}_j$, and $\mathbf{G}_j$ is the differential operator. For example, if we want to represent an as-rigid-as-possible energy [CPSS10], we can simply set $\mathcal{M}_j$ to $SO(3)$, and use $\mathbf{G}_j\mathbf{x}$ to represent the deformation gradient of the j-th element.

In projective dynamics, the position $\mathbf{x}$ and projection variable $\mathbf{p}$ are updated using a local/global solver. After substituting Eq. 8 into Eq. 4, we can rewrite the new objective function as:

$$\begin{aligned}&\frac{1}{2h^2}||\mathbf{x}-\mathbf{y}||^2_{\mathbf{M}} + \frac{1}{2}\mathbf{x}^\mathsf{T}(c_1c_2^2\mathbf{L})\mathbf{x} \\ &-\mathbf{x}^\mathsf{T}(c_1c_2\mathbf{J}\mathbf{p} - c_1c_2\mathbf{L}\mathbf{z}) + const.\end{aligned} \quad (10)$$

where $\mathbf{L} = \sum_j w_j\mathbf{G}_j^\mathsf{T}\mathbf{G}_j$ and $\mathbf{J} = \sum_j w_j\mathbf{G}_j^\mathsf{T}\mathbf{S}_j$. Whenever $\mathbf{x}$ is fixed, updating $\mathbf{p}$ only requires one loop over all elements and projecting $\mathbf{G}_j(c_2\mathbf{x}+\mathbf{z})$ onto the desired manifolds $\mathcal{M}_j$; this is usually referred to as the "local step" of PD. With $\mathbf{p}$ is fixed, we can compute the optimal $\mathbf{x}$ by solving setting the gradient of Eq. 10 over $\mathbf{x}$ equal to zero. The solution $\mathbf{x}^*$ is:

$$\mathbf{x}^* = \left(\frac{\mathbf{M}}{h^2} + c_1c_2^2\mathbf{L}\right)^{-1}\left(\frac{\mathbf{M}\mathbf{y}}{h^2} + c_1c_2\mathbf{J}\mathbf{p} - c_1c_2\mathbf{L}\mathbf{z}\right) \quad (11)$$

Note that the system matrix $\frac{\mathbf{M}}{h^2} + c_1c_2^2\mathbf{L}$ only depends on the topology of the mesh, the stiffness of all elements, the mass of all vertices and the time-step where all of the quantities are unlikely to change during a simulation. [BML*14] pre-factorized this matrix and reused this factor to solve for $\mathbf{x}^*$. This solve is usually referred as the "global step". The local/global steps are repeated in an alternating way for a fixed number of iterations, typically between 10 to 20.

The original Projective Dynamics algorithm uses backward Euler integration [BML*14], which features artificial numerical damping which is not user-controllable. In real-time physics we typically use large time steps, where this artificial damping can be significant. If more damping is desired, [BML*14] uses an explicit post-processing damping method described in [MHHR07] to further damp the velocity without reducing the momentum. Implicit damping methods such as Rayleigh damping are hard to integrate with Projective Dynamics, because the stiffness matrix $\mathbf{K}$ in Rayleigh damping changes dynamically and thus needs to be refactorized often. This negates the performance gains of Projective Dynamics.

3. Method

We propose a modified implicit Rayleigh damping model, where the damping matrix is redefined as:

$$\mathbf{D}_{Ours} = \alpha_1\mathbf{M} + \alpha_2\mathbf{L} \quad (12)$$

where $\mathbf{L} = \sum_j w_j\mathbf{G}_j^\mathsf{T}\mathbf{G}_j$ is exactly the same Laplacian matrix as in Eq. 10. Unlike the original Rayleigh damping model where the stiffness matrix $\mathbf{K}$ plays the role of slowing down high frequency vibrations, we choose the Laplacian matrix $\mathbf{L}$ to achieve visually similar damping effects. We call our corresponding damping model the "Laplacian damping model".

The stiffness matrix $\mathbf{K}$ of a deformable body works as a high-pass filter – it produces large damping forces that act against high-frequency vibrations. The Laplacian matrix $\mathbf{L}$ shares the same property and, indeed, related Laplacian-type matrices are often used in geometry processing to "smooth out" high spatial frequencies. Because constant vectors are in the null-space of $\mathbf{L}$, if $\mathbf{v}$ corresponds to a global translation, then $\mathbf{L}\mathbf{v} = 0$. This shows there is no damping of global translational motion. In deformable body simulations, the Laplacian matrix also encodes the stiffness information and mesh topology like a stiffness matrix. Based on these observations, we believe that the Laplacian matrix could be a good constant replacement of the stiffness matrix used in Rayleigh damping. We verified this intuition experimentally, as shown in Figure 3, indicating that our Laplacian damping model can produce results that are qualitatively similar to those obtained from the original Rayleigh damping model.

After modifying the Rayleigh damping matrix, we can anti-differentiate our damping force to evaluate our damping energy:

$$\begin{aligned}E_d(\mathbf{x}) &= \int_{\mathbf{x}} \mathbf{D}_{Ours}\mathbf{v}_{n+1}(\mathbf{x})d\mathbf{x} \\ &= \frac{c_2 h}{2}||\mathbf{v}_{n+1}(\mathbf{x})||^2_{\mathbf{D}_{Ours}}\end{aligned} \quad (13)$$

where the scalar c_2 and velocity update rule $\mathbf{v}_{n+1}(\cdot)$ depends on the specific time integration rule. Since $\mathbf{D}_{Ours}$ is not dependent on the position $\mathbf{x}$, we do not need to use a "lagged" version (as in [GS14]) of the damping matrix anymore. Also, because the Laplacian matrix is guaranteed to be positive semi-definite, we do not need to fix its definiteness either, which simplifies implementation and improves run-time performance. We simply plug our damping energy Eq. 13 into our variational formulation (Eq. 4) and solve the optimization using local/global solve similar to Projective Dynamics.

To be specific, let us explain our local/global steps with implicit midpoint integration, where $c_1 = 1$, $c_2 = 0.5$, $\mathbf{z} = \frac{\mathbf{x}_n}{2}$, and the velocity update rule is $\mathbf{v}_{n+1} = 2\frac{\mathbf{x}_{n+1}-\mathbf{x}_n}{h} - \mathbf{v}_n$.

In the **local step** where the position vector $\mathbf{x}$ is fixed, we solve for the projection variable $\mathbf{p}_j$ by projecting $\mathbf{G}_j\left(\frac{\mathbf{x}+\mathbf{x}_n}{2}\right)$ to its corresponding manifold $\mathcal{M}_j$. This step is exactly the same as in the original Projective Dynamics, because the damping energy does *not* affect the elastic energy where the local step takes place.

Example	#Verts	#Elems	Integration Method	Integration Time (without damping)	Integration Time (with damping)	Damping Overhead
ribbon (Figure 2)	549	728	implicit midpoint	10ms	10ms	<1ms
bar (Figure 8)	738	1920	implicit midpoint	40ms	43ms	3ms
penguin (Figure 1)	1979	6915	implicit midpoint	151ms	151ms	<1ms
dummy (Figure 5)	4492	16890	implicit midpoint	670ms	674ms	4ms
pig (Figure 3)	5122	16505	implicit midpoint	477ms	486ms	9ms
hair (Figure 7)	1900	600	backward Euler	42ms	43ms	1ms

Table 1: *Summary of our example simulations, all using the Projective Dynamics framework with 10 iterations. The "Integration Time" is total time per frame of our method with / without damping.*

In the **global step** where the projection variable $\mathbf{p}$ is already computed, we compute the optimal positions by solving the following problem:

$$\frac{1}{2h^2}||\mathbf{x}-\mathbf{y}||^2_{\mathbf{M}}+\frac{1}{2}\mathbf{x}^{\mathsf{T}}(\frac{1}{4}\mathbf{L})\mathbf{x}-\mathbf{x}^{\mathsf{T}}(\frac{1}{2}\mathbf{Jp}-\frac{1}{2}\mathbf{Lz}) \\ +\frac{h}{4}\left|\left|2\frac{\mathbf{x}-\mathbf{x}_n}{h}-\mathbf{v}_n\right|\right|^2_{\mathbf{D}_{Ours}}+const. \tag{14}$$

which is exactly Eq. 10 with our damping energy Eq. 13 appended at the end. The exact solution $\mathbf{x}^*$ of this global problem can be computed analytically as follows:

$$\mathbf{x}^* = \left(\frac{\mathbf{M}}{h^2}+\frac{1}{4}\mathbf{L}+\frac{2}{h}\mathbf{D}_{Ours}\right)^{-1} \\ \left(\frac{\mathbf{My}}{h^2}+\frac{1}{2}\mathbf{Jp}-\frac{1}{2}\mathbf{Lz}+\mathbf{D}_{Ours}(\frac{2}{h}\mathbf{x}_n+\mathbf{v}_n)\right) \tag{15}$$

Note that because $\mathbf{D}_{Ours}$ is the combination of the mass matrix $\mathbf{M}$ and the Laplacian matrix $\mathbf{L}$, we can prefactorize the system matrix $\frac{\mathbf{M}}{h^2}+\frac{1}{4}\mathbf{L}+\frac{2}{h}\mathbf{D}_{Ours}$ as well. Therefore the global system can be solved efficiently at run time. As we can see from Table 1, the overhead of adding damping with our method is typically very small because we do not disturb the efficient local/global solve process of Projective Dynamics.

4. Results

Table 1 summarizes all of our experiments including the run times for our method utilizing the Projective Dynamics framework. All of our simulations use ten Projective Dynamics iterations. We can see that our damping model adds only a very small computing overhead compared to Projective Dynamics without damping. This small computational cost is spent on evaluating $\mathbf{D}_{Ours}(\frac{2}{h}\mathbf{x}_n+\mathbf{v}_n)$ in Eq. 15. All timings were measured with an Intel Core i7-4910MQ CPU. In all of our examples we have set the global damping coefficient α_1 (in Eq. 3) to zero to avoid damping of global translational motion, producing more vivid animations. However, our system supports any $\alpha_1 > 0$ in case global damping is requested by the user. We choose to compare our method with ether drag and PBD damping because they are fast damping method used with Projective Dyanmics so far. And we choose to compare our method with variational damping and the lagged Rayleigh damping because they are high-quality damping method used with more computationally intensive simulation methods In Figure 1, we show a deformable penguin falling and bouncing on the ground. In this example, implicit midpoint would explode without damping, as we show in the

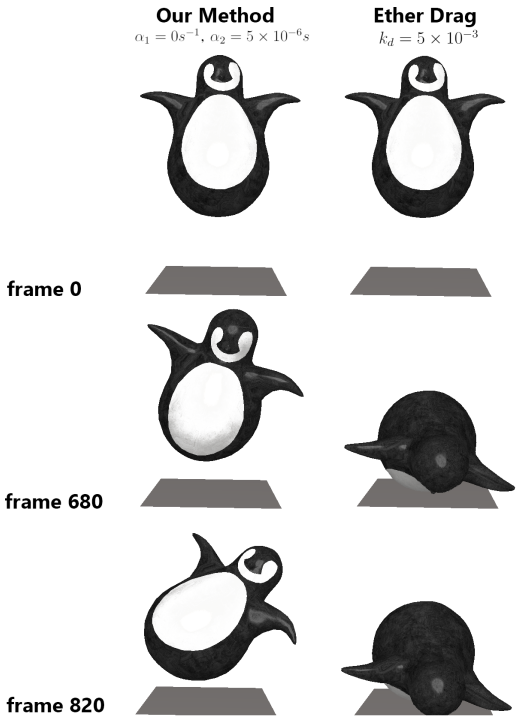

Figure 1: *Results produced by our method (left) and ether drag (right). Using our method, the penguin keeps bouncing. Using ether drag, the penguin stops moving earlier.*

accompanying video. This explosion can be prevented with ether drag. We found the minimal damping coefficient of ether drag that produces stable animation: 5×10^{-3}. Even though this prevents the explosions, the side effect is slowing of the global translational motion of the penguin. We repeat the same experiment with our method using implicit midpoint as the underlying integrator. In this case, we set the damping coefficient of our method to 5×10^{-6}, which is the smallest amount of damping needed to stabilize the simulation. We can see that in this case, the global motion of the penguin is not slowed down, while the simulation remains stable and visually pleasing.

PBD damping [MHHR07] is an explicit damping method which

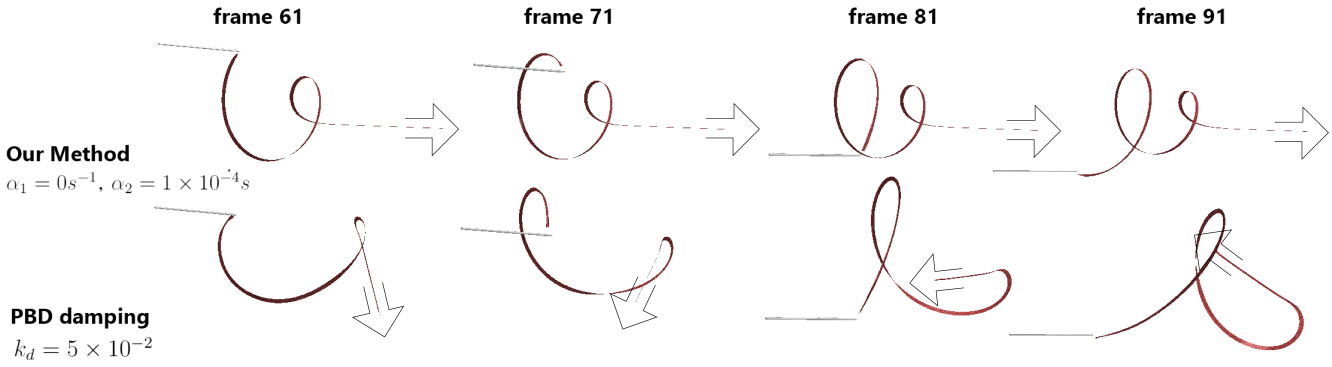

Figure 2: *Our method makes the ribbon coil in a natural and fabric-like fashion, similar to ribbon motions seen in rhythmic gymnastics performances. In contrast, PBD damping introduces unnatural early rotation of the tail of the ribbon, as if the ribbon was a rigid bar rather than fabric. The arrows point out the differences in the motion of the ribbon tail.*

preserves both linear and angular momenta. However, since this is achieved by explicitly factoring out global rigid motion, the PBD damping might preserve momenta in an unnatural, non-physical way. In Figure 2, we show an example of this behavior using a simulated ribbon attached to a wand. The wand circles around while translating backwards, similar to a performance of rhythmic gymnast. With the PBD damping applied to the ribbon, the tail of the ribbon is moving when the end attached to the wand just started rotating. But in real physical world, other than rigid body, the velocities cannot be transported immediately from one end to the other. The velocities from one end of the object are non-physically transported to velocities in the opposite end of the object, and the entire ribbon starts spinning almost like a rigid body. In contrast, our method models local damping forces, resulting in natural coiling motion of the ribbon as seen in real-world gymnastics performances.

In Figure 3, the snout of a pig is pulled and released, resulting in a comical animation. We compared our method with lagged Rayleigh damping [GS14], noticing the methods differ only in small details such as the motion of the ears. The visual differences are hard to distinguish without careful examination. The lagged Rayleigh damping uses the definiteness-fixed Hessian matrix from the previous time step. Because this matrix changes at each frame, we recompute the Cholesky factorization each frame, which is slower than our method, see Figure 4. However, we note that faster numerical methods for the lagged Rayleight damping are possible [GS14]. In Figure 5 we compare our method with the "variational damping" approach introduced by Kharevych and colleagues [KYT*06]. We implemented variational damping in a separate integration step, where the previous state is treated as rest-pose configuration. The resulting optimization problem is solved by Newton's method, which is more computationally intensive than Projective Dynamics. The visual difference of the final result with our damping method and the variational damping is barely noticeable, but our method is much faster, as shown in Figure 6.

In Figure 7, we simulate hair strands as a mass-spring system, shaken from side to side. In this simulation, we use backward Euler and observe that its artificial numerical damping is not sufficient – the hair is too bouncy. With our method, we can introduce additional damping and achieve a more sensible hair animation. In Figure 8 we show that our damping method behaves well under re-

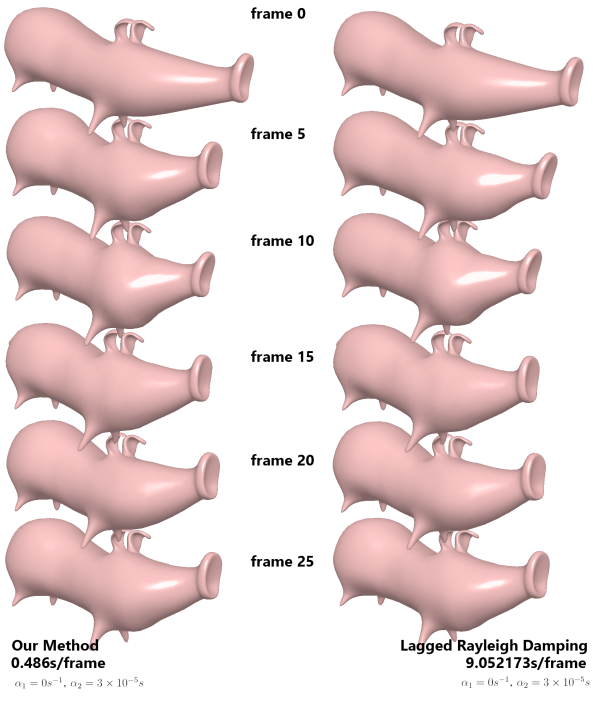

Figure 3: *Results produced by our method (left) and the lagged Rayleigh damping [GS14] (right). Our method looks almost the same as the lagged Rayleigh damping, but our method is much faster, as shown in Figure 4.*

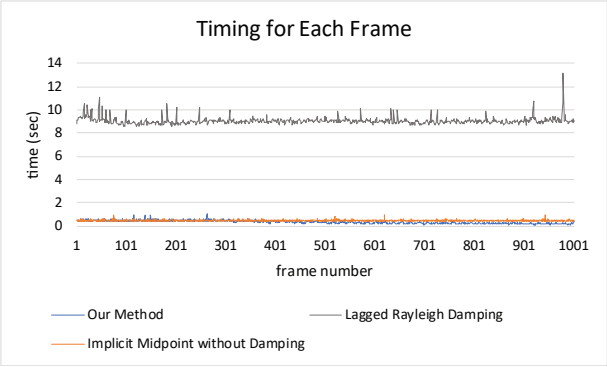

Figure 4: *Computing time per animation frame for our method with implicit midpoint, lagged Rayleigh damping, and implicit midpoint without damping in the deformable pig animation (Figure 3). All methods use ten iterations.*

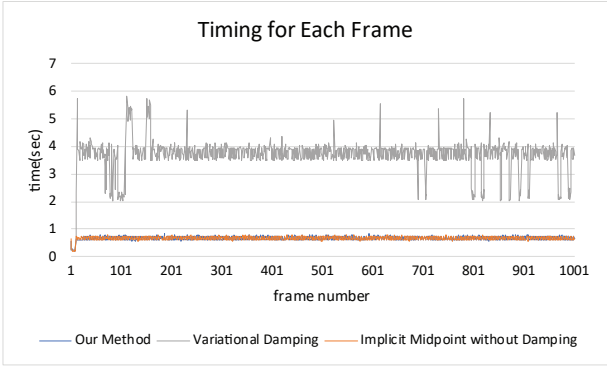

Figure 6: *Computing time per animation frame for variational damping, implicit midpoint without damping, and implicit midpoint with our method. Times are for the dummy-punching simulation shown in Figure 5.*

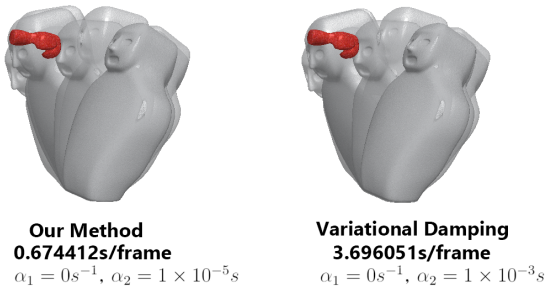

Figure 5: *Results produced by our method (left) and variational damping are visually similar, but our method is much faster, as shown in Figure 6.*

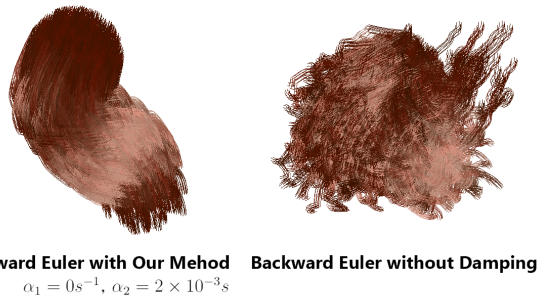

Figure 7: *Hair animation with backward Euler with damping (our method, left) and without damping (backward Euler only, right).*

finement. In this example, a thin sheet is fixed at one edge, and the other end drops freely under gravity. Notice that the results for time steps of 1.65ms and 0.165ms are very similar, but are different for time steps of 33ms.

5. Limitations and Future Work

Our method shares one limitations with Projective Dynamics: modifications of the connectivity of the simulated system or its parameters (including the damping coefficient) require re-factorization of the Cholesky factors. Our method conserves linear momentum, but does not conserve angular momentum exactly. In the future, we believe that we can derive a modification of our method to exactly conserve angular momentum. Another avenue for future work would involve extending our method to more advanced damping models, such as Caughey damping or example-based damping design [XB17].

6. Conclusion

We introduced a new implicit damping method which can be combined with various time integration schemes under the Projective Dynamics framework. Our method produced higher quality results than explicit damping methods. Unlike more expensive implicit damping methods such as lagged Rayleigh damping and variational damping, our method did not significantly reduce the performance of Projective Dynamics. The visual results of our method were comparable to results produced with more computationally intensive implicit damping methods. We believe our new damping method will find use in interactive applications such as games or surgical training simulators.

Acknowledgements

We thank Junior Rojas, Saman Sepehri, and Cem Yuksel for many inspiring discussions. We also thank Yasunari Ikeda for help with hair rendering, Nathan Marshak and Dimitar Dinev for proofreading, Shirley Han and Jessica Hair for narrating the accompanying video. This work was supported by NSF awards IIS-1622360 and IIS-1350330

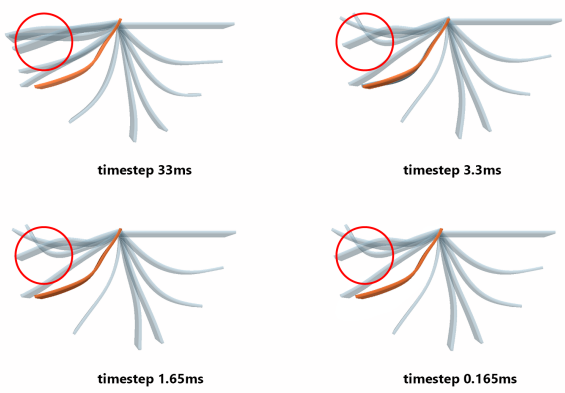

Figure 8: *Convergence experiment with a flexible sheet simulated with different time steps. Small time steps lead to similar results.*

References

[BMF05] BRIDSON R., MARINO S., FEDKIW R.: Simulation of clothing with folds and wrinkles. In *ACM SIGGRAPH 2005 Courses* (2005), ACM, p. 3. 2

[BML*14] BOUAZIZ S., MARTIN S., LIU T., KAVAN L., PAULY M.: Projective dynamics: fusing constraint projections for fast simulation. *ACM Trans. Graph. 33*, 4 (2014), 154. 1, 3

[CAP17] CHEN Y. J., ASCHER U., PAI D.: Exponential rosenbrock-euler integrators for elastodynamic simulation. *IEEE Transactions on Visualization and Computer Graphics* (2017). 1

[CO65] CAUGHEY T., O'KELLY M. E.: Classical normal modes in damped linear dynamic systems. *Journal of Applied Mechanics 32*, 3 (1965), 583–588. 2

[CPSS10] CHAO I., PINKALL U., SANAN P., SCHRÖDER P.: A simple geometric model for elastic deformations. In *ACM SIGGRAPH 2010 Papers* (2010), SIGGRAPH '10, ACM, pp. 38:1–38:6. 3

[FM17] FRÂNCU M., MOLDOVEANU F.: Unified simulation of rigid and flexible bodies using position based dynamics. 1

[GMD13] GLONDU L., MARCHAL M., DUMONT G.: Real-time simulation of brittle fracture using modal analysis. *IEEE Transactions on Visualization and Computer Graphics 19*, 2 (2013), 201–209. 1

[GS14] GAST T. F., SCHROEDER C.: Optimization integrator for large time steps. In *Eurographics/ACM SIGGRAPH Symposium on Computer Animation* (Copenhagen, Denmark, 2014), Eurographics Association. 2, 3, 5

[KYT*06] KHAREVYCH L., YANG W., TONG Y., KANSO E., MARSDEN J. E., SCHRÖDER P., DESBRUN M.: Geometric, variational integrators for computer animation. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2006), Eurographics Association, pp. 43–51. 2, 5

[LBK17] LIU T., BOUAZIZ S., KAVAN L.: Quasi-newton methods for real-time simulation of hyperelastic materials. *ACM Transactions on Graphics (TOG) 36*, 3 (2017), 23. 1, 2

[LBOK13] LIU T., BARGTEIL A. W., O'BRIEN J. F., KAVAN L.: Fast simulation of mass-spring systems. *ACM Transactions on Graphics 32*, 6 (Nov. 2013), 209:1–7. (Proceedings of ACM SIGGRAPH Asia 2013, Hong Kong). 1

[LKSH08] LLOYD B. A., KIRAC S., SZÉKELY G., HARDERS M.: Identification of dynamic mass spring parameters for deformable body simulation. In *Eurographics (Short Papers)* (2008), Citeseer, pp. 131–134. 1

[MHHR07] MÜLLER M., HEIDELBERGER B., HENNIX M., RATCLIFF J.: Position based dynamics. *Journal of Visual Communication and Image Representation 18*, 2 (2007), 109–118. 1, 2, 3, 4

[MTGG11] MARTIN S., THOMASZEWSKI B., GRINSPUN E., GROSS M.: Example-based elastic materials. In *ACM Transactions on Graphics (TOG)* (2011), vol. 30, ACM, p. 72. 2

[Ray96] RAYLEIGH J. W. S. B.: The theory of sound. *Macmillan 2* (1896). 2

[SSF13] SU J., SHETH R., FEDKIW R.: Energy conservation for the simulation of deformable bodies. *IEEE Transactions on Visualization and Computer Graphics 19*, 2 (Feb. 2013), 189–200. 1, 2

[Tau95] TAUBIN G.: A signal processing approach to fair surface design. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (1995), ACM, pp. 351–358. 1

[TPBF87] TERZOPOULOS D., PLATT J., BARR A., FLEISCHER K.: Elastically deformable models. *SIGGRAPH Comput. Graph. 21*, 4 (Aug. 1987), 205–214. 2

[Wil02] WILSON E. L.: *Three-Dimensional Static and Dynamic Analysis of Structures*, 3rd ed. Computers and Structures, Inc., 2002. 1

[XB17] XU H., BARBIČ J.: Example-based damping design. *ACM Trans. Graph. 36*, 4 (July 2017), 53:1–53:14. 2, 6

[Zha04] ZHANG H.: Discrete combinatorial laplacian operators for digital geometry processing. In *Proceedings of SIAM Conference on Geometric Design and Computing. Nashboro Press* (2004), pp. 575–592. 1

Appendices

A. Assembly of $\mathbf{G}_j$ and $\mathbf{p}$

A.1. Finite Element Method

We denote the number of vertices as n and the number of elements (tetrahedra) as m. For one single tetrahedron, the four vertex indices are i_1, i_2, i_3, i_4. In this tetrahedron, we assume that $i_2 < i_1 < i_3 < i_4$

$$\mathbf{G}_j = \mathbf{D}_M^{-T} \mathbf{A}_j \otimes \mathbf{I}_3 \tag{16}$$

where $\mathbf{D}_M$ is the reference shape matrix, and $\mathbf{A}_j \in \mathbb{R}^{3\times n}$ can be written as

$$\mathbf{A}_j = \begin{array}{c} \begin{matrix} i_2 & i_1 & i_3 & i_4 \end{matrix} \\ \begin{pmatrix} & 1 & & -1 \\ 1 & & & -1 \\ & & 1 & -1 \end{pmatrix} \end{array} \tag{17}$$

where i_1, i_2, i_3, i_4 denote the column numbers, and all the entries other than 1 and -1 are 0.

$$\mathbf{p} = \begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \vdots \\ \mathbf{p}_m \end{pmatrix} \tag{18}$$

where $\mathbf{p}_j \in \mathbb{R}^{9\times 1}$, $\mathbf{p} \in \mathbb{R}^{9m\times 1}$.

A.2. Spring

We denote the number of vertices as n and the number of elements(springs) as m. For one single spring, the two vertex indices are i_1 and i_2.

$$\mathbf{G}_j = \frac{1}{r} \mathbf{A}_j \otimes \mathbf{I}_3 \tag{19}$$

where r is the rest length of the spring, and $\mathbf{A}_j \in \mathbb{R}^{1\times n}$ can be written as

$$\mathbf{A}_j = \begin{array}{c} \\ (\end{array} \begin{array}{cc} i_1 & i_2 \\ 1 & -1 \end{array} \begin{array}{c} \\ ) \end{array} \tag{20}$$

where i_1, i_2 denote the column numbers, and all the entries other than 1 and -1 are 0.

$$\mathbf{p} = \begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \vdots \\ \mathbf{p}_m \end{pmatrix} \tag{21}$$

where $\mathbf{p}_j \in \mathbb{R}^{3\times 1}$, $\mathbf{p} \in \mathbb{R}^{3m\times 1}$.

B. Proof of preservation of linear momentum

For a single tetrahedron, the vertices of which are $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$, $\mathbf{x}_4$, the velocities of each vertex are $\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{v}_3$, $\mathbf{v}_4$. We consider the case where the motion of the tetrahedron is pure translation, which means $\mathbf{v}_1 = \mathbf{v}_2 = \mathbf{v}_3 = \mathbf{v}_4$.

The deformation gradient is given as

$$\mathbf{F} = \mathbf{D}_S \mathbf{D}_M^{-1} \tag{22}$$

We vectorize $\mathbf{D}_S$ as

$$\begin{aligned} vec(\mathbf{D}_S) &= \begin{bmatrix} \mathbf{x}_1 - \mathbf{x}_4 \\ \mathbf{x}_2 - \mathbf{x}_4 \\ \mathbf{x}_3 - \mathbf{x}_4 \end{bmatrix} \\ &= (\mathbf{A_j} \otimes \mathbf{I}_3)\mathbf{x} \end{aligned} \tag{23}$$

Thus the deformation gradient can be vectorized as well as:

$$\begin{aligned} vec(\mathbf{F}) &= vec(\mathbf{D}_S \mathbf{D}_M^{-1}) \\ &= (\mathbf{D}_M^{-T} \otimes \mathbf{I}_3) vec(\mathbf{D}_S) \\ &= (\mathbf{D}_M^{-T} \otimes \mathbf{I}_3)(\mathbf{A}_j \otimes \mathbf{I}_3)\mathbf{x} \\ &= (\underbrace{\mathbf{D}_M^{-T}\mathbf{A}_j \otimes \mathbf{I}_3}_{\mathbf{G}_j \in \mathbb{R}^{9\times 3n}})\mathbf{x} \end{aligned} \tag{24}$$

Our corresponding Laplacian matrix will be:

$$\begin{aligned} \mathbf{L} &= \sum \omega_j (\mathbf{D}_M^{-T}\mathbf{A}_j \otimes \mathbf{I}_3)^T (\mathbf{D}_M^{-T}\mathbf{A}_j \otimes \mathbf{I}_3) \\ &= \sum \omega_j (\mathbf{D}_M^{-T}\mathbf{A}_j)^T \otimes \mathbf{I}_3 \mathbf{D}_M^{-T}\mathbf{A}_j \otimes \mathbf{I}_3 \\ &= \sum \omega_j \mathbf{A_j}^T \mathbf{D}_M^{-1}\mathbf{D}_M^{-T}\mathbf{A}_j \otimes \mathbf{I}_3 \end{aligned} \tag{25}$$

yielding the damping force:

$$\mathbf{L}\mathbf{v} = \sum \omega_j \mathbf{A}_j^T \mathbf{D}_M^{-1}\mathbf{D}_M^{-T}(\mathbf{A}_j \otimes \mathbf{I}_3\mathbf{v}) \tag{26}$$

where

$$\mathbf{A}_j \otimes \mathbf{I}_3\mathbf{v} = \begin{pmatrix} \mathbf{v}_1 - \mathbf{v}_4 \\ \mathbf{v}_2 - \mathbf{v}_4 \\ \mathbf{v}_3 - \mathbf{v}_4 \end{pmatrix} = \mathbf{0} \tag{27}$$

Therefore, the damping force $\mathbf{L}\mathbf{v}$ is $\mathbf{0}$ under the translation mode, preserving the linear momentum of the system.