Average Vector Field Integration for St. Venant-Kirchhoff Deformable Models

Junior Rojas, Tiantian Liu, and Ladislav Kavan

Abstract—We propose Average Vector Field (AVF) integration for simulation of deformable solids in physics-based animation. Our method achieves exact energy conservation for the St. Venant-Kirchhoff material without any correction steps or extra parameters. Exact energy conservation implies that our resulting animations 1) cannot explode and 2) do not suffer from numerical damping, which are two common problems with previous numerical integration techniques. Our method produces lively motion even with large time steps as typically used in physics-based animation. Our implicit update rules can be formulated as a minimization problem and solved in a similar way as optimization-based backward Euler, with only a mild computing overhead. Our approach also supports damping and collision response models, making it easy to deploy in practical computer animation pipelines.

Index Terms—Animation, Three-Dimensional Graphics and Realism

1 Introduction

Energy conservation is a fundamental principle in nature, but it is rather elusive in numerical simulations. Some well known artifacts in physics-based computer animation can be attributed to lack of energy conservation: numerical "explosions" commonly observed with schemes such as explicit or symplectic Euler manifest themselves as non-physical energy increases, eventually rendering the simulation unstable. Other popular integrators, such as the backward Euler methods, have the opposite problem: non-physical energy decreases (numerical damping). Numerical errors can be reduced by lowering the time step and the necessity of step size control is a widely accepted fact in engineering and applied mathematics. For animators, however, integration time step is a "nuisance parameter", i.e., a parameter needed by the algorithm but not by the user. In physics-based animation used in applications such as film and games, the goal is to produce visually pleasing animations rather than to compute sufficiently accurate solutions of differential equations.

Animations also often take place in stylized virtual worlds, which is very different from numerical simulations in engineering, where we typically simulate real-world objects with known or measurable mechanical properties.

Even though animations do not strictly have to be physically realistic, violation of energy conservation often leads to degraded visual quality of motion. This is manifestly true with numerical explosions, which render the result useless. Therefore, to avoid instabilities, animators typically prefer dissipative integration schemes. Unfortunately, the numerical dissipation present in these integration schemes is not user controllable and hinders production of lively, vivid motion sequences.

Why is energy conservation difficult to achieve in numerical simulations? Visually rich motions such as the humorous jiggling of cartoon characters are characterized by large deformations, necessitating non-linear strain measures and often also non-linear material models [1]. The resulting elastic potentials are highly non-linear, making exact conservation of total mechanical energy (potential + kinetic) hard. In fact, early attempts to conserve energy via projections [2] or explicit energy-conservation constraints [3] resulted in the observation that enforcing energy conservation can lead to loss of accuracy [4]. Combined with negative theoretical results [5] (indicating that energy conservation is de facto incompatible with symplecticity), attempts to conserve energy fell out of favor. The late J. C. Simo was a pioneer of modern energy-conserving methods [6], which continue to be an active research topic in applied mathematics [7]. However, outside of few notable exceptions [8], the application of energy-conserving integrators in computer animation has not been extensively explored yet.

The key idea of the Average Vector Field (AVF) integrator [9] is best explained in comparison with classical integration schemes. The hard part of numerical integration is integrating the internal forces $\int_{t_n}^{t_{n+1}} \mathbf{f}(\mathbf{x}(t))dt$, where $\mathbf{f}$ is a force function and $\mathbf{x}(t)$ is the time-dependent trajectory (see Figure 1 left). The forward Euler method approximates this integral with a one point quadrature $h\mathbf{f}(\mathbf{x}_n)$, where $h = t_{n+1} - t_n$ is the time step. The backward Euler method uses instead $h\mathbf{f}(\mathbf{x}_{n+1})$, requiring an implicit solve process. The Newmark-beta method uses the trapezoidal rule $\frac{h}{2}(\mathbf{f}(\mathbf{x}_n) + \mathbf{f}(\mathbf{x}_{n+1}))$. It can be shown that the Newmark's scheme conserves energy if the force function $\mathbf{f}$ is linear but, unfortunately, energy conservation is lost with non-linear $\mathbf{f}$. The AVF integrator uses a more general formula:

$$h \int_0^1 \mathbf{f}((1-t)\mathbf{x}_n + t\mathbf{x}_{n+1})dt \tag{1}$$

- *Junior Rojas is with the School of Computing, University of Utah.*
  E-mail: jrojasdavalos@gmail.com
- *Tiantian Liu is with the Department of Computer and Information Science, University of Pennsylvania.*
  E-mail: ltt1598@gmail.com
- *Ladislav Kavan is with the School of Computing, University of Utah.*
  E-mail: ladislav.kavan@gmail.com

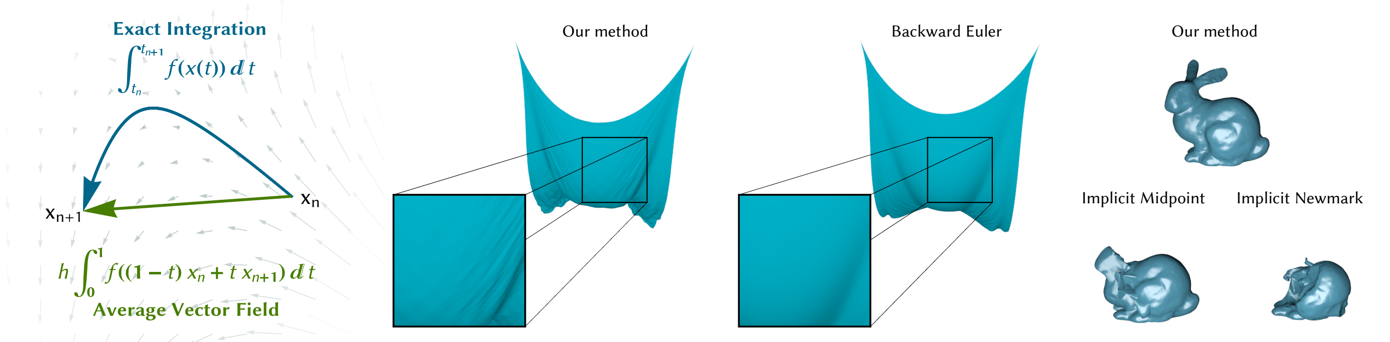

Fig. 1. The Average Vector Field (AVF) method approximates ground truth motion trajectories with straight lines (left). We propose an implicit integration scheme based on AVF which exactly conserves energy for St. Venant-Kirchhoff materials. Compared to backward Euler, our method does not introduce numerical damping and corresponding loss of details (middle). Compared to implicit midpoint and Newmark methods, our method avoids explosions even with large time steps and long simulation runs (right).

Indeed, for linear $\mathbf{f}$, the integral in Eq. 1 can be evaluated *exactly* using the trapezoidal rule $\frac{h}{2}(\mathbf{f}(\mathbf{x}_n) + \mathbf{f}(\mathbf{x}_{n+1}))$ and thus the AVF method reduces itself to the Newmark scheme. We remark that Eq. 1 is different from the ground truth solution $\int_{t_n}^{t_{n+1}} \mathbf{f}(\mathbf{x}(t))dt$ because the unknown trajectory $\mathbf{x}(t)$ is usually not a straight line. However, Eq. 1 copies the energy conservation property of the ground truth solution, which is guaranteed for conservative $\mathbf{f}$. This is true even for non-linear forces $\mathbf{f}$, as long as the integral in Eq. 1 is evaluated *exactly*.

We observe that this is possible for elastic forces $\mathbf{f}$ corresponding to St. Venant-Kirchhoff (StVK) materials. The StVK forces are degree three polynomials, which means that we can exactly evaluate Eq. 1 using Simpson's rule (three-point Newton-Cotes quadrature) using Eq. 2. This observation was also made before in [10], as a particular quadrature rule that can achieve energy preservation for quartic Hamiltonians.

$$\frac{h}{6}\left(\mathbf{f}(\mathbf{x}_n) + 4\mathbf{f}\left(\frac{\mathbf{x}_n + \mathbf{x}_{n+1}}{2}\right) + \mathbf{f}(\mathbf{x}_{n+1})\right) \tag{2}$$

Contributions. Our main contribution consists in showing that Eq. 2 lends itself to an implicit solve process which is analogous to the commonly used backward Euler integration. This is in stark contrast with other types of energy-conserving integrators, such as midpoint discrete gradient [11], which lead to much more difficult systems of non-linear equations that can potentially make a root finding procedure converge to invalid solutions (see Figure 2), especially with large time steps commonly used in computer graphics. Although previous work outside of the graphics literature, such as [10] and [12] pointed out that energy conservation in Hamiltonian systems can be achieved with AVF integration, we show that we can recast these non-linear equations into an optimization problem, which leads to a robust and practical simulator of deformable solids which exactly conserves the total energy of StVK materials for arbitrarily large time steps. Numerical advantages of solving optimization problems over non-linear root finding in the context of symplectic integration methods for computer animation were discussed in [13]. With the AVF integrator, the optimization-problem formulation is even more important because it enables us to guarantee exact energy conservation even if the solver converges to a local minimum. The energy-preservation property of our method implies that we avoid both "explosions" (unbounded energy increases) as well as non-physical energy dissipation (artificial energy decreases). This is achieved without introducing any additional parameters or correction steps, making our method very easy to use and implement. Its implementation is only slightly more complicated compared to classical implicit methods such as backward Euler or implicit Newmark.

2 Related Work

Numerical integration of ordinary differential equations dates back to the seminal work of Euler in the 18th century and continues to be an active area of research in applied mathematics and engineering. Differential equations appear in many areas of science: physics and astronomy, chemistry, biology, ecology, and economics. Each application domain has different requirements on the numerical solution procedures, which explains a rich variety of numerical integration methods. We focus on methods related to computer animation and refer the reader interested in the broader picture to books [14], [15], [16].

Physics-based animation requires fast computations but not necessarily highly accurate solutions, which lead pioneers in this area to using larger time steps and implicit time stepping methods [17], [18], [19]. The backward Euler method (BDF1) became particularly popular [20] in computer animation, despite its significant numerical damping. Its second order variant (BDF2) [21] is more accurate, but still contains numerical damping which cannot be controlled by the user. More recent real-time physics methods, such as Position Based Dynamics and related approaches [22], [23], [24], [25], [26], [27] continue to rely on the backward Euler formulas as the underlying integrator. A significant amount of recent work in physics-based animation focuses on speeding up the numerical solvers of implicit integrators, capitalizing on their formulation as minimization problems [28], [29], [30], [31], [32], [33]. Alternatively, it is possible to get more accurate (and less damped) results by carefully reducing the time step size of backward Euler [34] or by employing more advanced integrators [35], [36]. While these methods help, they do not completely eliminate numerical damping.

Another approach to avoid numerical dissipation and related deficiencies of numerical integration is by preserving the symplectic structure of mechanical systems [13], [37], [38], [39], [40], [41]. Even though symplectic schemes exhibit energy that oscillates about its correct value [16] (Chapter 9.8), these oscillations can be dramatic, rendering the resulting animation visually implausible. This is true even with implicit symplectic schemes such as implicit midpoint or Newmark, unless the time step is sufficiently small.

Most closely related to our work are energy-conserving integrators. Early approaches [2], [3] did not enjoy widespread use. A more recent proponent of energy (and momentum) conserving methods was J. C. Simo [6]. After Simo's premature death in 1994, his students and other experts further developed these ideas, culminating in discrete gradient methods [9], [11]. Specifically, the "midpoint discrete gradient" method had been applied to non-linear elastodynamics [42], achieving conservation of both energy and angular momenta. Unfortunately, with time steps typically used in computer animation, implicit solvers using midpoint discrete gradient are not robust, as we discuss in Section 3. Energy-conserving methods are relatively rare in the physics-based animation literature. One example is the energy-projection method proposed by Su et al. [8] which helps to improve visual quality, but does not always succeed in conserving energy, thus the simulation can still explode. This has been improved upon by Dinev et al. [43], who proposed to track the total energy and correct for instabilities of implicit midpoint by blending with backward Euler. Their method is fast and stable, but does not guarantee energy conservation, because in the worst case the integrator falls back to standard backward Euler. Our method avoids any correction steps or even monitoring of the total energy, because the AVF integrator conserves energy automatically.

However, our method exactly conserves energy only for polynomial material models, such as StVK. Even though this is undoubtedly a limitation, the StVK material has been shown to be effective in physics-based animation [44]. Specifically, the polynomial nature of the StVK material has been exploited to achieve dramatic acceleration of subspace simulations.

3 Method

According to Newton's laws of motion, the ground truth solution of advancing a mechanical system from time t_n to time t_{n+1} is given by:

$$\mathbf{x}^*_{n+1} = \mathbf{x}_n + \int_{t_n}^{t_{n+1}} \mathbf{v}(t)dt \quad (3)$$

$$\mathbf{v}^*_{n+1} = \mathbf{v}_n + \mathbf{M}^{-1}\int_{t_n}^{t_{n+1}} \mathbf{f}(\mathbf{x}(t))dt \quad (4)$$

where $\mathbf{x}_n \in \mathbb{R}^m, \mathbf{v}_n \in \mathbb{R}^m$ are positions and velocities in time t_n (the *known* previous state), $\mathbf{x}^*_{n+1} \in \mathbb{R}^m, \mathbf{v}^*_{n+1} \in \mathbb{R}^m$ is the ground truth solution in time t_{n+1}, $\mathbf{x} : \mathbb{R} \to \mathbb{R}^m$ and $\mathbf{v} : \mathbb{R} \to \mathbb{R}^m$ are the exact trajectories of the positions and velocities according to Newton's laws of motion, $\mathbf{f} : \mathbb{R}^m \to \mathbb{R}^m$ are position-dependent forces, and $\mathbf{M} \in \mathbb{R}^{m\times m}$ is the mass matrix. If we assume $\mathbf{f}$ to be conservative, there exists potential energy $E : \mathbb{R}^m \to \mathbb{R}$ such that $\mathbf{f} = -\nabla E$ and the total energy is conserved: $E(\mathbf{x}_{n+1}) + \frac{1}{2}\|\mathbf{v}_{n+1}\|^2_{\mathbf{M}} = E(\mathbf{x}_n)+\frac{1}{2}\|\mathbf{v}_n\|^2_{\mathbf{M}}$, where $\|.\|_{\mathbf{M}}$ denotes the mass-matrix norm (kinetic energy).

In practice, the integrals in formulas Eq. 3 and Eq. 4 must be approximated numerically, because the trajectories $\mathbf{x}(t)$ and $\mathbf{v}(t)$ are not known and $\mathbf{x}^*_{n+1}$ and $\mathbf{v}^*_{n+1}$ can only be approximated (except for simple cases such as linear $\mathbf{f}$). The Average Vector Field (AVF) method uses the following approximation:

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\int_0^1 [(1-t)\mathbf{v}_n + t\mathbf{v}_{n+1}]dt \quad (5)$$

$$\mathbf{v}_{n+1} = \mathbf{v}_n + h\mathbf{M}^{-1}\int_0^1 \mathbf{f}((1-t)\mathbf{x}_n + t\mathbf{x}_{n+1})dt \quad (6)$$

which corresponds to replacing the unknown trajectories $\mathbf{x}(t)$ and $\mathbf{v}(t)$ with straight lines. It can be shown that this approximation retains the exact energy conservation property, as long as the integrals in Eq. 5 and Eq. 6 are evaluated exactly [12]. The proof is very elegant and we include it in Appendix A. The integral in Eq. 5 can be exactly evaluated using the trapezoidal rule. To evaluate the integral in Eq. 6, we use a quadrature rule. For arbitrary polynomial potentials, we can use, for example, Newton-Cotes formulas to evaluate the integral exactly. More complex material models may also benefit from other quadrature rules such as Gauss quadrature rules. We provide more general formulas involving arbitrary quadrature rules in Appendix B. In particular, for forces which are polynomials of degree at most three (such as the StVK material model), the integral in Eq. 6 can be exactly evaluated using Simpson's rule (Eq. 2). This leads to:

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \frac{h}{2}(\mathbf{v}_n + \mathbf{v}_{n+1}) \quad (7)$$

$$\mathbf{v}_{n+1} = \mathbf{v}_n + \frac{h}{6}\mathbf{M}^{-1}\left[\mathbf{f}(\mathbf{x}_n) + 4\mathbf{f}\left(\frac{\mathbf{x}_n + \mathbf{x}_{n+1}}{2}\right) + \mathbf{f}(\mathbf{x}_{n+1})\right] \quad (8)$$

Note that these formulas are exactly equivalent to Eq. 5 and Eq. 6 (assuming the forces are degree three polynomials) and thus the total energy is still exactly conserved. Eq. 7 and Eq. 8 represent a system of non-linear equations which needs to be solved for the unknown next state $\mathbf{x}_{n+1}, \mathbf{v}_{n+1}$. To accomplish this, we start by substituting Eq. 8 into Eq. 7, eliminating $\mathbf{v}_{n+1}$:

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\mathbf{v}_n + \frac{h^2}{12}\mathbf{M}^{-1}\left[\mathbf{f}(\mathbf{x}_n) + 4\mathbf{f}\left(\frac{\mathbf{x}_n + \mathbf{x}_{n+1}}{2}\right) + \mathbf{f}(\mathbf{x}_{n+1})\right] \quad (9)$$

To simplify the notation, we denote the only remaining unknown as $\mathbf{x} := \mathbf{x}_{n+1}$ and group the known terms into $\mathbf{y}_n := \mathbf{x}_n + h\mathbf{v}_n + \frac{h^2}{12}\mathbf{M}^{-1}\mathbf{f}(\mathbf{x}_n)$. With this new notation, we can write Eq. 9 more concisely:

$$\mathbf{x} = \mathbf{y}_n + \frac{h^2}{3}\mathbf{M}^{-1}\mathbf{f}\left(\frac{\mathbf{x}_n + \mathbf{x}}{2}\right) + \frac{h^2}{12}\mathbf{M}^{-1}\mathbf{f}(\mathbf{x}) \quad (10)$$

Multiplying by $\mathbf{M}$ and re-arranging, we obtain:

$$F(x) = \mathbf{M}(\mathbf{x} - \mathbf{y}_n) - \frac{h^2}{3}\mathbf{f}\left(\frac{\mathbf{x}_n + \mathbf{x}}{2}\right) - \frac{h^2}{12}\mathbf{f}(\mathbf{x}) = 0 \quad (11)$$

This expression can be anti-differentiated to:

$$g(\mathbf{x}) = \frac{1}{2}\|\mathbf{x} - \mathbf{y}_n\|^2_{\mathbf{M}} + \frac{2h^2}{3}E\left(\frac{\mathbf{x}_n + \mathbf{x}}{2}\right) + \frac{h^2}{12}E(\mathbf{x}) \quad (12)$$

such that $\nabla g(\mathbf{x}) = 0$ is equivalent to Eq. 11. The corresponding optimization problem $\arg\min_{\mathbf{x}} g(\mathbf{x})$ is similar to the optimization form of backward Euler [45], [46]. Most importantly, our optimization problem can be robustly solved by Newton's method using the same definiteness fixes and line search strategies with the same initial guess: $\mathbf{x}_n + h\mathbf{v}_n$ as in the backward Euler case [45], [46], [47].

We also tried using $\mathbf{y}_n = \mathbf{x}_n + h\mathbf{v}_n + \frac{h^2}{12}\mathbf{M}^{-1}\mathbf{f}(\mathbf{x}_n)$ as an alternative initial guess. However, we found that $\mathbf{y}_n$ is typically a worse initial guess, leading to more Newton iterations than $\mathbf{x}_n + h\mathbf{v}_n$. We found this is because the forces $\mathbf{f}(\mathbf{x}_n)$ can be large and lead to a poor estimate of the solution.

This initial guess of $\mathbf{x}_{n+1}$ gets updated each Newton iteration until convergence. To compute a descent direction Δx to update the current estimate of $\mathbf{x}_{n+1}$, we have to solve the linear system $H\Delta\mathbf{x} = -\nabla g(x_{n+1})$. In an ideal case, H would be the Hessian matrix $\nabla^2 g(\mathbf{x}_{n+1})$, but since it is not necessarily positive definite, we have to apply definiteness fixes to get a valid descent direction.

In practice, to fix potential issues caused by indefiniteness, we repeatedly apply Tikhonov regularization: We begin by setting H to $\nabla^2 g(\mathbf{x}_{n+1})$ and compute a (sparse) Cholesky decomposition. If the decomposition is successful, we can use it to solve the linear system and get a valid descent direction that we then pass to a backtracking line search procedure to compute the new estimate of $\mathbf{x}_{n+1}$. If the decomposition is not successful, H is not positive definite, so we update H by adding an identity matrix scaled by s (initially 10^{-10}) and try the Cholesky decomposition again. We repeat this process until the decomposition is successful. Each time the decomposition fails, s gets multiplied by 10. Another possibility would be to implement the definiteness fix explained in Section 8 of [48], which is also compatible with our method.

When the solve is completed, we set $\mathbf{x}_{n+1} = \arg\min_{\mathbf{x}} g(\mathbf{x})$ and compute $\mathbf{v}_{n+1}$ in a straightforward way from Eq. 7. The key advantage over other implicit integrators such as backward Euler or Newmark is that our resulting state $\mathbf{x}_{n+1}, \mathbf{v}_{n+1}$ exactly conserves the total energy.

Damping. Our method does not impose any specific damping model and naturally supports damping models that can be expressed as polynomials. To show that user-defined damping can be added into the system, we implemented the standard Rayleigh model with damping forces defined as:

$$\mathbf{f}_{\text{damp}}(\mathbf{v}_{n+1}) = -\mathbf{D}\mathbf{v}_{n+1} \tag{13}$$

where $\mathbf{D}$ is a damping matrix defined as:

$$\mathbf{D} = \eta\mathbf{M} + \delta\mathbf{K} \tag{14}$$

where $\mathbf{M}$ is the mass matrix, $\mathbf{K}$ is stiffness matrix, and η and δ are damping coefficients controlled by the users. In our experiments we set $\eta := 0$ and vary only the δ parameter. Following Gast [46], we set $\mathbf{K} = \nabla^2 E(\mathbf{x}_n)$ with the usual definiteness fixes to ensure that $\mathbf{K}$ is positive definite.

To embed $\mathbf{f}_{\text{damp}}(\mathbf{v}_{n+1})$ in our framework, we first replace the velocity with positions using Eq. 7:

$$\mathbf{f}_{\text{damp}}(\mathbf{x}_{n+1}) = -\mathbf{D}\left(\frac{2(\mathbf{x}_{n+1} - \mathbf{x}_n)}{h} - \mathbf{v}_n\right) \tag{15}$$

This force is now only dependent on the next state $\mathbf{x}_{n+1}$ and therefore can be converted into an "energy" term:

$$E_{\text{damp}}(\mathbf{x}_{n+1}) = \frac{h}{4}\left\|\frac{2(\mathbf{x}_{n+1} - \mathbf{x}_n)}{h} - \mathbf{v}_n\right\|_{\mathbf{D}}^2 \tag{16}$$

Even though this term changes at each time step, it can be inserted into our objective Eq. 12 which corresponds to correct inclusion of the damping forces $\mathbf{f}_{\text{damp}}$. Energy conservation of course no longer applies, because damping forces are not conservative. A convenient feature of our approach is the AVF integrator does not need to be aware whether the forces are conservative or not.

Collisions. We use only a basic collision response method in our prototype implementation; for collision-dominated simulations, more advanced methods [49] are recommended. We implemented the standard repulsion-springs model [50]. In the beginning of each time step, we find all colliding vertices and for each colliding vertex, we find its projection to the closest collision-free surface point $\mathbf{x}_s$. We denote the surface normal at $\mathbf{x}_s$ as $\mathbf{n}_s$. For the subsequent integration step, we instantiate a temporary collision spring with potential $E_{\text{collision}}(\mathbf{x}) = k_{\text{collision}}((\mathbf{S}\mathbf{x} - \mathbf{x}_s)^\mathsf{T}\mathbf{n}_s)^2$, where $\mathbf{S}$ is a selector matrix that selects the inter-penetrated vertex. This collision spring will be removed when the vertex is no longer in collision and its velocity is pointing away from $\mathbf{n}_s$. Similarly to the damping energy term, we append $E_{\text{collision}}$ into our optimization problem Eq. 12.

4 Discussion

The AVF method falls in a more general category of numerical integrators known as discrete gradient methods [9]. In particular, the use of "midpoint discrete gradient" has been proposed for non-linear elastodynamics problems in mechanical engineering [42]. Unlike our AVF method, the midpoint discrete gradient exactly conserves angular momenta if the simulated mechanical system exhibits corresponding rotational symmetries (i.e., if the ground truth solution conserves angular momenta, according to Noether's theorem). This is an advantage over our AVF approach, which does not conserve angular momenta. Unfortunately, we found the midpoint discrete gradient has significant drawbacks. Specifically, the system of non-linear equations resulting from midpoint discrete gradient does not have the same structure as Eq. 7 and Eq. 8; the implicit rules are much more complicated and contain not only force terms $\mathbf{f}$, but also the potential E and additional non-linearities (see Eq. 3.5 in [9]). This presents two problems: 1) the non-linear equations are "more non-linear", e.g., for StVK materials E is a degree four polynomial; 2) the non-linear equations cannot be converted to an optimization form as in Eq. 12, because the Jacobian of these non-linear equations is not symmetric and thus cannot be written as a Hessian of some function (the Hessian is always symmetric).

Instead of solving a minimization problem as in our method, Gonzalez [42] uses an iterative root finding solver. We implemented this approach but found that with larger time steps, as typically used in computer animation, the root finder can often get stuck in a local minimum, failing to solve the midpoint discrete gradient update rule. We ran a simple test to demonstrate this: consider a single

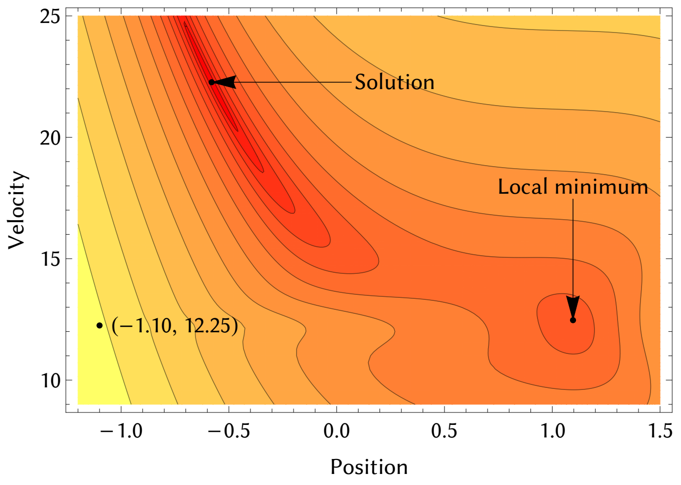

Fig. 2. Contour plot of the merit function for midpoint discrete gradient to update the state $(x_0, v_0) = (-1.10, 12.25)$ using a time step of $h = 0.003s$. The root solver fails to find the solution and gets stuck in a local minimum that does not conserve energy.

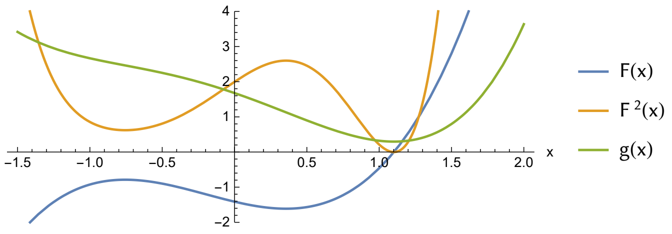

Fig. 3. Plot of different functions associated with the non-linear AVF equation of a single-particle simulation in 1D with potential energy $E(x) = 10x + 1500(x^2 - 1)^2$. The corresponding AVF equation $F(x)$ has only one root, but $F^2(x)$ has two local minima, which can make the solver fail to converge to the energy-preserving solution for certain initial values of x. In contrast, the only local minimum of $g(x)$ matches with the root of $F(x)$ and the solver converges to the energy-preserving solution regardless of the initialization.

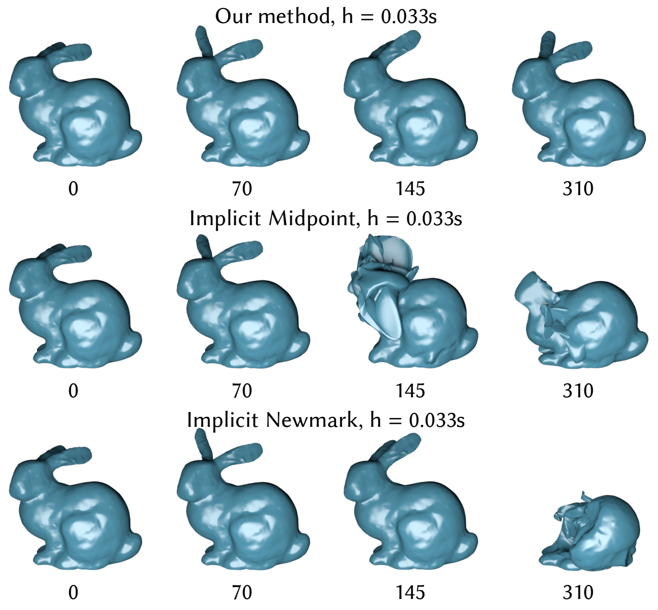

Fig. 4. Frames of the bunny simulation with different implicit integration methods.

particle of unit mass in 1D, with a quartic elastic potential $E(x) = 128x^4$. The root finder is solving a system of two non-linear equations $\mathbf{a}(x, v) = \mathbf{0}$, where x is 1D position and v is 1D velocity. As visualized in the contour plot of the merit function $\|\mathbf{a}\|^2$ in Figure 2, with a time step of $h = 0.033s$, the solver gets stuck in a local minimum. This problem can be avoided by using smaller time steps, e.g., for $h = 0.0066s$ the local minimum problem disappears. In the "Closing remarks", Gonzalez [42] suggests more efficient solve strategies as topic for future work but, to our knowledge, this remains an open problem.

Similarly, the difficulties that arise when using a root finder to solve the non-linear AVF equations can be seen in a simple 1D didactic example. Consider a single particle whose position is given by x, with mass = 1, affected by gravity (gravity constant = 10) and an StVK spring potential with one end attached to $x_{fixed} = 0$ (stiffness = 1500 and rest length = 1). The total potential energy is given by $E(x) = 10x + 1500(x^2 - 1)^2$. Consider an initial state given by $x_0 = 0.6$, $v_0 = -9$ and $h = 0.04$. Plotting the function $F(x)$ from Equation 11, we can see that the root or energy-preserving solution is close to $x = 1.0$ (Figure 3). If we try to solve this non-linear equation using Newton's method with a merit function $F^2(x)$, the solver will fail to converge for some initial values of x due to the local minimum between -1.5 and -0.5. Note that this problem is independent of line search strategy – convergence to the wrong local minimum can occur even with exact line search. In contrast, if we minimize the function $g(x)$ we introduced in Equation 12, the solver will converge to the root regardless of the initialization (Figure 3), since the local minimum between -1.5 and 0.5 is not present anymore. Not all local minima of $F^2(x)$ are roots of $F(x)$, but all local minima of $g(x)$ are roots of $F(x)$ because $\nabla_x g(x) = F(x)$.

Even though our method does not conserve angular momenta, in our experiments with rotationally symmetric potentials we observed that angular momenta oscillate near their constant values. The lack of angular momentum conservation in the AVF integrator does not introduce any visually disturbing artifacts, see Section 5. In general, we argue that energy conservation is a more critical quality of an integrator. For example, the implicit midpoint method conserves angular momenta, but energy can oscillate so dramatically the resulting animation is useless (it is interesting to note that despite visually implausible results, angular momenta are still exactly conserved).

5 Results

We demonstrate the properties of our method on different types of deformable objects. We simulate both volumetric solids (3D) discretized using linear finite elements [51] as well as thin shells (2D) discretized using StVK springs. While the potential of regular Hookean springs is $\frac{k}{2}(\|\mathbf{p}_1 - \mathbf{p}_2\| - r)^2$, our StVK springs use potential $\frac{k}{2}(\|\mathbf{p}_1 - \mathbf{p}_2\|^2 - r^2)^2$, avoiding the square root in $\|\mathbf{p}_1 - \mathbf{p}_2\|$. Therefore, the corresponding forces are degree three polynomials and we can guarantee exact energy conservation.

Bunny comparison. To demonstrate that our method is immune to both explosions and artificial damping, we

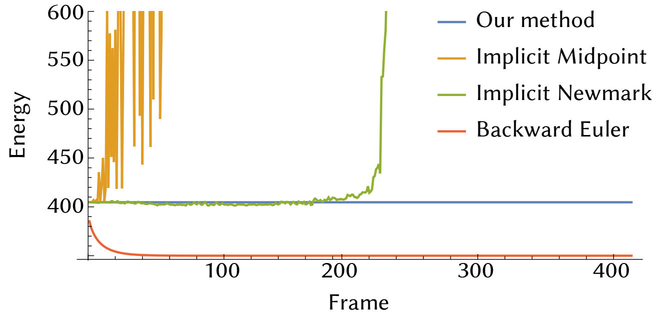

Fig. 5. Total energy vs. simulation frame of the bunny simulation.

simulate a bunny with an initial defomation corresponding to pulling its ears. We compare our method with implicit integrators commonly used in physics-based animation: backward Euler, implicit midpoint, and implicit Newmark (with the usual settings of $\beta = 0.25$ and $\gamma = 0.5$ corresponding to the trapezoidal rule).

As shown in Figure 4, both implicit midpoint and implicit Newmark explode. Our method conserves energy and thus remains stable regardless of the length of the simulation run. By "stability" we mean upper bound on the deformations and velocities of the simulated object, i.e., stability follows trivially from energy conservation.

Backward Euler does not cause explosions, but quickly slows down the motion due to numerical damping (please see the accompanying video). The corresponding energy plots can be seen in Figure 5. Note that implicit Newmark survives longer than implicit midpoint and even damps slightly before the eventual explosion.

Damping. Animators often want to add damping to achieve a desired visual effect. We demonstrate this on a simple hanging cloth example. With our method, we can add an arbitrary physics-based damping model which can be fully controlled by the user (as opposed to being a side-effect of the integration method). In Figure 6 we add Rayleigh damping (with $\eta = 0$ and $\delta = 0.01$) in order to suppress overly lively motion of the wrinkles. Thus, our method allows users to explore different amounts of physical damping in a straightforward way that does not require tweaking the time step. If we use other implicit methods such as implicit Midpoint or implicit Newmark, reducing the amount of damping might require using smaller time steps to avoid numerical instability. If we use backward Euler (even with zero Rayleigh damping), the motion is damped too much and more subtle wrinkles disappear.

Angular momentum. Our method does not conserve angular momentum, and to study the effect of this potential issue we designed a simulation of a dragon spinning around one single point in zero gravity. In this case, the potential is rotationally symmetric, thus angular momenta are conserved in the ground truth solution according to Noether's theorem. We verified this by running a simulation with symplectic Euler using very small time steps ($0.000033s$), dubbed "gold standard". We also compare against implicit midpoint, which conserves angular momentum. Unfortunately, implicit midpoint does not conserve energy, which manifests itself as explosions, see Figure 7. Even though our method does not conserve angular momentum, the angular momenta

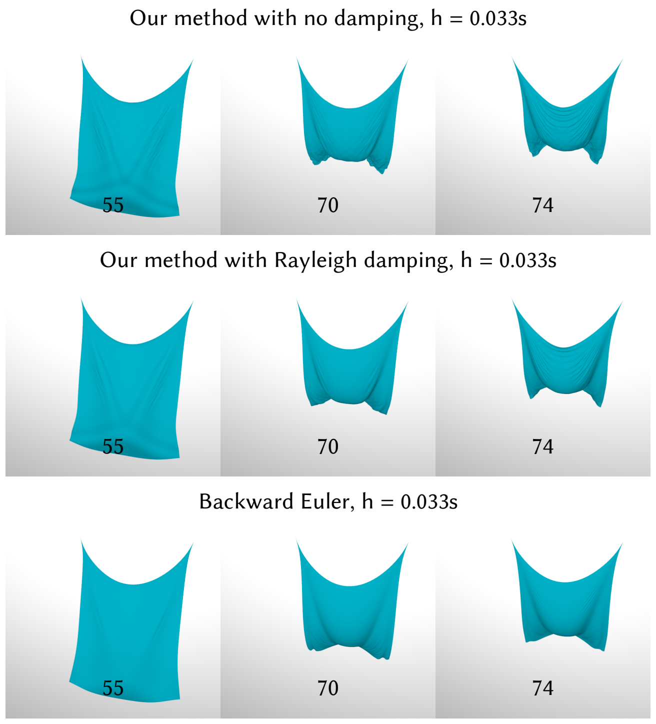
Our method with no damping, h = 0.033s
55
70
74
Our method with Rayleigh damping, h = 0.033s
55
70
74
Backward Euler, h = 0.033s
55
70
74

Fig. 6. We demonstrate adding Rayleigh damping to a cloth simulation. This suppresses too vivid motion of the wrinkles without excessive smoothing as produced by backward Euler (without damping).

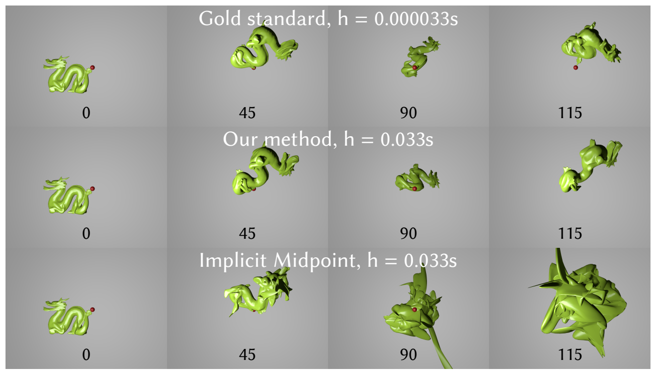

Fig. 7. Frames of the spinning dragon simulation with different integration methods.

produced by our method oscillate with a small amplitude close to their correct values, see Figure 8. The results produced with our method start to visibly diverge from the gold standard solution after about 100 frames of simulation. This is an expected consequence of our relatively large time step of $h = 0.033s$. Even though we do not make any claims of accuracy, we argue the two animations are qualitatively similar. We believe it is hard or even impossible to tell which method is more accurate with the naked eye, as both animations are perfectly visually plausible. This is an encouraging result from the point of view of physics-based animation where visual plausibility is a key consideration.

Accuracy. Even though we do not make any claims about accuracy of solving the underlying differential equations of

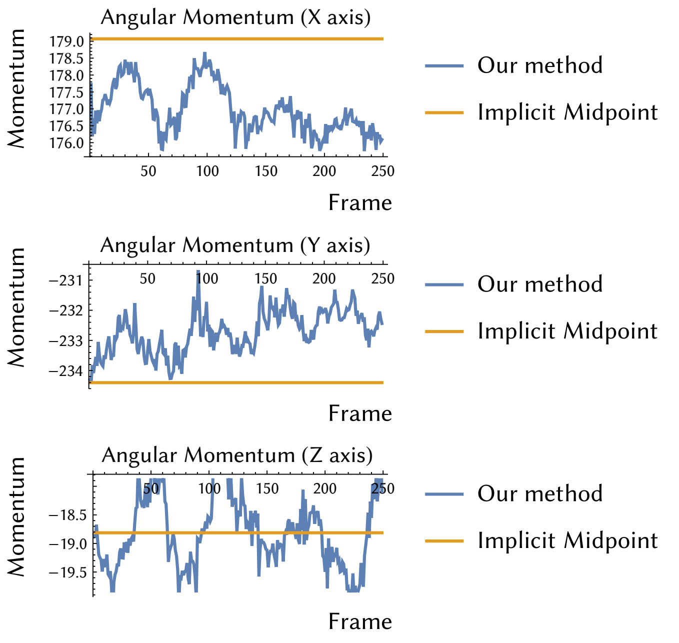

Fig. 8. Angular momentum per frame of the spinning dragon simulation.

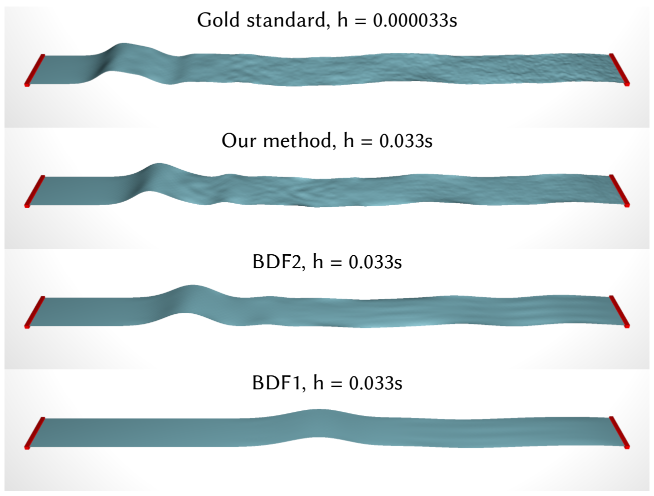

Fig. 9. Wave propagation in a thin sheet with no damping. While BDF1 and BDF2 are stable, they achieve so by introducing non-physical damping. Our method is stable and avoids numerical damping, which leads to wrinkles similar to those that emerge in the gold standard solution.

motion, we were curious about how the accuracy of our method compares to backward Euler formulas (BDF1 and BDF2). We tested this on a simulation of a wave propagating through a thin sheet. All integrators were run with $h = 0.033s$ except for the "gold standard" simulation which uses $h = 0.000033s$. We were pleased to find out that our method produces animation closest to the gold standard. The second best is BDF2, which is - not surprisingly - more accurate than BDF1. However, even BDF2 adds a significant amount of damping and smoothes away the small emerging waves apparent in the gold standard solution, see Figure 9.

Even in cases where damping is needed, our method gives more freedom to the users to add physics-based damping, because it does not introduce artificial damping as in BDF1 and BDF2.

Bathe's method. Another integrator which has been very recently introduced to physics-based animation [36] is "Bathe's method" [35]. Bathe's method composes trapezoidal rule (implicit Newmark) with BDF2, which results in a very good behavior in practice. However, energy is not conserved and with extreme initial conditions the simulation can still explode. While Bathe's method introduces only mild numerical damping, in some cases this can still visibly slow the simulation down, as we demonstrate on an example of a twisting elastic bar, see Figures 10 and 11.

Collisions. We designed an experiment to test our collision response model based on repulsion springs, see Figure 12. For this experiment, the stiffness of the repulsion springs was set to 200 and the material parameters of the model are summarized in Table 1 (Model: Ditto). Due to the non-polynomial nature of the repulsion springs, our approach can no longer guarantee energy conservation after collision response. Figure 13 shows an energy plot of the simulation of colliding Ditto (Figure 12) with and without damping. Note that in the non-damped case, the energy slowly drifts up, which is a limitation of our current collision

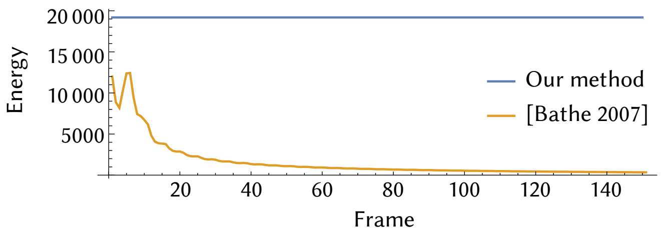

Fig. 10. Total energy vs. simulation frame of the twisting bar.

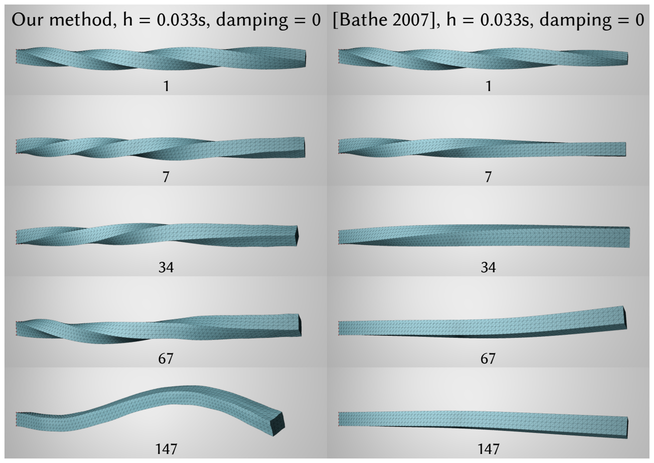
Our method, h = 0.033s, damping = 0
[Bathe 2007], h = 0.033s, damping = 0
1
7
34
67
147

Fig. 11. Example frames of the twisting bar simulation showing noticeable artificial damping introduced by Bathe's method.

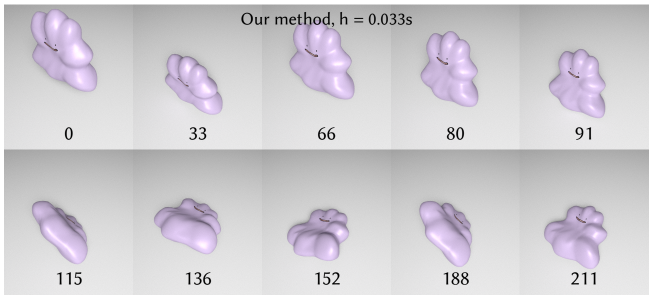

Fig. 12. Collisions example: a deformable character bouncing off the ground.

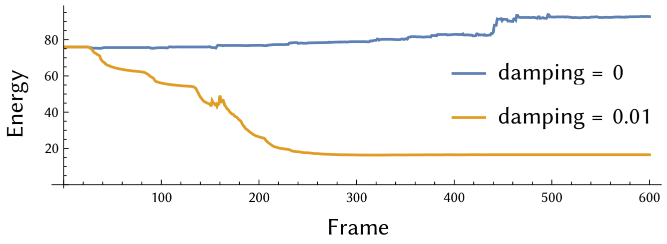

Fig. 13. Energy plot of the simulation setup shown in Figure 12, with and without damping. Due to the non-polynomial nature of the repulsion springs, our approach can no longer guarantee energy conservation after collision response.

resolution method; exact energy preservation even during contact events remains to be addressed in the future.

Runtime performance. In Table 1, we report the statistics of our experiments. In terms of runtime performance, our quadrature rule includes one more quadrature point than standard implicit methods (implicit midpoint / Newmark / backward Euler). This extra quadrature point translates itself to slightly more expensive gradient and Hessian evaluations. However, as seen in Table 1, the runtime overhead of these extra computations is small. This is because we did not construct all the required data structures (gradient and Hessian) at two quadrature points explicitly as written in Eq. 12. We instead computed the results using a single vector and matrix inside the same function. With this saving in memory allocation and access, we saw that the cost of each Newton iteration of our method is only about 20% more compared to other standard implicit methods.

How many iterations of Newton's method are required by each method? We compared the total number of iterations required until convergence in Figure 14. The dramatic increase in the number of iterations for implicit midpoint and Newmark is due to the fact these integrators started to explode. Prior to these explosions, the number of Newton iterations used by our method is very similar to the numbers required by implicit midpoint and Newmark. An opposite extreme is the very low number of iterations required by backward Euler. As can be seen in the accompanying video, these low iteration counts are easy to explain by the fact that backward Euler quickly stops the motion. The slow or even no motion makes the initial guess an excellent predictor of the solution and thus convergence is very rapid. Unfortunately, the resulting animation is rather boring as all of the motion quickly dies away.

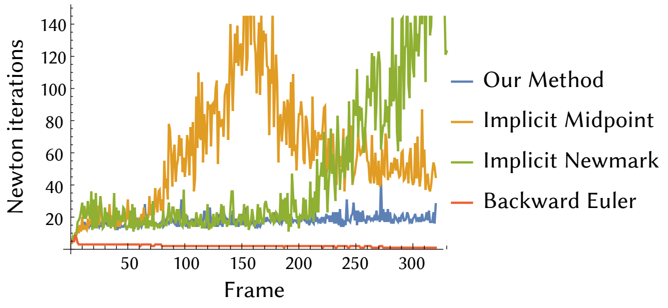

Fig. 14. Number of Newton iterations per frame for bunny simulation.

6 Limitations and Future Work

The main limitation of our approach is the exact numerical quadrature necessary to evaluate the average vector field integrals in Eq. 5 and Eq. 6. For this reason, we currently achieve exact energy conservation only for StVK materials using Simpson's rule. Higher order quadrature rules would allow us, for example, to extend the basic StVK material with a compression resistance term to mitigate element inversions as explained in [52]. Our proposed AVF method with Simpson's rule for force integration could be of course applied to arbitrary elastic models, however, the quadrature in Eq. 5 and Eq. 6 will no longer be exact and the exact energy conservation property is lost, as we showed in our simulations involving collisions with repulsion springs with non-polynomial potentials (Figure 13). Our preliminary tests indicate that the AVF integrator still produces high quality results, but more research is necessary. In the future, it may be possible to devise quadrature rules for other material models popular in physics-based animation, such as the corotated model or even more general models based on principal stretches [1]. Without switching to different material models, subspace StVK integration [44] is another very interesting area to explore using AVF integration. Our method also exposed the need for accurate modeling of viscoelasticity. Even though the energy conservation property of our method guarantees unconditional stability even without any damping, real-world mechanical systems always contain some amount of dissipation, typically involving transfer of mechanical energy into heat. Accurate modeling of damping is somewhat moot when using integrators that already contain large amounts of numerical damping. Our proposed method is free of numerical damping and thus presents motivation to investigate more realistic, physically-based or artist-driven damping models as very recently discussed by [36].

7 Conclusion

We have shown that Average Vector Field (AVF) integration naturally lends itself to practical simulation problems in physics-based animation. With St. Venant-Kirchhoff material models, the AVF integrator exactly conserves energy without any extra effort. This works robustly even with large time steps and - perhaps most importantly - the resulting numerical solver is only slightly more complicated than previous implicit integrators such as backward Euler methods or implicit midpoint / Newmark. Compared to these previous

Model	#Verts.	#Elems.	Discret.	Total mass	μ	λ	Time per Newton iteration (ms)			
							Our method	Impl. midpoint	Impl. Newmark	Backw. Euler
Bunny	1839	5891	Tets	10	500	10	193.08	172.36	188.47	145.20
Dragon	1000	3065	Tets	1	400	10	91.48	77.57	63.63	62.14
Thin Sheet	5061	24300	Springs	1	200	100	201.37	219.82	218.45	200.21
Cloth	6400	31363	Springs	1	20	10	949.16	951.72	905.02	829.65
Ditto	157	415	Tets	1	35	20	39.95	22.24	26.70	21.91
Twisting bar	1314	3440	Tets	1	1000	500	102.89	82.63	79.17	84.09

TABLE 1
Summary of parameters used in our simulation examples and average time per Newton iteration for different implicit integration methods. For tetrahedral meshes, μ and λ refer to the Lamé parameters of the StVK material. For spring-based models, they refer to the stiffnesses of stretching and bending springs.

integrators, the exact energy conservation property implies that our simulations avoid both "explosions" as well as numerical damping.

Acknowledgments

We would like to thank Bernhard Thomaszewski and Eftychios Sifakis for the valuable discussions, Petr Kadlecek for his help with rendering and Jing Li for her help generating our simulation examples. This material is based upon work supported by the National Science Foundation under Grant Numbers IIS-1617172 and IIS-1622360. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. We also gratefully acknowledge the support of Activision.

References

[1] H. Xu, F. Sin, Y. Zhu, and J. Barbic, "Nonlinear material design using principal stretches," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, p. 75, 2015.

[2] R. A. LaBudde and D. Greenspan, "Energy and momentum conserving methods of arbitrary order for the numerical integration of equations of motion," *Numerische Mathematik*, vol. 25, no. 4, pp. 323–346, 1975.

[3] T. J. Hughes, T. K. Caughey, and W. K. Liu, "Finite-element methods for nonlinear elastodynamics which conserve energy," *Journal of Applied Mechanics*, vol. 45, no. 2, pp. 366–370, 1978.

[4] E. Hairer, *Long-time energy conservation of numerical integrators*, ser. Foundations of computational mathematics, Santander 2005. Cambridge: Cambridge University Press, 2006, pp. 162–180, id: unige:12115. [Online]. Available: https://archive-ouverte.unige.ch/unige:12115

[5] Z. Ge and J. E. Marsden, "Lie-poisson hamilton-jacobi theory and lie-poisson integrators," *Physics Letters A*, vol. 133, no. 3, pp. 134–139, 1988.

[6] J. C. Simo, N. Tarnow, and K. Wong, "Exact energy-momentum conserving algorithms and symplectic schemes for nonlinear dynamics," *Computer methods in applied mechanics and engineering*, vol. 100, no. 1, pp. 63–116, 1992.

[7] L. Brugnano and F. Iavernaro, *Line Integral Methods for Conservative Problems*, ser. Chapman & Hall/CRC Monographs and Research Notes in Mathematics. CRC Press, 2016. [Online]. Available: https://books.google.com/books?id=5R6vCgAAQBAJ

[8] J. Su, R. Sheth, and R. Fedkiw, "Energy conservation for the simulation of deformable bodies," *TVCG*, vol. 19, no. 2, pp. 189–200, 2013.

[9] R. I. McLachlan, R. Quispel, and N. Robidoux, "Geometric integration using discrete gradients," *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 357, no. 1754, pp. 1021–1045, 1999.

[10] Celledoni, Elena, McLachlan, Robert I., McLaren, David I., Owren, Brynjulf, Reinout W. Quispel, G., and Wright, William M., "Energy-preserving runge-kutta methods," *ESAIM: M2AN*, vol. 43, no. 4, pp. 645–649, 2009. [Online]. Available: https://doi.org/10.1051/m2an/2009020

[11] O. Gonzalez, "Time integration and discrete hamiltonian systems," *Journal of Nonlinear Science*, vol. 6, no. 5, pp. 449–467, 1996.

[12] R. Quispel and D. McLaren, "A new class of energy-preserving numerical integration methods," *Journal of Physics A: Mathematical and Theoretical*, vol. 41, 2008.

[13] L. Kharevych, W. Yang, Y. Tong, E. Kanso, J. E. Marsden, P. Schröder, and M. Desbrun, "Geometric, variational integrators for computer animation," 2006, pp. 43–51.

[14] A. Iserles, *A first course in the numerical analysis of differential equations*. Cambridge University Press, 2009.

[15] U. M. Ascher and L. R. Petzold, *Computer methods for ordinary differential equations and differential-algebraic equations*, 1998, vol. 61.

[16] E. Hairer, C. Lubich, and G. Wanner, *Geometric numerical integration: structure-preserving algorithms for ordinary differential equations*, 2006, vol. 31.

[17] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer, "Elastically deformable models," in *Computer Graphics (Proceedings of SIGGRAPH)*, vol. 21, no. 4, 1987, pp. 205–214.

[18] D. Terzopoulos and K. Fleischer, "Deformable models," *The Visual Computer*, vol. 4, no. 6, pp. 306–331, 1988.

[19] D. Terzopoulos and K. Fleischer, "Modeling inelastic deformation: viscolelasticity, plasticity, fracture," in *Computer Graphics (Proceedings of SIGGRAPH)*, vol. 22, no. 4, 1988, pp. 269–278.

[20] D. Baraff and A. Witkin, "Large steps in cloth simulation," in *Proc. of ACM SIGGRAPH*, 1998, pp. 43–54.

[21] K.-J. Choi and H.-S. Ko, "Stable but responsive cloth," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 604–611, 2002.

[22] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff, "Position based dynamics," *Journal of Visual Communication and Image Representation*, vol. 18, no. 2, pp. 109–118, 2007.

[23] M. Müller, "Hierarchical Position Based Dynamics," in *Workshop in Virtual Reality Interactions and Physical Simulation "VRIPHYS" (2008)*, 2008, pp. 1–10.

[24] M. Müller, N. Chentanez, T.-Y. Kim, and M. Macklin, "Strain based dynamics," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '14. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2014, pp. 149–157. [Online]. Available: http://dl.acm.org/citation.cfm?id=2849517.2849542

[25] M. Macklin and M. Müller, "Position based fluids," *ACM Trans. Graph.*, vol. 32, no. 4, p. 104, 2013.

[26] M. Macklin, M. Müller, N. Chentanez, and T.-Y. Kim, "Unified particle physics for real-time applications," *ACM Trans. Graph.*, vol. 33, no. 4, p. 153, 2014.

[27] M. Macklin, M. Müller, and N. Chentanez, "Xpbd: position-based simulation of compliant constrained dynamics," in *Proc. of Motion in Games*, 2016, pp. 49–54.

[28] T. Liu, A. W. Bargteil, J. F. O'Brien, and L. Kavan, "Fast simulation of mass-spring systems," *ACM Trans. Graph.*, vol. 32, no. 6, pp. 209:1–7, 2013.

[29] S. Bouaziz, S. Martin, T. Liu, L. Kavan, and M. Pauly, "Projective dynamics: fusing constraint projections for fast simulation," *ACM Trans. Graph.*, vol. 33, no. 4, p. 154, 2014.

[30] H. Wang and Y. Yang, "Descent methods for elastic body simulation on the gpu," *ACM Trans. Graph.*, vol. 35, no. 6, p. 212, 2016.

[31] R. Narain, M. Overby, and G. E. Brown, "ADMM $\supseteq$ projective dynamics: Fast simulation of general constitutive models," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '16. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2016, pp. 21–28. [Online]. Available: http://dl.acm.org/citation.cfm?id=2982818.2982822

[32] M. Fratarcangeli, V. Tibaldo, and F. Pellacini, "Vivace: a practical gauss-seidel method for stable soft body dynamics," *ACM Trans. Graph.*, vol. 35, no. 6, p. 214, 2016.

[33] T. Liu, S. Bouaziz, and L. Kavan, "Quasi-newton methods for real-time simulation of hyperelastic materials," *ACM Trans. Graph.*, vol. 36, no. 3, p. 23, 2017.

[34] D. Zhao, Y. Li, and J. Barbic, "Asynchronous implicit backward euler integration," *Proc. EG/ACM Symp. Computer Animation*, 2016.

[35] K.-J. Bathe, "Conserving energy and momentum in nonlinear dynamics: a simple implicit time integration scheme," *Computers & structures*, vol. 85, no. 7, pp. 437–445, 2007.

[36] H. Xu and J. Barbic, "Example-based damping design," *ACM Trans. Graph.*, vol. 36, no. 4 (to appear), 2017. [Online]. Available: http://run.usc.edu/exampleBasedDamping/

[37] A. Stern and M. Desbrun, "Discrete geometric mechanics for variational time integrators," in *ACM SIGGRAPH Courses*, 2006, pp. 75–80.

[38] A. Stern and E. Grinspun, "Implicit-explicit variational integration of highly oscillatory problems," *Multiscale Modeling & Simulation*, vol. 7, no. 4, pp. 1779–1794, 2009.

[39] B. Fierz, J. Spillmann, and M. Harders, "Element-wise mixed implicit-explicit integration for stable dynamic simulation of deformable objects," 2011, pp. 257–266.

[40] D. L. Michels, G. A. Sobottka, and A. G. Weber, "Exponential integrators for stiff elastodynamic problems," *ACM Trans. Graph.*, vol. 33, no. 1, p. 7, 2014.

[41] D. L. Michels and J. P. T. Mueller, "Discrete computational mechanics for stiff phenomena," in *SIGGRAPH ASIA Courses*, 2016, p. 13.

[42] O. Gonzalez, "Exact energy and momentum conserving algorithms for general models in nonlinear elasticity," *Computer Methods in Applied Mechanics and Engineering*, vol. 190, no. 13, pp. 1763–1783, 2000.

[43] D. Dinev, T. Liu, and L. Kavan, "Stabilizing integrators for real-time physics," *ACM Trans. Graph.*, vol. (currently under review), 2016. [Online]. Available: https://www.cs.utah.edu/~ddinev/

[44] J. Barbic and D. L. James, "Real-time subspace integration for st. venant-kirchhoff deformable models," in *ACM Trans. Graph.*, vol. 24, no. 3, 2005, pp. 982–990.

[45] S. Martin, B. Thomaszewski, E. Grinspun, and M. Gross, "Example-based elastic materials," in *ACM Trans. Graph.*, vol. 30, no. 4, 2011, p. 72.

[46] T. F. Gast, C. Schroeder, A. Stomakhin, C. Jiang, and J. M. Teran, "Optimization integrator for large time steps," *TVCG*, vol. 21, no. 10, pp. 1103–1115, 2015.

[47] J. Nocedal and S. Wright, *Numerical optimization*, 2006.

[48] J. Teran, E. Sifakis, G. Irving, and R. Fedkiw, "Robust Quasistatic Finite Elements and Flesh Simulation," in *Symposium on Computer Animation*, D. Terzopoulos, V. Zordan, K. Anjyo, and P. Faloutsos, Eds. The Eurographics Association, 2005.

[49] D. Harmon, E. Vouga, B. Smith, R. Tamstorf, and E. Grinspun, "Asynchronous contact mechanics," in *ACM Trans. Graph.*, vol. 28, no. 3, 2009, p. 87.

[50] A. McAdams, Y. Zhu, A. Selle, M. Empey, R. Tamstorf, J. Teran, and E. Sifakis, "Efficient elasticity for character skinning with contact and collisions," in *ACM Trans. Graph.*, vol. 30, no. 4, 2011, p. 37.

[51] E. Sifakis and J. Barbic, "Fem simulation of 3d deformable solids: a practitioner's guide to theory, discretization and model reduction," in *ACM SIGGRAPH Courses*, 2012, p. 20.

[52] R. Kikuuwe, H. Tabuchi, and M. Yamamoto, "An edge-based computationally efficient formulation of saint venant-kirchhoff tetrahedral finite elements," *ACM Trans. Graph.*, vol. 28, no. 1, pp. 8:1–8:13, Feb. 2009. [Online]. Available: http://doi.acm.org/10.1145/1477926.1477934

Junior Rojas is currently a Ph.D. student at the University of Utah, working on physics-based simulation with professor Ladislav Kavan. He was previously a research programmer at Wolfram Research and received his Bachelor of Science from Pontifical Catholic University of Peru.

Tiantian Liu is currently a Ph.D. candidate at the University of Pennsylvania, working with professor Ladislav Kavan. The research focus of his work is mainly on physically based simulation and fast numerical methods for optimization. He previously received his Master of Engineering degree from University of Pennsylvania, and Bachelor of Engineering degree from Zhejiang University.

Ladislav Kavan is an assistant professor of computer science at the University of Utah. Prior to joining Utah, he was an assistant professor at the University of Pennsylvania and research scientist at Disney Interactive Studios. Ladislav's research focuses on interactive computer graphics, physics-based animation, and geometry processing. His goal is to combine computer graphics with biomechanics and medicine. Ladislav is a member of the ACM SIGGRAPH community and serves as an Associate Editor for ACM Transactions on Graphics.

APPENDIX A

In this Appendix we include a proof (adapted from [12]) that the Average Vector Field (AVF) method described in Eq. 5 and Eq. 6 exactly preserves the total energy of the system.

We first define a state vector $\mathbf{q} \in \mathbb{R}^{2m}$ that contains both the positions and the velocities of a system:

$$\mathbf{q} = \begin{bmatrix} \mathbf{x} \\ \mathbf{v} \end{bmatrix} \tag{17}$$

The total energy (Hamiltonian) $H(\mathbf{q})$ is defined as:

$$H(\mathbf{q}) = E(\mathbf{x}) + \frac{1}{2}\|\mathbf{v}\|^2_{\mathbf{M}} \tag{18}$$

Our goal is to prove the total energy does not change after the time integration step according to Eq. 5 and Eq. 6, i.e., $H(\mathbf{q}_n) = H(\mathbf{q}_{n+1})$. Differentiating Eq. 18, we can rewrite the force and velocity terms as:

$$\mathbf{f}((1-t)\mathbf{x}_n + t\mathbf{x}_{n+1}) = -\nabla_{\mathbf{x}} H((1-t)\mathbf{q}_n + t\mathbf{q}_{n+1}) \tag{19}$$

$$(1-t)\mathbf{v}_n + t\mathbf{v}_{n+1} = \mathbf{M}^{-1}\nabla_{\mathbf{v}} H((1-t)\mathbf{q}_n + t\mathbf{q}_{n+1}) \tag{20}$$

Substituting Eq. 19 and Eq. 20 into Eq. 5 and Eq. 6 leads to:

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\mathbf{M}^{-1}\int_0^1 \nabla_{\mathbf{v}} H((1-t)\mathbf{q}_n + t\mathbf{q}_{n+1})dt \tag{21}$$

$$\mathbf{v}_{n+1} = \mathbf{v}_n - h\mathbf{M}^{-1}\int_0^1 \nabla_{\mathbf{x}} H((1-t)\mathbf{q}_n + t\mathbf{q}_{n+1})dt \tag{22}$$

which can be combined into one equation:

$$\mathbf{q}_{n+1} - \mathbf{q}_n = \mathbf{S}\int_0^1 \nabla_{\mathbf{q}} H((1-t)\mathbf{q}_n + t\mathbf{q}_{n+1})dt \tag{23}$$

where $\mathbf{S}$ is a skew-symmetric matrix defined as:

$$\mathbf{S} = \begin{bmatrix} \mathbf{0} & \mathbf{M}^{-1} \\ -\mathbf{M}^{-1} & \mathbf{0} \end{bmatrix} \tag{24}$$

Now let us take the dot product of both sides of Eq. 23 with $\int_0^1 \nabla_{\mathbf{q}} H((1-t)\mathbf{q}_n + t\mathbf{q}_{n+1})dt$:

$$(\mathbf{q}_{n+1} - \mathbf{q}_n)^{\mathsf{T}}\left(\int_0^1 \nabla_{\mathbf{q}} H((1-t)\mathbf{q}_n + t\mathbf{q}_{n+1})dt\right) = 0 \tag{25}$$

where the right hand side vanishes because $\mathbf{v}^{\mathsf{T}}\mathbf{S}\mathbf{v} = 0$ for any vector $\mathbf{v}$ due to the skew-symmetry of $\mathbf{S}$. Because $\mathbf{q}_{n+1} - \mathbf{q}_n$ is independent of t, we can move it inside the integral:

$$\int_0^1 (\mathbf{q}_{n+1} - \mathbf{q}_n)^T(\nabla_{\mathbf{q}} H((1-t)\mathbf{q}_n + t\mathbf{q}_{n+1}))dt = 0 \tag{26}$$

We can then apply the chain rule to get:

$$\int_0^1 \frac{d}{dt} H((1-t)\mathbf{q}_n + t\mathbf{q}_{n+1})dt = 0 \tag{27}$$

and using the first fundamental theorem of calculus, we conclude that:

$$H(\mathbf{q}_{n+1}) - H(\mathbf{q}_n) = 0 \tag{28}$$

which completes the proof.

APPENDIX B

Our results have focused on the StVK material and Simpson's rule for numerical integration. In this appendix, we derive a general formula for the objective function $g(x)$, considering a more general quadrature rule for numerical integration.

Recall that the AVF integrator can be formulated with update rules for velocity and position given by Eq. 29 and Eq. 30 .

$$v_1 = v_0 - M^{-1}\int_0^1 \nabla E((1-t)x_0 + tx_1)dt \tag{29}$$

$$x_1 = x_0 + \frac{h}{2}(v_0 + v_1) \tag{30}$$

We can replace v_1 from Eq. 29 into Eq. 30 to get Eq. 31

$$x_1 = x_0 + hv_0 - \frac{hM^{-1}}{2}\int_0^1 \nabla E((1-t)x_0 + tx_1)dt \tag{31}$$

At this point, we will replace the integral in Eq. 31 with a quadrature rule. A quadrature is defined by a set of weights w_i and abscissae t_i. We will adopt a format similar to that used in Gauss quadrature. That is, the weights w_i must sum to 2 and abscissae t_i are numbers between -1 and 1, which get mapped to some value in $[x_0, x_1]$. We define a general formula for a quadrature term Q_i in Eq. 32.

$$Q_i = w_i \nabla E\Big(\frac{(x_0 + x_1)}{2} + \frac{(x_1 - x_0)}{2}t_i\Big) \tag{32}$$

The integral from Eq. 31 will be replaced with the formula given in Eq. 33, according to a certain quadrature rule.

$$\int_0^1 \nabla E((1-t)x_0 + tx_1)dt = \frac{h}{2}\sum_i Q_i \tag{33}$$

After replacing the integral in Eq. 31 with the quadrature formula introduced in Eq. 33, we get Eq. 34.

$$x_1 = x_0 + hv_0 - \frac{h^2M^{-1}}{4}\sum_i Q_i \tag{34}$$

Certain quadratures might include x_0 as a quadrature point, which corresponds to $t_0 = -1$. This is the case, for example, in Simpson's rule, and in other Newton-Cotes quadrature rules. Since this quadrature term is constant with respect to x_1, it is convenient to take it out of the sum for differentiation. Including this point explicitly in this manner, we get Eq. 35.

$$x_1 = x_0 + hv_0 - \frac{h^2M^{-1}}{4}Q_{i \mid t_i=-1} - \frac{h^2M^{-1}}{4}\sum_{i \mid t_i \neq -1} Q_i \tag{35}$$

Then, we introduce y to capture the part of the expression that is constant with respect to x_1 as shown in Eq. 36. Note that if the chosen quadrature does not use the abscissa $t_i = -1$, the last term of this expression is zero.

$$y = x_0 + hv_0 - \frac{h^2M^{-1}}{4}Q_{i \mid t_i=-1} \tag{36}$$

Now the system equation can be written more concisely, as shown in Eq. 37.

$$2M(x_1 - y) + \frac{h^2}{2} \sum_{i \mid t_i \neq -1} Q_i = 0 \tag{37}$$

To robustly solve Eq. 37, we will anti-differentiate it with respect to x_1 and recast the problem as an optimization problem. The anti-derivative E_i of each quadrature term Q_i is given in Eq. 38.

$$E_i = \frac{2w_i}{1+t_i} E\left(\frac{(x_0 + x_1)}{2} + \frac{(x_1 - x_0)}{2} t_i \right) \tag{38}$$

Now we can anti-differentiate the system equation (Eq. 37) to get the a general form for $g(x)$ (originally introduced in Eq. 12 for Simpson's rule) using arbitrary quadrature rules (Eq. 39).

$$g(x) = (x_1 - y)^T M(x_1 - y) + \frac{h^2}{2} \sum_{i \mid t_i \neq -1} E_i \tag{39}$$

Additionally, the formulas for the gradient and Hessian of $g(x)$ are shown in Eq. 40 and Eq. 41.

$$\nabla g(x) = 2M(x_1 - y) + \frac{h^2}{2} \sum_{i \mid t_i \neq -1} Q_i \tag{40}$$

$$\nabla^2 g(x) = 2M + \frac{h^2}{2} \sum_{i \mid t_i \neq -1} H_i \tag{41}$$

$$H_i = \frac{(1+t_i)w_i}{2} \nabla^2 E\left(\frac{(x_0 + x_1)}{2} + \frac{(x_1 - x_0)}{2} t_i \right) \tag{42}$$