*Volume 37 (2018), Number 8*

# Adaptive Time-Stepping with Linearized Force Error Estimation

paper1001

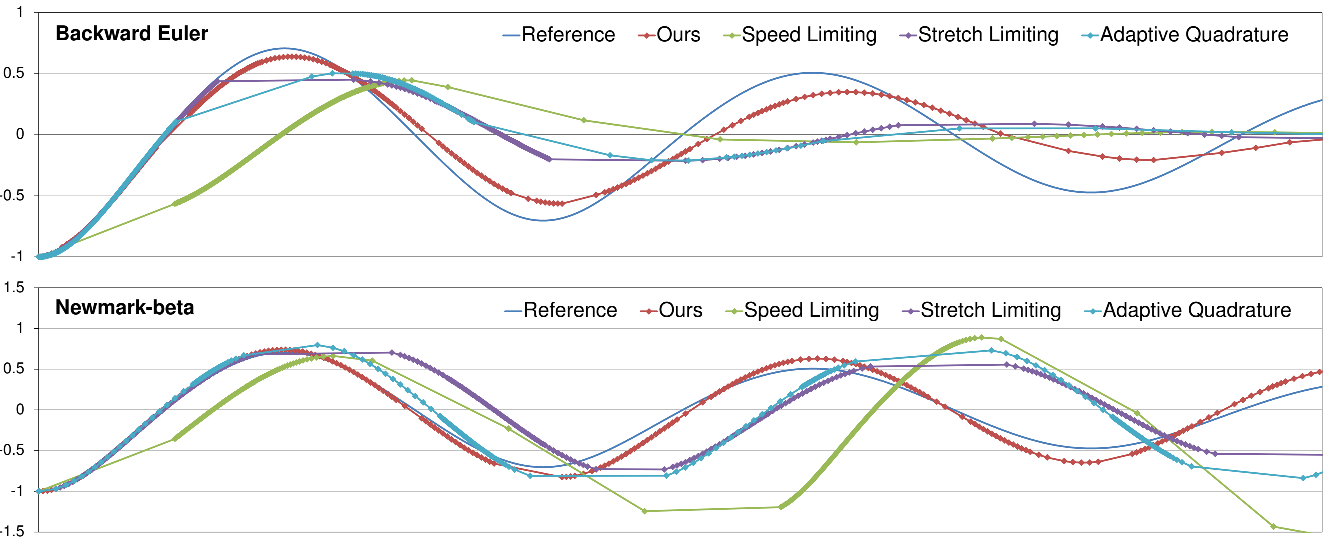

**Figure 1:** *A simple test with a single mass attached to a zero rest-length spring, comparing different adaptive time stepping methods with backward Euler and Newmark-beta integrations. The parameters of all methods are adjusted such that they take approximately the same number of time steps. Note that our method clearly produces closer results to the reference solution.*

**Abstract**

*We introduce a novel adaptive time stepping technique for implicit integration that allows the integrator to take orders of magnitude larger time steps as compared to a stable fixed time step size counterpart. We formulate our adaptive time stepping approach based on the error metric introduced by the linearization of the internal force term that arises during solving linear system of implicit integrator. Our adaptive strategy does not calculate time step size directly from the current system states such as velocity magnitude or kinetic energy, thereby allows taking large time step while providing maximizing stability and accuracy. We show that with a similar amount of computational budget, our method produces superior accuracy in comparison with prior work.*

*(see http://www.acm.org/about/class/class/2012)*

**CCS Concepts**

•***Computing methodologies*** $\rightarrow$ ***Physical simulation;***

## 1. Introduction

Computational performance is an important concern in most computer graphics applications. In physically-based simulations, implicit integration techniques can deliver the desired performance for complex simulation scenarios when they can provide stable simulations with large-enough time-step sizes. However, typical implicit integration systems used in computer graphics often do not guarantee unconditional stability. Therefore, a safe time-step size must be established to keep the simulation stable. Obviously, computational performance is directly correlated with the time-step size. Therefore, the ideal time-step size is typically the largest one that leads to a stable simulation with a desirable level of accuracy.

When using fixed-size time-steps, the most unstable step in the entire simulation scenario limits the time-step size. Since it can be difficult to determine this limit prior to simulation, in practice a conservative time-step size is used, thereby providing limit performance benefits from implicit integration.

A typical solution is dynamically adjusting the time-step size via adaptive time-stepping. There is a large body of work in computational science and engineering on adaptive time-stepping; however, these methods primarily concentrate on the accuracy of simulations. The adaptive time-stepping approaches in computer graphics primarily focus on simulation stability and performance, but they are usually based on ad-hoc formulations and simple heuristics with unintuitive parameters. Unfortunately, these formulations and heuristics do not always correlate with simulation stability or accuracy. Thus, they are typically used with conservative parameters that do not fully take advantage of the potential performance benefits of adaptive time-stepping.

In this paper, we introduce a principled adaptive time-stepping method for implicit systems formed with linearized force formulations [BW98] that are commonplace in computer graphics. We recognize that a substantial portion of simulation error in such systems is related to linearized force estimation. Based on this observation, our adaptive time-stepping formulation adjusts the time-step size according to the error in force estimation. This error can be calculated efficiently with minimal additional computation. We convert the error in force estimation to error in position, so that the user can simply provide an acceptable error tolerance in distance for controlling our method.

We explain how our formulation can be used with different numerical integration techniques. In particular, we evaluate our adaptive time-stepping approach using backward Euler [BW98] and and Newmark-beta [New59] integration schemes. Backward Euler is arguably the most popular numerical scheme used in computer graphics for implicit integration, regardless of the excessive numerical damping it introduces, especially with large time-steps. Newmark-beta integration avoids numerical damping, but it is less stable. Yet, we show that we can achieve stable simulations with Newmark-beta using our adaptive time-stepping formulation and thereby avoid the numerical damping issues of backward Euler.

We compare our approach to typical adaptive time-stepping formulations used in computer graphics. For fair comparisons, we adjust the parameters of all methods such that they produce the same number of time-steps. Our results show that our formulation leads to substantially improved simulation accuracy as compared to alternative methods using the same number of time-steps. Thus, we conclude that our approach does a better job at determining how the time-step size must be adjusted.

## 2. Prior Work

Adaptive time-stepping for numerical integration has been studied for decades in computational science and engineering since the introduction of the Simson's rule [Hen61] and the first recursive adaptive integration method based on this rule [Kun62]. Many variants of these methods have been proposed over the years [McK62,MT63,McK63,But64,Lyn69,FMM77]. Other methods use error estimation [OS68, Gen91, Lau65], recursively monotone stable quadrature rules [SS89], and integration with stratified sequences [VL02]. Yet, the primary concern for computational science and engineering has been simulation accuracy. The numerical stability is automatically achieved as a byproduct of accurate integration.

In computer graphics, on the other hand, the simulation accuracy has often been a secondary concern to computational performance and numerical stability. Often times, largest possible time-steps are preferred, regardless of the simulation accuracy as long as the simulations are numerically stable. Therefore, the adaptive time-stepping methods used in computer graphics have been extremely simple. Arguably the most popular approach is speed limiting (using a very basic CFL condition) [Bri09, IAGT10, GP11, FSH11, SSB13, TLK16] by determining a maximum permissible speed as a function of time-step size. Indeed, unstable simulations lead to high-velocity motion, which can be detected and avoided using speed limiting. Yet, high-velocity does not necessarily mean simulation instability, so speed limiting can be too conservative for achieving large stable time-steps. Furthermore, high-speed motion that is produced by an unstable simulation step can be the result of simulation inaccuracies in prior time-steps, which cannot always be detected by speed limiting. Therefore, some methods combine speed limiting with additional heuristics, such as the rate of change in particle density [DC99] or spring stretch [Pro96, BW98, BFA02, BWHT07]. However, none of these approaches provide a reliable estimation of simulation stability or accuracy. As a result, in practice their parameters are set conservatively, which leads to limited performance gain from adaptive time-stepping.

In general, these adaptive time-stepping methods typically treat the numerical integrator as a black-box, and they alter the time-step size solely based on the input of the integrator. The method we introduce in this paper, however, is based on the foreknowledge of the implicit integration scheme and the inherent error in the construction of the implicit system. Thus, we can achieve improved simulation accuracy, stability, and performance, as compared to alternative methods often used in computer graphics.

## 3. Adaptive Time-Stepping

Our adaptive time-stepping formulation depends on the details of the numerical integrator. Therefore, we first describe our approach using the backward Euler scheme (Section 3.1) and present how we compute a safe time-step size (Section 3.2), then we show how the same approach can be applied to the Newmark-beta integration scheme (Section 3.3).

## 3.1. Error Estimation for Force Linearization

Let $\mathbf{x}^n$ and $\mathbf{v}^n$ represent the position and velocity state of the system at time-step $n$. For a given time-step size $\Delta t$, backward Euler computes the state for the next time-step $(n+1)$ using

$$\mathbf{v}^{n+1} = \mathbf{v}^n + \Delta t\,\mathbf{M}^{-1}\mathbf{F}^{n+1}, \tag{1}$$
$$\mathbf{x}^{n+1} = \mathbf{x}^n + \Delta t\,\mathbf{v}^{n+1}. \tag{2}$$

where $\mathbf{M}$ is the diagonal mass matrix and $\mathbf{F}^{n+1}$ is the forces at the end of the time-step. Since $\mathbf{F}^{n+1}$ depends on the unknowns $\mathbf{x}^{n+1}$ and velocity $\mathbf{v}^{n+1}$, it cannot be computed directly. A common solution is to approximate $\mathbf{F}^{n+1}$ using Taylor series expansion up to the first derivatives [BW98], such that

$$\widetilde{\mathbf{F}}^{n+1} = \mathbf{F}^n + \frac{\partial \mathbf{F}^n}{\partial \mathbf{x}}\Delta\mathbf{x} + \frac{\partial \mathbf{F}^n}{\partial \mathbf{v}}\Delta\mathbf{v}. \tag{3}$$

A linear system is formed using this linear approximation $\widetilde{\mathbf{F}}^{n+1}$ in Equation 1 and combining it with Equation 2, which can then be solved using numerical methods like conjugate gradient. This corresponds to a single Newton iteration step. It is typical for graphics applications to use a single Newton step for computing $\mathbf{x}^{n+1}$ and $\mathbf{v}^{n+1}$, since each additional Newton step would have a similar computation cost as the first one. If the simulation requires more Newton steps, it is solved using a smaller time-step size $\Delta t$.

Our observation is that the linear approximation of the force term $\widetilde{\mathbf{F}}^{n+1}$ using only the first derivatives is a major source of error that leads to numerical instability. Indeed, $\widetilde{\mathbf{F}}^{n+1}$ can substantially deviate from $\mathbf{F}^{n+1}$, especially when the function $\mathbf{F}$ is highly nonlinear within the time-step. Once the new time step is computed and $\mathbf{x}^{n+1}$ and $\mathbf{v}^{n+1}$ are known, however, we can directly compute $\mathbf{F}^{n+1}$. In fact, computing $\mathbf{F}^{n+1}$ is a part of the computation for the next time-step $(n+1)$. Therefore, at this point we can directly measure the error introduced due to the linearized force approximation by simply comparing $\widetilde{\mathbf{F}}^{n+1}$ to $\mathbf{F}^{n+1}$. The difference indicates how much the linear approximation of forces deviate from the actual forces of the simulation state $\mathbf{x}^{n+1}$ and $\mathbf{v}^{n+1}$.

Thus, we can estimate the error of time integration using $\|\mathbf{F}^{n+1} - \widetilde{\mathbf{F}}^{n+1}\|_\infty$. If this error is large, we reject the time-step computation and reduce the time-step size $\Delta t$. If the error is within a user-defined tolerance, we accept the time-step and try to increase the time-step size for the next time-step computation.

One difficulty with this approach is setting a proper threshold for the force error. Therefore, we convert the error in force approximation into error in position update. Note that the positions at the end of the time-step are computed using the linearized forces (combining Equations 1 and 2), such that

$$\mathbf{x}^{n+1} = \mathbf{x}^n + \Delta t\mathbf{v}^n + \Delta t^2\mathbf{M}^{-1}\widetilde{\mathbf{F}}^{n+1}. \tag{4}$$

After the time-step integration, we can compute a different position state $\bar{\mathbf{x}}^{n+1}$ by directly using $\mathbf{F}^{n+1}$, resulting

$$\bar{\mathbf{x}}^{n+1} = \mathbf{x}^n + \Delta t\mathbf{v}^n + \Delta t^2\mathbf{M}^{-1}\mathbf{F}^{n+1}. \tag{5}$$

The difference between these two position states indicates the error in position update $\varepsilon$, such that

$$\varepsilon = \|\mathbf{x}^{n+1} - \widetilde{\mathbf{x}}^{n+1}\|_\infty, \tag{6}$$

which can be computed directly from the difference in forces, using

$$\varepsilon = \Delta t^2 \left\|\mathbf{M}^{-1}\left(\mathbf{F}^{n+1} - \widetilde{\mathbf{F}}^{n+1}\right)\right\|_\infty. \tag{7}$$

Thus, for bounding the linearization error, we use a user-defined tolerance parameter $\varepsilon_{\max}$. If $\varepsilon \le \varepsilon_{\max}$, we accept the time-step computation. Otherwise, we reduce the time-step size $\Delta t$ and recompute the time-step.

## 3.2. Time-Step Size Estimation

Another challenge of adaptive time-stepping is estimating a "safe" time-step size $\Delta t$ for producing results that would be accepted by the adaptive time-stepping criteria. In our case, an ideal time-step size would produce results with $\varepsilon = \varepsilon_{\max}$, so that the time-step integration result would be accepted and that the time-step size would be as large as possible. However, we can compute $\varepsilon$ only after time-step integration. Therefore, we must use an estimation of $\varepsilon$ for adjusting the time-step size prior to time-step integration.

For estimating $\varepsilon$ we use the results of a previously computed time-step integration. We approximate the force linearization error $\mathbf{F}_E^{n+1}$ as a linear function of the *new* time-step size $h$, using

$$\mathbf{F}_E^{n+1}(h) = \frac{h}{\Delta t}\left(\mathbf{F}^{n+1} - \widetilde{\mathbf{F}}^{n+1}\right), \tag{8}$$

Using $\mathbf{F}_E^{n+1}$, the position-based linearization error $\varepsilon$ can be estimated as a function of the new time step size $h$, such that

$$\varepsilon_E(h) = h^2\left\|\mathbf{M}^{-1}\mathbf{F}_E^{n+1}(h)\right\|_\infty \tag{9}$$
$$= \frac{h^3}{\Delta t}\left\|\mathbf{M}^{-1}\left(\mathbf{F}^{n+1} - \widetilde{\mathbf{F}}^{n+1}\right)\right\|_\infty. \tag{10}$$

Let $h_{\max}$ represent the estimated ideal time-step size, such that $\varepsilon_E(h_{\max}) = \varepsilon_{\max}$. We can calculate $h_{\max}$ using

$$h_{\max} = \left(\frac{\Delta t\,\varepsilon_{\max}}{\left\|\mathbf{M}^{-1}\left(\mathbf{F}^{n+1} - \widetilde{\mathbf{F}}^{n+1}\right)\right\|_\infty}\right)^{1/3}. \tag{11}$$

Using this formulation, we estimate that a time-step size of $h_{\max}$ would produce a maximum position error of $\varepsilon_{\max}$ due to force linearization. Thus, if this estimation is correct, using $h_{\max}$ as our time-step size, the time-step integration would have the maximum permitted error $\varepsilon_{\max}$. However, when this formulation underestimates the error, using $h_{\max}$ as the time-step size would lead to a larger error that would cause the time-step integration result to be rejected. Therefore, it would be favorable to pick a time-step size that is expected to produce less than $\varepsilon_{\max}$ error, so that we can reduce the possibility of getting more than $\varepsilon_{\max}$ error after time-step integration. We achieve this by introducing a secondary user-defined parameter $\alpha \in (0,1]$, such that the time-step size is updated using

$$\Delta t \leftarrow \left(\frac{\Delta t\,\alpha\,\varepsilon_{\max}}{\left\|\mathbf{M}^{-1}\left(\mathbf{F}^{n+1} - \widetilde{\mathbf{F}}^{n+1}\right)\right\|_\infty}\right)^{1/3}. \tag{12}$$

The $\alpha$ parameter simply defines an acceptable window for the estimated error bound. Using smaller $\alpha$ values makes our adaptive time-stepping less aggressive, but it also reduces the possibility of compute time-step integration that would ultimately be rejected. We use $\alpha = 0.5$ for all examples in this paper. As a result, the time-step sizes we use correspond to approximately $0.79h_{\max}$.

We use time-step size estimation after every time-step integration, regardless of whether the time-step computation is accepted or rejected. Note that the updated time-step size automatically becomes smaller if the previous time-step integration result is rejected. Therefore, even if our linear error estimation is inaccurate, our formulation guarantees convergence with $\alpha < 1$.

### 3.3. Newmark-beta Integration

Numerical damping is a well-known problem of backward Euler integration. This is due to the fact that the position update in Equation 2 assumes constant velocity within each time-step. Obviously, the error for this assumption is larger for larger time-step sizes. Therefore, numerical damping due to backward Euler integration varies as we modify the time-step size.

Newmark-beta integration [New59] (with parameters $\gamma = 1/2$ and $\beta = 1/4$) avoids numerical damping by using a slightly different formulation for velocity and position updates, such that

$$\mathbf{v}^{n+1} = \mathbf{v}^n + \Delta t\, \mathbf{M}^{-1}\left(\frac{\mathbf{F}^n + \mathbf{F}^{n+1}}{2}\right),$$
$$\mathbf{x}^{n+1} = \mathbf{x}^n + \Delta t\left(\frac{\mathbf{v}^n + \mathbf{v}^{n+1}}{2}\right).$$

In this case, the velocity update assumes linearly changing acceleration and position update assumes linearly changing velocity. These update rules can also be considered as a combination of explicit and implicit integrations. Therefore, Newmark-beta integration is less stable than backward Euler, but its ability to avoid numerical damping can make it preferable for some computer graphics problems [GHDS03, BMF03].

The stability issues of Newmark-beta integration can be addressed by adaptive time-stepping. Also, integrators like Newmark-beta can be favorable in the context of adaptive time-stepping, since the variations in the time-step size do not impact the damping behavior of the simulation.

Using the update rules of Newmark-beta integration, our position-based error estimate becomes

$$\varepsilon = \frac{\Delta t^2}{4}\left\|\mathbf{M}^{-1}\left(\mathbf{F}^{n+1} - \widetilde{\mathbf{F}}^{n+1}\right)\right\|_\infty , \tag{13}$$

and the new time-step size is computed using

$$h_{\max} = \left(\frac{4\,\Delta t\,\varepsilon_{\max}}{\left\|\mathbf{M}^{-1}\left(\mathbf{F}^{n+1} - \widetilde{\mathbf{F}}^{n+1}\right)\right\|_\infty}\right)^{1/3} . \tag{14}$$

Note that the same amount of force error leads to four times less position error with Newmark-beta integration, as compared to backward Euler, and the computation of $h_{\max}$ multiples $\varepsilon_{\max}$ by four. Therefore, using a smaller value for $\varepsilon_{\max}$ would be advisable with Newmark-beta integration.

## 4. Results

We test our adaptive time stepping method with a few simulation systems: a simple 1D mass-spring system, a cloth simulator, and a brittle fracture simulation system using peridynamics [LBC*14].

The 1D mass-spring simulation contains a single unit-mass particle attached to the origin ($x = 0$) by a zero rest-length non-linear spring. The spring force is given by

$$\mathbf{F}(x,v) = -K_x(x + x^3) - K_v(v + v^3) ,$$

where the spring constants are set as $K_x = 100$, $K_v = 1$. The initial states are set as $x^0 = -1$, $v^0 = 0$. Integration results using backward Euler and Newmark-beta schemes with different adaptive time stepping methods are shown in Figure 1. The parameters of each adaptive time stepping method are adjusted such that the all methods (except the reference) take approximately the same number of steps. Note that the reference solution in each figure is computed using the fixed time step integration method but with much smaller time step size. It can be seen in Figure 1 that our method clearly provides closer results to the reference solution in comparison to the other methods. Also note that the backward Euler integration introduces much numerical damping, which significantly affects the particle evolution.

Speed limiting that uses a simple CFL condition to determine the time step size based on the speed of the particle. Although it is widely used in computer graphics, in this test it fails to respond to strong acceleration quickly and it deviates the most from the reference solution. Moreover, it spends too much computation when the particle is moving fast, even though a relatively small number of steps can accurately reproduce the reference motion.

Stretch limiting also reduces the time step size significantly when the particle is moving fast, as it depends on the change of particle position per time step. Similar to speed limitting, it spends most of its computation when in fast motion and misses the details of the motion with relatively minor stretching of the spring.

Adaptive quadrature [McK62] compute an approximation error by trying to integrate the system at current state using a full time step size and two consecutive half time step size, then compute the difference between the integration results. It recursively subdivides the time step size until the estimate error is smaller than a given threshold. Besides the extra computation it includes for more integrations at every step, it fails to reproduce the results closed to the reference motion using the same number of time steps as our method.

In Figure 2, we compare a number of brittle fracture simulations using peridynamics [Sil00, LBC*14]. The results are produced by Newmark-beta integration scheme [New59, GHDS03, BMF03] using both fixed time step size and our adaptive time stepping strategy. As the time step computation using adaptive time stepping may be discarded, we introduce fracture by removing the connection between particles only after we accept to take a time step. Therefore, our implementation does not consider the fracturing events as a condition for reducing the time step size. Thus, adaptive time stepping impacts the frequency of the fracturing tests, which is expected to alter the propagation of the simulated fracture. Nonetheless, our results show that our adaptive time integration produces fracture patterns similar to the more expensive fixed time step integration.

In addition, because our adaptive time stepping strategy does not depend on the particle velocities but internal forces, it allows to save much computation by taking very large time step size when the system is free from collision. Thus, it achieves overall $3X$ to $6X$ speed up over fixed time step integration, which is impossible with other adaptive time stepping approaches. As the simulation size increases, the speedup with adaptive time stepping increases as well. Yet, it is important to note that the speedup depends on the events that happen during the simulation. More importantly, even though adaptive time stepping allows taking substantially larger time steps, our Conjugate Gradients solver (if applicable) can take more than an order of magnitude more iterations to converge when using large time steps. Therefore, introducing a preconditioner that could reduce the number of iterations with large time steps can substantially improve the speedup of our adaptive time stepping.

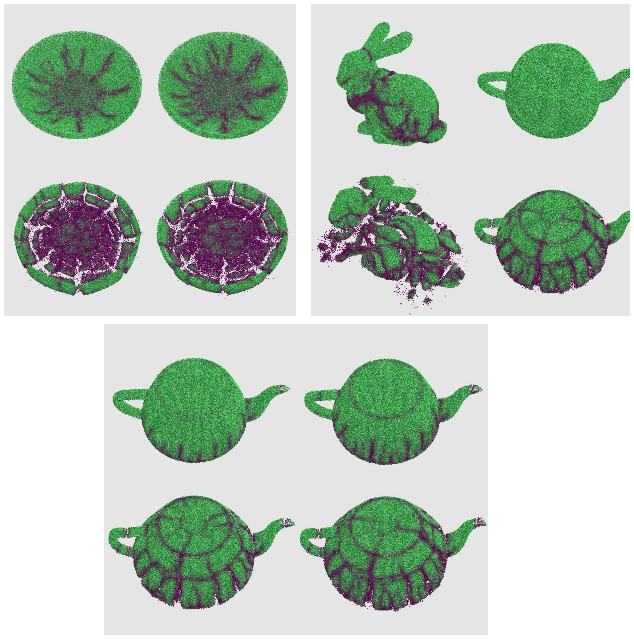
**Figure 2:** *Comparison of simulations running with fixed time step sizes (left columns) and adaptive time step sizes (right columns) at frames 30 (top) and 35 (bottom).*

## 5. Conclusion

We introduce an adaptive time stepping technique for implicit time integration. Our method is based on the error originated from the linearization of the internal force term in implicit integration scheme therefore we can bound the error introduced by the numerical time integration. We have showed that, under the same computational budget, the simulations using our adaptive time stepping approach produce superior accuracy and stability as compared to other prior work.

While we demonstrate examples using Backward Euler and Newmark-beta schemes only, the adaptive time stepping strategy we introduce is general and it is applicable to any of other implicit integration systems that solving solution by using linearization of the force model.

## References

[BFA02] Bridson R., Fedkiw R., Anderson J.: Robust treatment of collisions, contact and friction for cloth animation. *ACM Trans. Graph. 21*, 3 (July 2002), 594–603. 2

[BMF03] Bridson R., Marino S., Fedkiw R.: Simulation of clothing with folds and wrinkles. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2003), SCA '03, Eurographics Association, pp. 28–36. 4

[Bri09] Bridson R.: *Fluid Simulation For Computer Graphics*. A.K. Peters, 2009. 2

[But64] Butler H. S.: Certification of algorithm 182: Nonrecursive adaptive integration. *Commun. ACM 7*, 4 (Apr. 1964), 244–. 2

[BW98] Baraff D., Witkin A.: Large steps in cloth simulation. In *Proceedings of SIGGRAPH '98* (1998), pp. 43–54. 2, 3

[BWHT07] Bargteil A. W., Wojtan C., Hodgins J. K., Turk G.: A finite element method for animating large viscoplastic flow. *ACM Trans. Graph. 26*, 3 (July 2007). 2

[DC99] Desbrun M., Cani M.-P.: *Space-Time Adaptive Simulation of Highly Deformable Substances*. Research Report RR-3829, INRIA, 1999. 2

[FMM77] Forsythe G. E., Malcolm M. A., Moler C. B.: *Computer Methods for Mathematical Computations*. Prentice Hall Professional Technical Reference, 1977. 2

[FSH11] Fierz B., Spillmann J., Harders M.: Element-wise mixed implicit-explicit integration for stable dynamic simulation of deformable objects. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2011), SCA '11, ACM, pp. 257–266. 2

[Gen91] Genz A.: *An adaptive numerical integration algorithm for simplices*. Springer New York, New York, NY, 1991, pp. 279–285. 2

[GHDS03] Grinspun E., Hirani A. N., Desbrun M., Schröder P.: Discrete shells. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2003), SCA '03, Eurographics Association, pp. 62–67. 4

[GP11] Goswami P., Pajarola R.: Time Adaptive Approximate SPH. In *Workshop in Virtual Reality Interactions and Physical Simulation "VRIPHYS" (2011)* (2011), Bender J., Erleben K., Galin E., (Eds.), The Eurographics Association. 2

[Hen61] Henriksson S.: Contribution no. 2: Simpson numerical integration with variable length of step. *BIT Numer. Math. 1* (1961), 290. 2

[IAGT10] Ihmsen M., Akinci N., Gissler M., Teschner M.: Boundary Handling and Adaptive Time-stepping for PCISPH. In *Workshop in Virtual Reality Interactions and Physical Simulation "VRIPHYS" (2010)* (2010), Erleben K., Bender J., Teschner M., (Eds.), The Eurographics Association. 2

[Kun62] Kuncir G. F.: Algorithm 103: Simpson's rule integrator. *Commun. ACM 5*, 6 (June 1962), 347–. 2

[Lau65] Laurie D. P.: Sharper error estimates in adaptive quadrature. *BIT Numerical Mathematics*, 23 (1965), 258–261. 2

[LBC*14] Levine J. A., Bargteil A. W., Corsi C., Tessendorf J., Geist R.: A peridynamic perspective on spring-mass fracture. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2014), SCA '14, pp. 47–55. 4

[Lyn69] Lyness J. N.: Notes on the adaptive simpson quadrature routine. *J. ACM 16*, 3 (July 1969), 483–495. 2

[McK62] McKeeman W. M.: Algorithm 145: Adaptive numerical integration by simpson's rule. *Commun. ACM 5*, 12 (Dec. 1962), 604–. 2, 4

[McK63] McKeeman W. M.: Algorithm 198: Adaptive integration and multiple integration. *Commun. ACM 6*, 8 (Aug. 1963), 443–444. 2

[MT63] McKeeman W. M., Tesler L.: Algorithm 182: Nonrecursive adaptive integration. *Commun. ACM 6*, 6 (June 1963), 315–. 2

[New59] Newmark N. M.: A method of computation for structural dynamics. *Journal of the Engineering Mechanics Division 85*, 3 (1959). 2, 4

[OS68] O'Hara H., Smith: Error estimation in clenshaw-curtis quadrature formula. *Comput. 11*, 2 (1968), 213–219. 2

[Pro96] Provot X.: Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *In Graphics Interface* (1996), pp. 147–154. 2

[Sil00] Silling S. A.: Reformulation of elasticity theory for discontinuities and long-range forces. *Journal of the Mechanics and Physics of Solids 48*, 1 (2000), 175 – 209. 4

[SS89] Sugiura H., Sakurai T.: On the construction of high-order integration formulae for the adaptive quadrature method. *Journal of Computational and Applied Mathematics 28* (1989), 367 – 381. 2

[SSB13] Sin F. S., Schroeder D., Barbic J.: Vega: Non-linear fem deformable object simulator. *Computer Graphics Forum 32*, 1 (2013), 36–48. 2

[TLK16] Teng Y., Levin D. I. W., Kim T.: Eulerian solid-fluid coupling. *ACM Trans. Graph. 35*, 6 (Nov. 2016), 200:1–200:8. 2

[VL02] Venter A., Laurie D. P.: A doubly adaptive integration algorithm using stratified rules. *BIT Numerical Mathematics 42*, 1 (2002), 183–193. 2