Adaptive Rigidification of Discrete Shells

ALEXANDRE MERCIER-AUBIN, McGill University, Canada
PAUL G. KRY, McGill University, Canada

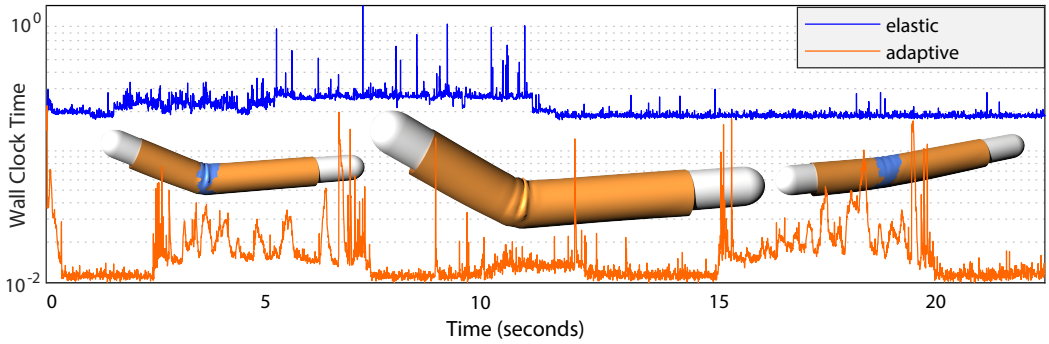

Fig. 1. Local deformations at the elbow require elastic deformation during bending, while the rest of the sleeve simulates as a rigid body, providing an order of magnitude faster computation in comparison to a fully elastic simulation. A bending arm tends to only generate local deformation on a sleeve near the elbow with otherwise rigidly moving wrinkles. This allows the coarsening of fine detailed wrinkles through adaptive rigidification. In this example, our implementation achieves a nearly constant improvement of performances more than an order of magnitude faster than the elastic simulation.

We present a method to improve the computation time of thin shell simulations by using adaptive rigidification to reduce the number of degrees of freedom. Our method uses a discretization independent metric for bending rates, and we derive a membrane strain rate to curvature rate equivalence that permits the use of a common threshold. To improve accuracy, we enhance the elastification oracle by considering both membrane and bending deformation to determine when to rigidify or elastify. Furthermore, we explore different approaches that are compatible with the previous work on adaptive rigidifcation and enhance the accuracy of the elastification on new contacts without increasing the computational overhead. Additionally, we propose a scaling approach that reduces the conditioning issues that arise from mixing rigid and elastic bodies in the same model.

Additional Key Words and Phrases: cloth, shells, adaptive simulation, rigid bodies, finite element

ACM Reference Format:
Alexandre Mercier-Aubin and Paul G. Kry. 2023. Adaptive Rigidification of Discrete Shells. *Proc. ACM Comput. Graph. Interact. Tech.* 6, 2, Article 1 (August 2023), 17 pages. https://doi.org/10.1145/3606932

Authors' addresses: Alexandre Mercier-Aubin, McGill University, Canada, alexandre.mercier-aubin@mail.mcgill.ca; Paul G. Kry, McGill University, Canada, kry@cs.mcgill.ca.

2577-6193/2023/8-ART1 $15.00
https://doi.org/10.1145/3606932

1 INTRODUCTION

In any standard elastodynamic simulation, interactions between models are governed by physical laws for realistic simulations. Integrating in time the degrees of freedom often requires expensive iterative solvers that terminate before convergence or introduce significant numerical damping. Scaling these simulations to large scenes without approximations can be challenging, as many approximations are set a priori, such as discretizing time according to a fixed step size and discretizing models into elements. However, precomputing the initial discretization of models can help alleviate some of the computational stress, and adaptive techniques can refine or coarsen the objects as needed. As the number of elements in a scene grows, reducing the size of the system becomes increasingly important for achieving efficient and accurate simulations.

The simulation of thin tetrahedral meshes is subject to even more problems, as it often leads to poorly conditioned system due to sliver shaped elements, or extremely high resolution models necessary for the deformation of very thin models. Instead of volumetric meshes, 2D models with a thickness parameter and assumption of non-deformation in the normal directions can be used to create plausible simulations with a potentially coarser mesh while achieving similar results. However, thin shells using 2D elements poses its own set of challenges. The bending energies introduce non-linearity, necessitating smaller step sizes or an higher number of iterations for time integration. Nevertheless, bending energies are necessary to create dynamic, cloth-like wrinkles with rich wavy motions.

In this paper, we propose a method to adaptively coarsen and refine thin shells at runtime. This builds upon the rigidification method introduced by Mercier-Aubin et al. [2022]. In all large scale simulations, some elements will inevitably be static or moving rigidly, and therefore wasting computational resources on the deformation of non-deforming degrees of freedom. In Figure 1, a horizontally oriented sleeve has large undeforming regions when it bends, but also features dynamic deformation near the elbow. Adaptive rigidification is efficient even on highly wrinkled models. On-demand elastification makes dynamic scenes properly dependent on their environment, allowing elements to alternate between a fast rigid representation and the accurate elastic counterpart.

Direct application of the previous adaptive rigidification method in the case of shells introduces new challenges. The bending deformation occurs over edges, making it impossible to evaluate using only the per triangle strain rate from membrane deformation. Additionally, bending along the discretized edges is unrelated to stretching. For example, a flat hanging piece of cloth experiences only stretching in the planar directions due to gravity and does not express any bending deformation. Likewise, making each triangle an independent rigid body capable of bending would require more degrees of freedom per element, which defeats the purpose of adaptive rigidification. Because bending and membrane deformation are two independent types of deformation, we must consider both in our rigidification process as opposed to the previous work. Shells also require a new threshold for the bending motions to prevent premature rigidification, which would prevent elements from bending.

Collisions are a common mechanism for elastification, and for good performance we need an oracle that can identify a small local region of the mesh to elastify based on a threshold. The previous work uses a single iteration of preconditioned conjugate gradient, which can be problematic for shells, in particular, those with high membrane stiffness. If only a small number of contacts are active, the single iteration of preconditioned conjugate gradient is not enough to properly propagate the elastification and cause the bending component to wake up. We present different approaches, including a fast contact filter that approximate the behaviour of the impact on the nodes adjacent to the contact location.

In the following sections, we present a new adaptive simulation method for thin shells, which can produce results that closely resemble a fully elastic simulation while often having computation times more than an order of magnitude faster. We test our improvement to the oracle's contact handling by comparing our filter with the previous method for handling contacts as well as slower, but more accurate approaches to solve for the approximate contact velocities. We also discuss the difference between the curvature rate and the dihedral angle rate as well as some of their potential applications in the rigidification process.

2 RELATED WORK

Remeshing [Alliez et al. 2008] is often used to adaptively coarsen shells on the fly. This generates new positions using either subdivision rules [Bender and Deul 2012; Cirak et al. 2000; Li and Volkov 2005; Loop 1987] or splitting coupled with edge flipping to improve the conditioning elements [Jiao et al. 2006]. This type of technique generally must handle the buckling or instabilities that can arise when remeshing curved surfaces with coarser triangles. One way to handle this is to use a measure the quality of the new elements to determine when to remesh them [Narain et al. 2012; Wicke et al. 2010]. These techniques are good for surfaces that are developable or that have large nearly flat regions, because these regions will remain indistinguishable from fine to coarse. Another approach refines basis functions [Grinspun et al. 2002; Hahn et al. 2014] to allow deformation in the coarse models by adding new degrees of freedom. In contrast, our method is orthogonal to these approaches and reduces the size of the system by simulating those regions of a mesh without dynamic deformations as rigid bodies, regardless if the region is flat or consisting of a dense collection of triangulated folds.

In contrast to mesh refinement, via remeshing, with the goal of simulating a mesh with the appropriate resolution, other approaches separate the problem into two distinct parts, i.e., a coarse simulation and a second mechanism for adding details. For instance, the approach of Rohmer et al. [2010] first simulates a coarse models, and the refines the coarse mesh in a post-process refinement to add wrinkle details. A different strategy is taken by Müller and Chentanez [2010], where fine mesh vertices permit wrinkles via approximate constraints to a coarse simulation. There likewise exists other strategies and heuristics for upscaling simulations. The formation of wrinkles can be data-driven [Miguel et al. 2012; Popa et al. 2009], and machine learning techniques can predict physically plausible coarse-to-fine mappings of vertices [Kavan et al. 2011]. Our approach instead starts with a high-resolution model and lets the simulation choose which parts need the degrees of freedom, and which regions can be evolved with only rigid motion.

While not directly related to adaptive simulation, we note that the simulation of stiff shells can be challenging because large stiffness ratios can lead to poorly conditioned systems. Physical fabrics typically have a non-linear stress-strain relationship that increase the stiffness rapidly as it deforms. One way to reduce the conditioning issue is to add strain limits. This can be done with correction strain constraints or projections after each integration step [Goldenthal et al. 2007; Provot 1995]. In the case of inextensible cloth [English and Bridson 2008], constraints provide this strain limit in a quadrilateral mesh which avoids locking artifacts due to discretization. In contrast to these other techniques, our method makes strain limited regions rigid when they maintain the limit (and have zero deformation rate in the orthogonal direction). This addresses the conditioning problems, i.e., when these strain limited regions are simulated as rigid.

Freezing and sleeping techniques are well known approaches to save computation during static periods of dynamic simulations. For instance, using an adaptive Hamiltonian can reduce the size of the simulations at run-time by disabling positional degrees of freedom of a system in regions where there is low momentum [Artemova and Redon 2012; Manteaux et al. 2013]. Freezing can likewise be implemented in a hierarchy to dynamically tune the complexity of a viscoelastic body [Tournier

et al. 2014]. Different metrics can be used to initiate freezing, for example, by analyzing kinetic energies or velocities in simulations of rigid body contact [Erleben 2004; Schmidl and Milenkovic 2004]. In the context of rigid bodies, another approach is to merge [Coevoet et al. 2020] collections of contacting bodies when they have zero relative velocity. This is similar to our work in that it reduces the degrees of freedom of the system, simplifies collision detection cost, while still permitting the collection to move as a rigid body.

Merging degrees of freedom into rigid bodies is likewise a key idea in adaptive rigidification of elastic solids [Mercier-Aubin et al. 2022], i.e., that it is valuable to permit regions of an elastic simulation to continue moving rigidly, as opposed to freezing degrees of freedom in the inertial frame.

3 METHODS

We first extend adaptive rigidification to support triangular elements in a 3D setting. Thin shells feature two major hindrances that require a different approach for rigidification; the bending component cannot be analyzed from the deformation gradient, and the constant thickness of shells requires the definition of the deformation gradient to be changed for rigid rotations so as not to yield a non-zero strain.

3.1 Simulation of Shells

For standard volumetric meshes, we define the deformation gradient $F = Bx$ from a kinematic mapping B and the generalized coordinates x of our element's vertices [Bro-Nielsen and Cotin 1996]. However, shells are not inherently volumetric. We use an approximation of the thickness by projecting out any deformation in the normal directions of the faces [Levin 2020]. This is akin to simulating a fake prismatic element. We do this projection by adding a new term η to the definition of the deformation gradient

$$F = Bp + \eta n, \tag{1}$$

which then gives us the deformation gradient for shells. This $\eta \in \mathbb{R}^{9\times 3}$ term

$$\eta = \begin{bmatrix} \boldsymbol{n^0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{n^0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{n^0} \end{bmatrix} \tag{2}$$

is a block matrix where the zeros are 3 by 1 column vectors of zeros. The B and η matrices are pre-computed at the start of the simulation with the fully elastic mesh and cached.

Baraff and Witkin [1998] separate the potential energy into three energies. Both the shear and stretch energies use a function that maps a 2D projection of thin shells to their 3D world positions, effectively computing the energies as 2D problems. Grinspun et al. [2003] introduces a bending energy using the angle between two face normals, which is coupled with a stiffness parameter and creates resistance to bending. We use a similar approach by separating the membrane energies from the bending energies.

3.1.1 Membrane Energy. To help ensure stability we use implicit time integration [Etzmub et al. 2003], so for the membrane energy density Ψ_e we must compute the gradient $\frac{\partial \Psi_e}{\partial F}$ and Hessian $\frac{\partial^2 \Psi_e}{\partial F^2}$. To get the internal forces $f_e = -V\frac{\partial \Psi_e}{\partial F}$ we multiply the gradient by the negative volume. In the case of a shell, the volume is the area of the triangle multiplied by a thickness parameter. In our examples, we tried various membrane energy formulations including neoHookean, Saint-Venant Kirchoff, and more. We note that we use the Grinspun et al. [2003] membrane energy in our time comparisons because in our experience it allows a bigger step size with fewer Newton iterations.

3.1.2 Bending Energy. We compute the dihedral angle

$$\theta = \pi - \tan^{-1} \frac{\boldsymbol{e} \cdot (\boldsymbol{n}_1 \times \boldsymbol{n}_2)}{\boldsymbol{n}_1 \cdot \boldsymbol{n}_2}, \tag{3}$$

$$\Psi_b = \sum_e (\theta_e - \bar{\theta}_e) \|\bar{\boldsymbol{e}}\| / \bar{h}_e, \tag{4}$$

for each edge, where $\boldsymbol{n}_1$, and $\boldsymbol{n}_2$ are the adjacent face normals, and $\boldsymbol{e}$ is the edge vector (see Figure 2). We then use the resulting angles to compute the bending energies from Tamstorf and Grinspun [2013]. Here $\bar{\theta}_e$ is the dihedral angle at rest $\bar{h}_e$ is the length of the dual edge connecting the barycenters of two triangles at rest, and $\bar{\boldsymbol{e}}$ is the edge at rest.

We sum the bending energy to the membrane energy $\Psi = \Psi_b + V\Psi_e$, then find their gradient and Hessian to allow implicit time integration.

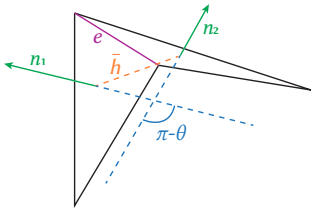

Fig. 2. dihedral angle

3.2 Simulating Coupled Rigid Bodies

In our mixed simulations, the elastic portions of the shell can have dynamic deformations while other portions replaced with rigid bodies do not allow for bending, with each connected component acting as a single thin rigid body. The adaptive model reduction uses the matrix

$$G = \begin{bmatrix} I & 0 \\ 0 & \Gamma \end{bmatrix}, \tag{5}$$

to map degrees of freedom from the mixed rigid-elastic system to the fully elastic model. Each vertex inside the rigidified body has a vector r that points to it from the center of mass in rigid body frame. From this we can build a matrix

$$\Gamma_i = \begin{bmatrix} I & -(R\,\boldsymbol{r}_i)^\times \end{bmatrix} \tag{6}$$

that maps the rigid body velocities to each individual vertex of the rigid body.

For every mesh, we group rigid elements using an adjacency graph of elements with shared edges for shells. We compute this graph prior to the simulation. Using a breadth first search algorithm, we navigate the graph to find the rigid connected component. Each connected component is a new rigid body in our simulation. When building the bodies, we compute properties like the center of mass p, the rotation $R \in SO(3)$, the angular velocity ω, the linear velocity v. These are set according to the state of the degrees of freedom rigidified to exactly preserve momentum under rigid motions.

On rigidification, we replace the elastic degrees of freedom in our system with the appropriate rigid degrees of freedom using the current step's G mapping from Equation 5.

Lumping elements together has an impact on the mass ratio of the system which can lead to a higher condition number. We scale the per Newton-step linear system to get the condition number down to a similar or better range as the fully elastic system. We use the sparse matrix scaling of Curtis and Reid [1972], which iteratively solves a least-square problem for the row and columns scaling factors. The inclusion of a scaling matrix therefore makes adaptive rigidification easier to integrate with iterative solvers.

3.3 Time Integration

Shells are subject to tricky scenarios like buckling that require more than one Newton iteration. Instead of a single linearized Newton step, we simulate shells by using the common optimization

input : Velocities $\dot{\boldsymbol{x}}_t$ at the start of the step
Function Q to minimize
Kinematic elastic-adaptive mapping G
Predicted reduction σ stop parameter
Scaling matrix S
output : Velocities $\dot{\boldsymbol{x}}_{t+1}$ after stepping
$\dot{x}_{t+1} \leftarrow \dot{x}_t$
for $i <$ *Newton iterations* **do**
  $A \leftarrow SG^T(\nabla^2 Q(\dot{\boldsymbol{x}}_t, \dot{\boldsymbol{x}}_{t+1}))G$
  $\boldsymbol{b} \leftarrow -SG^T(\nabla Q(\dot{\boldsymbol{x}}_t, \dot{\boldsymbol{x}}_{t+1}))$
  $\Delta\dot{\boldsymbol{x}} \leftarrow A^{-1}\boldsymbol{b}$
  $\alpha \leftarrow 1$
  while $Q(\dot{\boldsymbol{x}}_t, \dot{\boldsymbol{x}}_{t+1}) - Q(\dot{\boldsymbol{x}}_t, \dot{\boldsymbol{x}}_{t+1} + \alpha\Delta\dot{\boldsymbol{x}}) < \sigma\alpha Q(\dot{\boldsymbol{x}}_t, \dot{\boldsymbol{x}}_{t+1})$ **do**
    $\alpha \leftarrow \frac{1}{2}\alpha$
  end
  $\Delta\dot{\boldsymbol{x}}_c \leftarrow$ Contact Solve
  $\dot{\boldsymbol{x}}_{t+1} \leftarrow \dot{\boldsymbol{x}}_{t+1} + G^T(\alpha(\Delta\dot{\boldsymbol{x}}) + \Delta\dot{\boldsymbol{x}}_c)$
end

Algorithm 1: Adaptively rigidifying Newton's method

function

$$Q(\dot{\boldsymbol{x}}_t, \dot{\boldsymbol{x}}_{t+1}) = \dot{\boldsymbol{x}}_{t+1}^T M\left(\frac{1}{2}\dot{\boldsymbol{x}}_{t+1} - \dot{\boldsymbol{x}}_t\right) + \Psi - h\,\dot{\boldsymbol{x}}_{t+1}^T \boldsymbol{f}_{\text{ext}}, \tag{7}$$

where M is the mass matrix, $\boldsymbol{f}_{\text{ext}}$ are the external forces such as gravity, and h is the step size. Here $\dot{\boldsymbol{x}}_t$ is the velocity vector at time t. For each time step we use a Newton-Raphson algorithm to solve for the next step velocities

$$\dot{\boldsymbol{x}}_{t+1} = \underset{\dot{\boldsymbol{x}}_{t+1} \in \mathbb{R}^n}{\arg\min}\, Q(\dot{\boldsymbol{x}}_t, \dot{\boldsymbol{x}}_{t+1}). \tag{8}$$

We use a line search based on Armijo's rule where we simply reduce the Newton iteration's step length until we get an improved solution. We modify the optimization function to instead take as input the reduced system using the G matrix from Equation 5, as shown in Algorithm 1.

In the previous adaptive rigidification work [Mercier-Aubin et al. 2022], speedups were significant for a semi-implicit backward Euler integration method. Increasing the number of Newton iterations further increases the speedup as adaptive rigidification enhances the speed of each iteration, but has a linear overhead only prior to the solve.

3.4 Rigidification

If the membrane strain of an element and the bending deformation with its adjacent elements are constant over a period of time, we allow the element to become rigid. We monitor the change in deformations over several time steps to prevent momentary rigidification such as when a pendulum hits its highest point of swing, or when a cantilever plate hits its potential energy peak. Groups of adjacent elements are concatenated into a single rigid body using a connected component detection algorithm which is equivalent to the adaptive rigidification of 2D meshes.

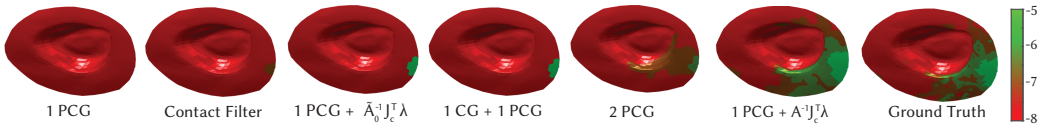

Fig. 3. A tiny upward impulse applied to the tip of the hat elastifies regions according to the oracle's contact handler and different thresholds on a log scale.

We must monitor all deformation changes. For the membrane deformations, we compute the membrane strain rate

$$\dot{E}_{t+1} = \frac{E_{t+1} - E_t}{h}, \tag{9}$$

using finite differences of the Green strain $E = \frac{1}{2}(F^TF - I)$. At the beginning of a simulation, we initialize the cached strain of the previous step to be the identity matrix. We monitor $\dot{E}$, and rigidify the elements of a mesh when they satisfy a threshold over a set of frames. This is enough for two dimensional simulations or for tetrahedral meshes.

For the bending deformations, we could monitor the dihedral angle for every edge with two adjacent elements. But to make the thresholds discretization independent, we instead monitor the discrete curvature

$$\kappa_e = 2\frac{\theta_e}{\bar{h}_e}. \tag{10}$$

Because we already need the dihedral angles for the bending energies, computing the curvature is simply an element-wise multiplication by a constant value for each edge. We then approximate the curvature rates with finite differences

$$\dot{\kappa}_{e,t} = \frac{\kappa_{e,t+1} - \kappa_{e,t}}{h}. \tag{11}$$

For an element to rigidify, it must satisfy both the threshold on the maximum of the per edge curvature rates $\dot{\kappa}_{e,t}$, and the threshold on the membrane strain rate $\dot{E}$.

3.5 Elastification

Rigidified parts of the models must become elastic again on visually significant deformation due to elastic waves, changing contact forces or new contact forces. Within a rigid body, because the elastic degrees of freedom are replaced, we do not have information about the deformation rate of the elastic model when it is solved as rigid. We must predict the parts of a rigidified model that will have a deformation velocity in the next time step.

We can approximate the change in velocities due to elastic motions with a single preconditioned conjugate gradient iteration on the first Newton step with a fully elastic system

$$\Delta\dot{\boldsymbol{x}} = A^{-1}(\boldsymbol{b} - J_c^T\boldsymbol{\lambda}) + h\ddot{\boldsymbol{x}}_g, \tag{12}$$

where $J_c^T\boldsymbol{\lambda}$ are contact forces, A and $\boldsymbol{b}$ are the derivatives of the reduced system as shown in Algorithm 1, and where f_{ext} in Q does not include forces due to gravity. Instead, the gravity velocities $\ddot{\boldsymbol{x}}_g$ are injected into $\boldsymbol{b}$ (i.e., $\dot{\boldsymbol{x}}_t$ modified to be $\dot{\boldsymbol{x}}_t + h\ddot{\boldsymbol{x}}_g$) and also added to the change in velocities after the solve, because this helps the approximate solve produce a better solution.

New or moving contacts can also create elastification. In the oracle, we handle previously existing contacts, and new contacts differently. We concatenate the two types of constraints in $J_c^T\boldsymbol{\lambda}$, and add them to the system to solve. For existing contacts, we reuse the previous adaptive step impulses

as contacts for the oracle. Like the original adaptive rigidification, we approximate only the new contacts for the oracle using a cheap bilateral constraint solve

$$J_{cn}\tilde{A}_0^{-1}J_{cn}^T\boldsymbol{\lambda}_n = J_{cn}(\tilde{A}_0^{-1}\boldsymbol{b} + \dot{\boldsymbol{x}}), \tag{13}$$

where the constraint Jacobian J_{cn} contains only the new contact constraints, and $\tilde{A}_0^{-1}$ is the precomputed 3-by-3 block diagonal inverse matrix at rest (i.e., an approximation of A^{-1} with zero coupling between vertices).

We use a preconditioner following that of Liu et al. [2017] and used by Mercier-Aubin et al. [2022], which is computed as $A_p = M + h^2L$ were $L = B^TWB$ is the Laplacian of the mesh with a block-diagonal weight matrix W, where each block entry contains the per-element Young's modulus. We use a single iteration of a preconditioned conjugate gradient using an incomplete Cholesky factorization of A_p to approximate the velocities due to elastic motions. With the approximated velocities we can use the same formula for the membrane strain rate $\dot{E}$ from Equation 9 and pick a threshold for elastification. Compared to the original adaptive rigidification we also use the approximate velocities to compute the approximate curvature rate $\dot{\kappa_e}$.

The original adaptive rigidification oracle, however, does not always produce a change in velocities significant enough to allow elastification for small localized impulses. This appears to be due to a lack of local propagation, and we speculate that this may stem from how the Laplacian-based preconditioner propagates information globally and changes convergence behaviour during the first iterations of PCG. We suggest a few cheap heuristics to warm-start the oracle with approximate contact velocities from the impulses of the bilateral solve in Equation 13. We evaluate these approaches only for new contacts and assume the old contacts were solved adequately during the time integration.

An intuitive solution for local propagation is to do a single conjugate gradient iteration without preconditioning, and then use the resulting approximate change in velocities due to contacts as an initial iterate $\Delta\dot{x}_0$ for the PCG solve. This only costs an extra linear pass over all the vertices. In a similar train of thought, we can use the precomputed $\tilde{A}_0^{-1}$ block diagonal matrix used for the bilateral solve and obtain approximate contact change in velocities,

$$\Delta\dot{\boldsymbol{x}}_0 = \Delta\dot{\boldsymbol{x}}_c + h\,\boldsymbol{f}_{\text{ext}}, \tag{14}$$

$$\Delta\dot{\boldsymbol{x}}_c = \tilde{A}_0^{-1}J_c^T\boldsymbol{\lambda}, \tag{15}$$

where $\boldsymbol{f}_{\text{ext}}$ is the precomputed external force vector. This option comes at the cost of sparse matrix multiplications with only the new contact constraints, and Lagrange multipliers.

This leads us to yet another strategy. While a single iteration of unpreconditioned conjugate gradient provides information about a local response, we can design an approach with a similar local propagation effect, without the need to iterate through all the vertices. We propose to warm-start the preconditioned conjugate gradient solve using approximate new contact velocities

$$\Delta\dot{x}_c = J_{cl}^T\boldsymbol{\lambda}_l, \tag{16}$$

with impulses $\boldsymbol{\lambda}_l$ obtained with a discrete Laplacian operator for local diffusion of the impulse over edges of the mesh with neighbouring non-colliding vertices. Figure 4 shows an example with a contact force at a center vertex, and the filter weights applied to the patch. We avoid applying the filter on neighbouring elements that are already in contact to avoid cancellation of the impulses for the warm-start when multiple neighbours are in contact. We add rows to the Jacobian J_c that contain the corresponding impulse normals aligned with the degrees of freedom of neighbouring vertices to obtain J_{cl}. This is the cheapest approach, with the cost being a simple sparse matrix multiplication of the Lagrange multipliers with the constraint Jacobian of new contacts.

While none of these approaches fundamentally alter the system to solve, they can significantly improve the accuracy of the approximate one iteration solve. Our warm-start techniques create initial solutions that mimic the mesh's behaviour under compression caused by a new contact with respect to the impact magnitude; a new contact will compress the mesh proportionally to how hard it hits a surface.

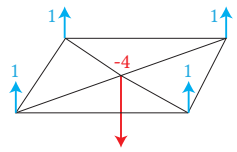

Fig. 4. Contact filter at a degree 4 vertex.

For existing contacts, we reuse the previous adaptive step impulses as contacts for the oracle, ensuring good continuity of the oracle's prediction with respect to the solve for the time integration.

3.6 Threshold Selection

Selecting the appropriate threshold for a simulation is akin to determining the acceptable level of error. In general, smaller thresholds result in a more conservative simulation. To optimize this process, we propose an approach that selects both thresholds simultaneously, using the concept of speed limits to establish a relationship between bending deformations and membrane stretches. However, for inextensible shells and other non-typical materials, decoupling the thresholds may be necessary and advantageous.

To begin, we select the membrane strain rate threshold based on the volumetric threshold of the original adaptive rigidification paper. We set the elastification threshold to be an order of magnitude higher than the rigidification threshold. We can then derive the curvature rate thresholds based on a corresponding relationship. This process ensures a cohesive and accurate simulation for a variety of materials and scenarios.

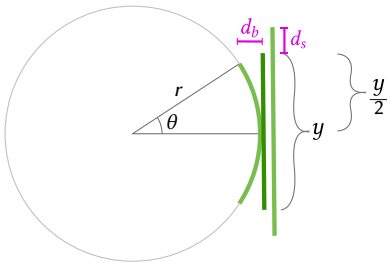

Fig. 5. A green line segment of length y is shown wrapped around a circle with radius r, and stretched in the vertical direction. Equating displacements d_b and d_s leads us to equivalent thresholds for curvature rate and membrane strain rate.

To form a relationship between curvature rate and membrane strain rate thresholds we consider stretching and bending an initially straight vertical line segment as shown in Figure 5. A small membrane stretch increases the length by d_s as measured in the vertical direction from the top and bottom of the original line segment. A small bend from flat to a curvature $\kappa = 1/r$ can be seen as creating a horizontal displacement of d_b at the top and bottom of the line segment when it wraps around a circle of radius r. This horizontal distance $d_b = r - r\cos(\theta)$.

Because the line wraps around the circle, we have

$$r\theta = \frac{y}{2}, \tag{17}$$

or $\theta = \kappa y/2$. If we approximate d_b with a Maclaurin expansion for the cos function up to and including the quadratic term, we have

$$d_b = r - r\left(1 - \frac{\theta^2}{2} + O(\theta^4)\right). \tag{18}$$

Ignoring higher order terms and substituting the angle in Equation 18 using Equation 17 we obtain

$$d_b \approx \frac{y^2}{8r}, \tag{19}$$

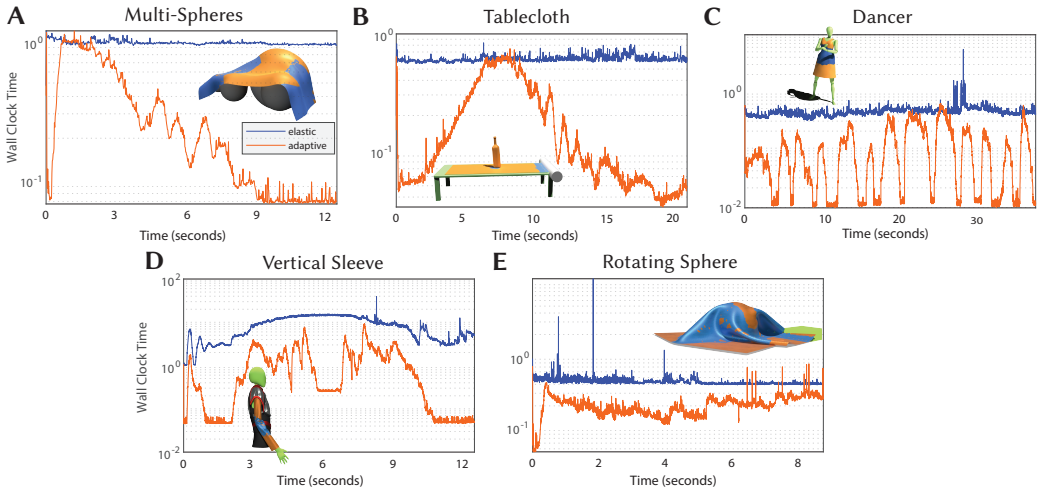

Fig. 6. Set of comparisons all using seconds as the unit for wall clock times.

or $d_b \approx \kappa y^2/8$. Thus, we know how displacement d_b grows as curvature is increased from zero. We can likewise see this as the displacement observed for a curvature rate change of $\dot{\kappa}$ over a small time step h, i.e., $d_b \approx h\dot{\kappa}y^2/8$.

Now, considering a stretch rate of $\dot{s}$ in the vertical direction over a small time step h we have $d_s = h\dot{s}y/2$. Equating d_b and d_s and cancelling terms gives $\dot{\kappa} = \frac{4}{y}\dot{s}$. Here, for simulating arbitrary models, we interpret y as the diameter of the mesh at rest. Finally, because the membrane thresholds are for the squared Frobenius norm of the Green strain rate (i.e., see $\tau_m = \dot{s}^2$ if we choose the stretch rate above to be at the threshold for a mesh at rest), we can compute the curvature rate threshold τ_b in terms of the membrane strain rate threshold τ_m using the formula

$$\tau_b = \frac{4}{y}\sqrt{\tau_m}. \tag{20}$$

The concept of discrete curvature, while resolution independent, can lead to large rates for highly bent elements. This is because the curvature is derived from the inverse radius of the osculating circle, which becomes small for a large bend, producing large curvatures. When dealing with large curvatures, even a slight change in angle on a highly bent edge can generate significant curvature rates. While the resolution-independence is a benefit for measuring curvature rates in meshes with different discretizations, the high rates in areas of high curvature can make the threshold selection challenging. Curvature rates are better suited for fine models where per-edge angles are flatter. In contrast, for a coarse model with elements of roughly consistent size, monitoring dihedral angles is a reasonable alternative, but it is much harder to choose a dihedral angle rate threshold for meshes with different element sizes.

4 RESULTS

Our oracle will produce elastification due to a new contact at a single vertex, and efficiently propagate the impact without expensive extra computation. In Figure 7, for instance, once the brim of the hat hits the floor, we obtain appropriate elastification of the deforming elements near the initial contact. The elements adequately elastify as the elastic wave propagates through the hat, while preserving rigidly moving parts at the opposite side of the model.

Our method allows locally bending or deforming regions to remain elastic while resting contacts and rigidly moving chunks of elements rigidify. For instance, a cloth hanging on a sphere in Figure 6-A has elastic regions where its edges are dangling and has a large central rigidified region where the cloth is in static equilibrium.

We believe our method is the first adaptive technique to completely coarsen densely wrinkled regions while preserving the fine definition of the cloth's curvature. Figure 1 and Figure 6-D show excellent examples of this. These examples feature sleeves with different cloth stiffness parameters, and we observe rigid motion of rigidified wrinkles during arm motions.

We note that care is necessary when setting aggressive (higher) thresholds because this can lead to visual artifacts. For example, there appears to be a small kick near the end of the vertical sleeve simulation partially due to our threshold selection, but also due to the choice of energies pushing the wrinkles downwards back to rest as the arm unbends.

Because our shell implementation natively works with tetrahedra (it uses the same 3 by 3 deformation gradient thanks to the normal projection), we can easily mix elastic solids and shells. In Figure 6-B we present a tablecloth modelled as a shell and wine bottle modelled as a tetrahedral mesh. The tablecloth has a large moving region that remains rigidified while pulled, and likewise demonstrates local deformation near the region stretched due to frictional contacts from both the table and the bottle.

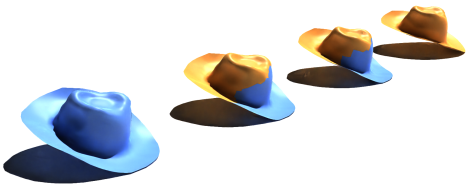

Fig. 7. From left to right: elastic, adaptive with contact filters, adaptive with a CG iteration before the PCG iteration, and the original adaptive rigidification algorithm.

Figure 6-E shows a cloth on a high friction spinning sphere which is inspired by an example of Bridson et al. [2002]. The floor and an adjacent obstacle are frictionless. While large portions of the mesh rigidify as the cloth spins on the sphere, the cloth also continues to exhibit significant dynamics because there is no self contact in the simulation. We believe that adaptive rigidification would work well with both penalty [Bridson et al. 2002] and barrier [Li et al. 2021] based contacts.

4.1 Speed

Table 1 shows performance measurements for the examples from the figures in this paper. The simulations were timed including and excluding contacts to give a fair assessment of the wall clock times. We note some improvements between $2x$ and $13x$ on scenes that were especially designed to generate elastification and dynamic motions. We ran the simulations on a Windows 10 PC with an Intel Core i7-6700K processor, and 64 GB of DDR3 RAM. The simulator is a fork of the publicly available github from the adaptive rigidification project, with the core system in Matlab, and critical snippets implemented in Mex C++.

Adaptive rigidification allows performance improvements more than an order of magnitude faster than the non-adaptive meshes at various steps of the simulations. We also note that in any scene, the overhead of the single PCG iteration and rigid body building remains negligible, even when fully elastic. We present our time comparisons using log scales for fairness. In Figure 1 we see that rigidification can accurately detect local deformation and maintain steady improvements in computation time that is more than an order of magnitude faster than a fully elastic model. The region of deformation is local near the elbow, where wrinkles form and rigidify, hence creating a coarser model of the mesh under rigid motions while preserving the fine details of the wrinkles.

4.2 Conditioning

Mixing rigid bodies with elastic elements increases the mass ratio, leading to poorly conditioned systems during time implicit integration. However, we show that a simple scaling technique [Curtis and Reid 1972] can effectively reduce the condition number of our system, resulting in faster simulations. This scaling $SA\,x = S\,b$ makes the condition number more competitive with the fully elastic system, while also benefiting from the reduced number of degrees of freedom inherent to adaptive rigidification.

While conditioning is generally not a problem for direct solvers, it is beneficial for iterative solvers. Likewise, while Jacobi preconditioning can easily resolve mass ratios, it is not as beneficial for stiffness ratios. Nonlinear elastic materials undergoing large deformations can also exhibit poor conditioning, as demonstrated by the streched cross model in Figure 8-A. Scaling will benefit many iterative solvers, and such solvers may be preferred or needed for bigger scenes. Figure 8-A shows that the Curtis and Reid [1972] scaling (CS) outperforms Jacobi scaling (JS) when it comes to reducing the condition number of the adaptive system. See that the condition number also sharply drops as the rigid chunks increase in size. This suggests that there is significant potential for optimization when using adaptive rigidification in conjunction with larger rigid chunks, and poorly shaped elements.

In Figure 8-B we compare the time for each step of the stretched cross simulation example using a direct solve via LDL decomposition and scaling to that of a PCG solve with scaling and tolerance of $1e-4$. For efficiency, our PCG solve reuses the oracle's preconditioner with kinematic mapping to the coupled rigid-elastic system, i.e., $SG^T A_p G$. As expected, stopping the iterative solver before full convergence provides big savings in the computation time. Nevertheless, while large systems can benefit from an iterative solver, it can be beneficial to switch to a direct solver for the small systems that arise when there is extensive rigidification.

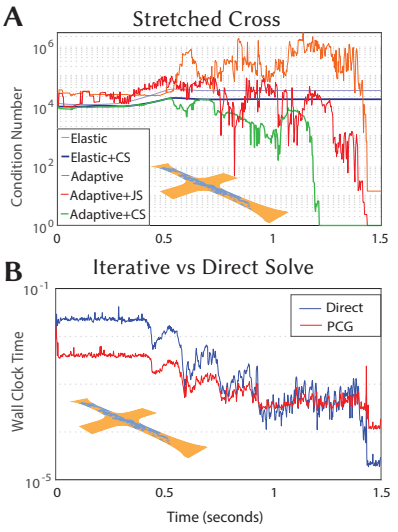

Fig. 8. A: comparing condition numbers with or without scaling. B: comparing a direct solver to an iterative solver.

4.3 Strain Limiting

Strain limiting prevents deformation above or below a strain limit. This models physical behaviour of deformable objects acting almost rigidly when under heavy loads. This has the added benefit of reducing the change in deformation, further increasing the rate of rigidification, and the speed of the simulation. We implement strain limiting with the singular value decomposition of $F = USV^T$ and clamp the principal stretches $s \in S$ like Wang et al. [2010]. This formulation of strain limiting is particularly elegant as it allows us to reuse the singular value decomposition when computing materials like the corotational energies. Moreover, we can save on the SVD computation for the strain limiting of rigidified elements as they are non-deforming.

By limiting the strain, we also limit the rates of deformations, hence increasing the rate of rigidification. Figure 6-A shows a piece of cloth with strain limiting where singular values are clamped between 0.90 and 1.1. A large patch of stretched cloth quickly rigidifies in the middle of the spheres, while motion is allowed where needed. The final result is visually indistinguishable from the fully elastic simulation as we present it in the supplemental video.

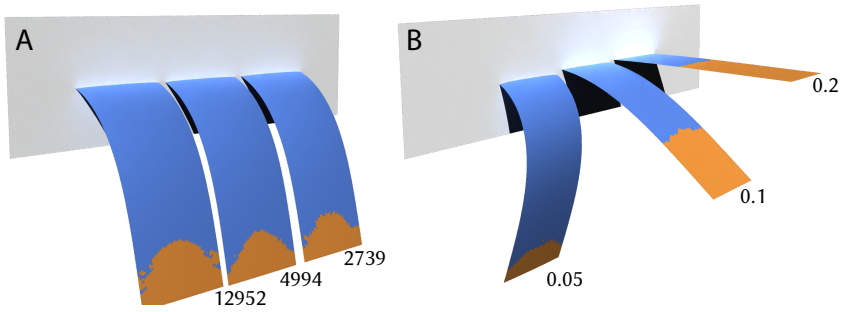

Fig. 9. A: The rigidification patterns are consistent across resolutions. We show the total number of elements in the shell cantilevers next to them. B: The simulation retains its quality regardless of the material properties like the thickness. We list the respective thickness of each shell cantilever next to them.

In Figure 6-C, we chose a crinolette type of dress for the dancer to show that rigidification natively handles stiff cloth/plastic even when the strain limits are reached. The adaptive approach is also significantly faster than the elastic simulation at almost any point in time. The mannequin model and dress are from Narain et al. [2012].

4.4 Oracle

In Figure 9-A, we see that using our bending threshold correspondence yields similar rigidification patterns independent of the resolutions of our meshes. This holds across all of our scenes. We can also pick custom bending thresholds for incompressible materials or if fine tuning is needed. The rigidification process is material independent as it relies on the speed of deformation. Therefore, choosing the thresholds for the membrane stretches, and then using our equivalence for the bending thresholds yield a consistent quality of simulation regardless of the material used. In Figure 9-B, we compare different material thicknesses and note that the regions of elastification intuitively match their regions of deformation.

We compare various approaches to improve the new contact elastification regions in Figure 3. Using the inverse matrix A^{-1} to find the contact velocities from the bilateral solve's impulses yields the closest elastification region to the ground truth of elastic propagation. However such approach is overly costly. The contacts from the bilateral solve are already approximations, and the oracle's contact velocities need not be accurate. We only require a solution that generates enough elastification to preserve momentum under contact. A reasonable alternative is to instead use the precomputed block diagonal inverse matrix $\tilde{A}_0^{-1}$ which has a notion of the geometry of our system at rest. Using this method yields an elastification region equivalent to doing a single CG iteration prior to the PCG iteration. Using two PCG iterations gives slightly worse local propagation, but better overall distant elastification.

In practice, most of the momentum loss on contact comes from simulating a contact as rigid when it should be deforming. The fast contact filter generates just enough elastification to handle typical simulations well, while the precomputed diagonal matrix multiplication is more appealing for fine contact handling like the soft touch of a feather. In Figure 7 the contact filter allows visually consistent simulation of the hat even with a small impact, and only requires the diffusion of an impulse over a few vertices as opposed to a CG iteration over all degrees of freedom or a sparse matrix multiplication with A. We see that using only a single PCG iteration without any of our contact handling approaches does not allow elastification of the hat on first contact, which leads to a loss in momentum.

Table 1. We generated and simulated scenes with varying numbers of elements, and provide the full simulation times, as well as the simulation times without contacts (NC).

Scene	Vertices	Elements	NC Time Adaptive(s)	NC Time Elastic(s)	Time Adaptive(s)	Time Elastic(s)	Speedup Factor
Dancer	2000	3861	289.32	977.24	815.51	3318.06	3.37x
Multi-Spheres	8437	16493	222.13	682.05	3653.78	14230.15	3.07x
Tablecloth	9896	20440	483.23	2433.54	10629.65	34496.75	5.04x
Horizontal Sleeve	2450	4830	79.66	1060.95	10322.95	43116.48	13.31x
Hat	1412	2760	12.98	18.49	28.20	42.04	1.42x
Vertical Sleeve	7777	15488	916.96	5524.75	5722.66	41937.27	6.03x
Rotating Sphere	4602	8932	707.48	1546.10	21398.00	47070.00	2.19x

Our approach is designed to work with any standard collision detection and handling techniques. We have implemented bounding volumes, signed distance functions, and use projected Gauss-Seidel to solve for contact impulses. We can see from Table 1 that the rigidification has an impact on collision handling times as it speeds up the assembly of the Delassus operator for a PGS solver. Other approaches such as that of Verschoor and Jalba [2019] would likely retain similar collision handling speeds regardless of rigidification.

5 DISCUSSION AND LIMITATIONS

Although our approach is currently efficient for wrinkled, non-deforming parts, our approach lacks the ability to remesh actively deforming regions. Nonetheless, we believe that adaptive rigidification of shells could complement remeshing, resulting in an even more efficient oracle by reducing the size of our linear pass over elements, as well as the coarsening of actively deforming elements in wrinkle-free areas. Speedups are dependent on the thresholds (level of accuracy), and the dynamics of the scenes, as opposed to remeshing where the speedups are dependant on the quality of the coarsening, and the continuous shape of the model. While adaptive rigidification cannot coarsen a flat actively stretching piece of cloth, remeshing would be able to simulate this deformation with a low number of degrees of freedom. Likewise, remeshing cannot coarsen dense triangle folds to a constant number of degrees of freedom as we do when the folds are moving rigidly.

Our approach to handle new contact reduces the likelihood of missing elastification, but some motions can still be problematic. A slow constant creep type of motion that accumulates over time while remaining below the elastification threshold would cause deformation on a fully elastic model, but could not deform a rigidified body. However, such cases are perhaps rare and can otherwise be addressed by adjusting thresholds or designing new custom solutions.

While not directly related to rigidification, our contact handling is slow for large number of contacts because the assembly of the Delassus operator creates a bottleneck. Using a contact handling method like that of Verschoor and Jalba [2019] does not require this assembly and would greatly speed up the simulations. There is also an opportunity to prune contacts on rigid bodies by considering only 3 contacts per body, and diffusing the impulses on the rigid body for the oracle's elastic contacts to reduce the number of constraints during the time integration. Likewise, the internal forces of rigidified elements could be cached on rigidification, and rotated with the rigid body properties to save on the computation of the energy derivatives.

The BFS algorithm to build rigid bodies is the same as the original adaptive rigidification with a minor modification to sequentially iterate through mixed geometry. Using parallel algorithms like

the ones described by Zhang et al. [2020] to find connected components would further reduce the overhead of adaptive rigidification.

Finally, we note that poorly conditioned elements will likely create large approximate changes in velocity. Thus, an adaptive threshold that considers conditioning could be created to uniformly set the tolerance across the meshes.

6 CONCLUSIONS

In this paper, we present an extension of adaptive rigidification to target thin shells. By adding a second rigidification criterion based on the curvature rate, we achieve the benefits of rigidification with minimal overhead, similar to the tetrahedral version. Our approach leverages computations from different parts of the simulation, such as the dihedral angles of the bending energy, so as to reduce redundant work and improve efficiency. To further enhance the performance of the elastification oracle in the presence of small contact patches, we incorporate a fast contact filter, and explore other valuable approaches for diffusing contact information during the oracle solve. Finally, we demonstrate that scaling the per-Newton step system can significantly reduce the condition number, making it competitive with that of the fully elastic scaled system while also benefiting from its reduced size.

7 ACKNOWLEDGMENT

This research was funded by the FRQNT Doctoral scholarship 332127. We are grateful to the Fonds de recherche du Québec for their resources. We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) via the Discovery grant program and an Alliance grant with Symgery.

REFERENCES

Pierre Alliez, Giuliana Ucelli, Craig Gotsman, and Marco Attene. 2008. *Recent Advances in Remeshing of Surfaces.* Springer Berlin Heidelberg, Berlin, Heidelberg, 53–82. https://doi.org/10.1007/978-3-540-33265-7_2

Svetlana Artemova and Stephane Redon. 2012. Adaptively Restrained Particle Simulations. *Phys. Rev. Lett.* 109 (2012), 190201. Issue 19.

David Baraff and Andrew Witkin. 1998. Large Steps in Cloth Simulation. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '98).* Association for Computing Machinery, New York, NY, USA, 43–54. https://doi.org/10.1145/280814.280821

Jan Bender and Crispin Deul. 2012. Efficient Cloth Simulation Using an Adaptive Finite Element Method. In *Virtual Reality Interactions and Physical Simulations (VRIPhys).* Eurographics Association, Darmstadt, Germany, 21–30. https://doi.org/10.2312/PE/vriphys/vriphys12/021-030

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

Morten Bro-Nielsen and Stephane Cotin. 1996. Real-time Volumetric Deformable Models for Surgery Simulation using Finite Elements and Condensation. *Computer Graphics Forum* 15, 3 (1996), 57–66. https://doi.org/10.1111/1467-8659.1530057 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/1467-8659.1530057

Fehmi Cirak, Michael Ortiz, and Peter Schröder. 2000. Subdivision surfaces: a new paradigm for thin-shell finite-element analysis. *Internat. J. Numer. Methods Engrg.* 47, 12 (2000), 2039–2072. https://doi.org/10.1002/(SICI)1097-0207(20000430)47:12<2039::AID-NME872>3.0.CO;2-1

Eulalie Coevoet, Otman Benchekroun, and Paul G. Kry. 2020. Adaptive Merging for Rigid Body Simulation. *ACM Trans. Graph.* 39, 4, Article 35 (2020), 12 pages.

A. R. Curtis and J. K. Reid. 1972. On the Automatic Scaling of Matrices for Gaussian Elimination. *IMA Journal of Applied Mathematics* 10, 1 (08 1972), 118–124. https://doi.org/10.1093/imamat/10.1.118

Elliot English and Robert Bridson. 2008. Animating Developable Surfaces Using Nonconforming Elements. *ACM Trans. Graph.* 27, 3 (aug 2008), 1–5. https://doi.org/10.1145/1360612.1360665

Kenny Erleben. 2004. *Stable, robust, and versatile multibody dynamics animation.* Ph. D. Dissertation. University of Copenhagen.

Olaf Etzmub, Michael Keckeisen, and Wolfgang Straber. 2003. A Fast Finite Element Solution for Cloth Modelling. In *Proceedings of the 11th Pacific Conference on Computer Graphics and Applications (PG '03)*. IEEE Computer Society, USA, 244.

Rony Goldenthal, David Harmon, Raanan Fattal, Michel Bercovier, and Eitan Grinspun. 2007. Efficient Simulation of Inextensible Cloth. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007)* 26, 3 (2007), to appear.

Eitan Grinspun, Anil N. Hirani, Mathieu Desbrun, and Peter Schröder. 2003. Discrete Shells. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (San Diego, California) *(SCA '03)*. Eurographics Association, Goslar, DEU, 62–67.

Eitan Grinspun, Petr Krysl, and Peter Schröder. 2002. CHARMS: A Simple Framework for Adaptive Simulation. *ACM Trans. Graph.* 21, 3 (2002), 281–290.

Fabian Hahn, Bernhard Thomaszewski, Stelian Coros, Robert W. Sumner, Forrester Cole, Mark Meyer, Tony DeRose, and Markus Gross. 2014. Subspace Clothing Simulation Using Adaptive Bases. *ACM Trans. Graph.* 33, 4, Article 105 (jul 2014), 9 pages. https://doi.org/10.1145/2601097.2601160

Xiangmin Jiao, Andrew Colombi, Xinlai Ni, and John C. Hart. 2006. Anisotropic Mesh Adaptation for Evolving Triangulated Surfaces. In *Proceedings of the 15th International Meshing Roundtable*, Philippe P. Pébay (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 173–190.

Ladislav Kavan, Dan Gerszewski, Adam W. Bargteil, and Peter-Pike Sloan. 2011. Physics-Inspired Upsampling for Cloth Simulation in Games. In *ACM SIGGRAPH 2011 Papers* (Vancouver, British Columbia, Canada) *(SIGGRAPH '11)*. Association for Computing Machinery, New York, NY, USA, Article 93, 10 pages. https://doi.org/10.1145/1964921.1964988

David I.W. Levin. 2020. Physics-based animation lecture 6: Cloth Simulation.

Ling Li and Vasily Volkov. 2005. Cloth Animation with Adaptively Refined Meshes. In *Proceedings of the Twenty-Eighth Australasian Conference on Computer Science - Volume 38* (Newcastle, Australia) *(ACSC '05)*. Australian Computer Society, Inc., AUS, 107–113.

Minchen Li, Danny M. Kaufman, and Chenfanfu Jiang. 2021. Codimensional Incremental Potential Contact. *ACM Trans. Graph. (SIGGRAPH)* 40, 4, Article 170 (2021).

Tiantian Liu, Sofien Bouaziz, and Ladislav Kavan. 2017. Quasi-Newton Methods for Real-Time Simulation of Hyperelastic Materials. *ACM Trans. Graph.* 36, 4, Article 116a (jul 2017), 16 pages. https://doi.org/10.1145/3072959.2990496

Charles Loop. 1987. *Smooth Subdivision Surfaces Based on Triangles.* Ph. D. Dissertation. The University of Utah.

Pierre-Luc Manteaux, François Faure, Stéphane Redon, and Marie-Paule Cani. 2013. Exploring the Use of Adaptively Restrained Particles for Graphics Simulations. In *Workshop on Virtual Reality Interaction and Physical Simulation*, Jan Bender, Jeremie Dequidt, Christian Duriez, and Gabriel Zachmann (Eds.). The Eurographics Association, Lille, France, 17–24. https://doi.org/10.2312/PE.vriphys.vriphys13.017-024

Alexandre Mercier-Aubin, Alexandre Winter, David I. W. Levin, and Paul G. Kry. 2022. Adaptive Rigidification of Elastic Solids. *ACM Trans. Graph.* 41, 4, Article 71 (jul 2022), 11 pages. https://doi.org/10.1145/3528223.3530124

E. Miguel, D. Bradley, B. Thomaszewski, B. Bickel, W. Matusik, M. A. Otaduy, and S. Marschner. 2012. Data-Driven Estimation of Cloth Simulation Models. *Comput. Graph. Forum* 31, 2pt2 (may 2012), 519–528.

Matthias Müller and Nuttapong Chentanez. 2010. Wrinkle Meshes. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Madrid, Spain) *(SCA '10)*. Eurographics Association, Goslar, DEU, 85–92.

Rahul Narain, Armin Samii, and James F. O'Brien. 2012. Adaptive Anisotropic Remeshing for Cloth Simulation. *ACM Trans. Graph.* 31, 6, Article 152 (2012), 10 pages.

Tiberiu Popa, Qingnan Zhou, Derek Bradley, Vladislav Kraevoy, Hongbo Fu, Alla Sheffer, and Wolfgang Heidrich. 2009. Wrinkling Captured Garments Using Space-Time Data-Driven Deformation. *Computer Graphics Forum (Proc. Eurographics)* 28, 2 (2009), 427–435.

Xavier Provot. 1995. Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behaviour. In *Proceedings of Graphics Interface '95* (Quebec, Quebec, Canada) *(GI '95)*. Canadian Human-Computer Communications Society, Toronto, Ontario, Canada, 147–154. http://graphicsinterface.org/wp-content/uploads/gi1995-17.pdf

Damien Rohmer, Tiberiu Popa, Marie-Paule Cani, Stefanie Hahmann, and Alla Sheffer. 2010. Animation Wrinkling: Augmenting Coarse Cloth Simulations with Realistic-Looking Wrinkles. *ACM Trans. Graph.* 29, 6, Article 157 (dec 2010), 8 pages. https://doi.org/10.1145/1882261.1866183

H. Schmidl and V. J. Milenkovic. 2004. A fast impulsive contact suite for rigid body simulation. *IEEE Transactions on Visualization and Computer Graphics* 10, 2 (2004), 189–197.

Rasmus Tamstorf and Eitan Grinspun. 2013. Discrete bending forces and their Jacobians. *Graphical Models* 75, 6 (2013), 362–370. https://doi.org/10.1016/j.gmod.2013.07.001

Maxime Tournier, Matthieu Nesme, Francois Faure, and Benjamin Gilles. 2014. Seamless Adaptivity of Elastic Models. In *Proceedings of Graphics Interface 2014* (Montreal, Quebec, Canada) *(GI '14)*. Canadian Information Processing Society, CAN, 17–24.

Mickeal Verschoor and Andrei C. Jalba. 2019. Efficient and Accurate Collision Response for Elastically Deformable Models. *ACM Trans. Graph.* 38, 2, Article 17 (mar 2019), 20 pages. https://doi.org/10.1145/3209887

Huamin Wang, James O'Brien, and Ravi Ramamoorthi. 2010. Multi-Resolution Isotropic Strain Limiting. *ACM Trans. Graph.* 29, 6, Article 156 (dec 2010), 10 pages. https://doi.org/10.1145/1882261.1866182

Martin Wicke, Daniel Ritchie, Bryan M. Klingner, Sebastian Burke, Jonathan R. Shewchuk, and James F. O'Brien. 2010. Dynamic Local Remeshing for Elastoplastic Simulation. *ACM Trans. Graph.* 29, 4, Article 49 (jul 2010), 11 pages. https://doi.org/10.1145/1778765.1778786

Yongzhe Zhang, Ariful Azad, and Aydın Buluç. 2020. Parallel algorithms for finding connected components using linear algebra. *J. Parallel and Distrib. Comput.* 144 (2020), 14–27. https://doi.org/10.1016/j.jpdc.2020.04.009