Adaptive Rigidification of Elastic Solids

ALEXANDRE MERCIER-AUBIN, McGill University, Canada
ALEXANDER WINTER, McGill University, Canada
DAVID I.W. LEVIN, University of Toronto, Canada
PAUL G. KRY, McGill University, Canada

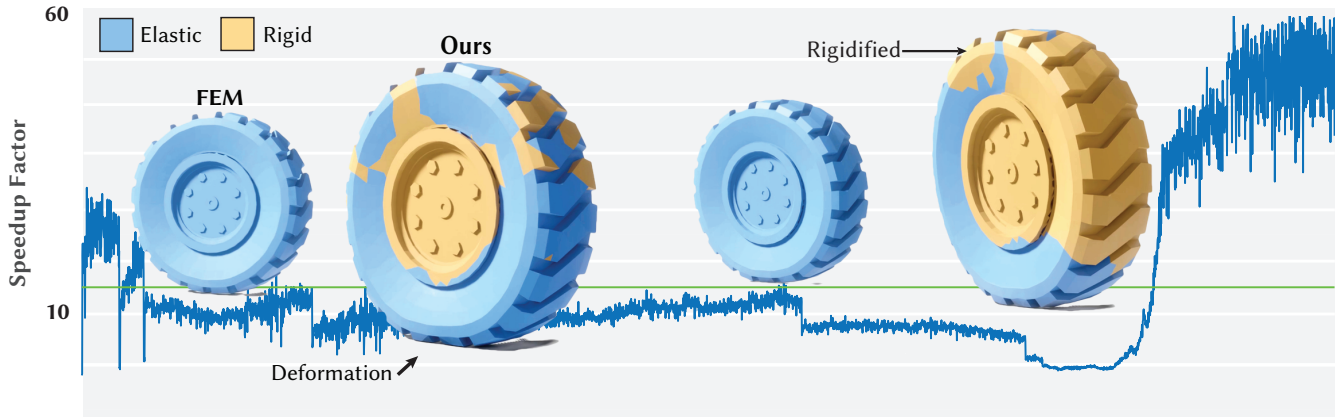


Fig. 1. Our algorithm can identify and adaptively rigidify undeforming portions of simulated elastic objects in order to improve performance without sacrificing visual fidelity. Per-step computation time for this tire simulation, with rubber tread and steel hub, shows a mean performance improvement of 10×, resulting in a 5× reduction in total simulation time.

We present a method for reducing the computational cost of elastic solid simulation by treating connected sets of non-deforming elements as rigid bodies. Non-deforming elements are identified as those where the strain rate squared Frobenius norm falls below a threshold for several frames. Rigidification uses a breadth first search to identify connected components while avoiding connections that would form hinges between rigid components. Rigid elements become elastic again when their approximate strain velocity rises above a threshold, which is fast to compute using a single iteration of conjugate gradient with a fixed Laplacian-based incomplete Cholesky preconditioner. With rigidification, the system size to solve at each time step can be greatly reduced, and if all elastic element become rigid, it reduces to solving the rigid body system. We demonstrate our results on a variety of 2D and 3D examples, and show that our method is likewise especially beneficial in contact rich examples.

CCS Concepts: • **Computing methodologies** → **Physical simulation**.

Authors' addresses: Alexandre Mercier-Aubin, alexandre.mercier-aubin@mail.mcgill.ca, McGill University, Montréal, Canada; Alexander Winter, alexandre.winter@mail.mcgill.ca, McGill University, Montréal, Canada; David I.W. Levin, diwlevin@cs.toronto.edu, University of Toronto, Toronto, Canada; Paul G. Kry, kry@cs.mcgill.ca, McGill University, Montréal, Canada.

0730-0301/2022/7-ART71 $15.00
https://doi.org/10.1145/3528223.3530124

Additional Key Words and Phrases: adaptive, finite element method, rigid bodies, physics simulation

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

1 INTRODUCTION

Physics-based animation produces complex emergent behavior and motion from relatively compact physical laws. Prescribing a handful of physical properties to an object can generate animations of phenomena as wide ranging as realistic dripping honey, to an immense castle collapsing, to the delightful bounciness of a cartoon character.

In solid mechanics, parameters such as the Young's modulus and Poisson's ratio determine the effective compliance of an object relative to its environment. For any finite, realistic choice of these parameters, there are scenarios during which an object will drastically deform, and others during which it will move rigidly through the world. Imagine squishing a sponge in your hand. During compression the sponge changes shape drastically, but if you swing your clenched fist, the sponge moves rigidly in its new deformed state. Complicating things further is the spatially varying nature of this effect. For large objects, local external forces cause deformation that quickly decays to rigid motion as distance from the loading point increases. Despite these effects being well-known, for the past 40 years, physics-based animation algorithms have forced practitioners to decide a-priori if an object should be modeled as a rigid body or a deformable one – regardless of the applied loads

it may experience. The goal of this work is to free users from this forced choice.

Simulating large deformations requires a suitably expressive kinematic map with sufficient degrees-of-freedom to model shape change. This comes with an unavoidable performance overhead. In contrast, purely rigid deformation can be represented compactly by an isometric transformation. This enables fast simulation, meaning large complicated scenes are often simulated using rigid bodies due to performance considerations. In doing so, they give up on potentially interesting behavior due to deformations. Ideally, simulation algorithms should apply the appropriate mapping based on the physical properties of the system and its underlying interactions, not based on user intuition. Rigid sections of objects should be simulated using the compact, performant, rigid body mapping while parts of the scene undergoing deformation should be upgraded to a deformable model. We should be able to have our simulated cake and eat it too.

In this paper we present the first hierarchy-free algorithm for the elastodynamic simulation of deformable objects that can dynamically transition between rigid and deformable kinematic models at runtime. Our method maintains two dynamically evolving, spatially varying kinematic maps, one rigid and one deformable. We introduce a deformation velocity metric that predicts which parts of an object can be represented as rigid bodies, and therefore integrated efficiently using geometric methods, leading to fast aggregate simulation times for complex scenes. Additionally, we devise an inexpensive method for estimating which parts of an object should transition between rigid and deformable states in the presence of transient forces such as contact and friction. Crucially we accomplish this without requiring predefined hierarchies or introducing additional constraints on the geometry or physical parameters of the simulation.

With our method for on-demand elastification, physics simulation of solid geometry becomes properly input-sensitive. The rigid/deformable modeling decision becomes a function of physical parameters and environmental interactions rather than a guess made prior to runtime. By putting the physics first, our algorithm reduces user burden, avoids filtering away salient emergent behavior and expedites computation of results. In what follows we detail our adaptive approach to the modeling and simulation of elastic, deformable objects and show that under a number of commonly encountered scenarios our method yields significant speed-ups over purely deformable finite element simulations.

2 RELATED WORK

Degree-of-freedom reduction is a common technique for accelerating physics-based animation. Modal analysis [Barbič and James 2005] projects the equations of motion into a reduced linear space while frame-based approaches [Gilles et al. 2011] replace dense volumetric discretizations with sparse skinning handles. Finally, numerical coarsening [Chen et al. 2017; Kharevych et al. 2009; Nesme et al. 2009] allows simulations to produce results, using a low resolution volumetric discretization to produce results commensurate with a more expensive, high resolution one. While these methods can produce significant speed-ups, they have two fundamental limitations. First, they are applied during the modeling phase, meaning they do not and cannot take into account environmental stimulus, only the geometry and material properties of the model, in isolation. Second they do not naturally progress to the completely rigid case, rather they approach it, but remain deformable, always including a few additional, potentially unnecessary degrees of freedom.

Runtime approaches attempt to modify the number of degrees-of-freedom as the simulation progresses. The simplest of such approaches are freezing methods, so named because they deactivate, or freeze, degrees-of-freedom when they are deemed unnecessary to evolve the system in time [Artemova and Redon 2012; Manteaux et al. 2013]. In contrast to these methods, which freeze in the inertial frame, our approach permits rigid motion of components with degrees-of-freedom freezing only relative to one another. Freezing is also popular in rigid body simulations [Erleben 2004; Schmidl and Milenkovic 2004], and more robust sleeping approaches have been developed for contacting rigid bodies [Coevoet et al. 2020], but these do not directly apply to the deformable bodies we study here.

Rather than deactivate degrees-of-freedom entirely, they can instead be adaptively down sampled. Early variants of this approach actually worked in reverse – they assume a coarse simulation mesh which was refined as a pre-process to create a fixed hierarchy [Grinspun et al. 2002]. At runtime this hierarchy could be traversed to locally enhance detail [Debunne et al. 2001]. However the availability of a coarse mesh should not be assumed, and it is not always possible to refine back to the input high-resolution geometry. To address this problem, fixed hierarchy approaches have evolved to act on embedded mesh [Nesme et al. 2009] and frame-based [Tournier et al. 2014] simulations. These approaches use relatively coarse discretizations for even the finest levels of their hierarchies, meaning that the full motion of an object can never be resolved. Additionally, fixed hierarchies limit the location and amount of refinement that can take place.

Chen et al. [2017] presents a special cases of the hierarchical approach, using a two-level hierarchy where the root is a rigid transform [Terzopoulos and Witkin 1988] and the second level is a hexahedral simulation mesh. They transition to using the rigid transformation only when elastic potential energy goes to zero, which misses the opportunity for rigidification in other equilibrium states (e.g., resting contact). Modal hierarchies have also been explored [Kim and James 2009; Teng et al. 2015]. These are close in spirit to our approach but require the precomputation of a modal basis and can have trouble using the reduced basis in the presence of large rotational motions. Adaptive remeshing [Narain et al. 2012; Schreck et al. 2016] combats this issue by using geometric operations to add and remove degrees-of-freedom from the simulation mesh. However mesh operations are complex, difficult to implement and are not typically capable of reducing to pure rigid motion as our method does.

Our approach is not based on a hierarchy, but on connected components. Lack of a fixed hierarchy gives us maximum flexibility in terms of when to treat parts of an object as rigid or deformable. This means our algorithm can collapse an arbitrarily complex deformable object to a rigid body if permitted. While methods for mixing simulations of rigid and elastic parts have been previously proposed [Jansson and Vergeest 2003; Lenoir and Fonteneau 2004;

Algorithm 1: Main loop

$Jc, \Phi \leftarrow$ Find contacts	// §3.3
$\lambda \leftarrow$ WarmStart, approximate for new contacts	// §3.3
$\Delta\dot{\boldsymbol{x}}_{\text{approx}}, \dot{E}_{\text{approx}} \leftarrow$ QuickSolve	// §3.5
$\dot{E} \leftarrow$ Compute strain rates	// §3.4
BFS to identify rigid components	// §3.4
$M_{\text{R}} \leftarrow$ Compute rigid properties	// §3.4
$\Delta\dot{\boldsymbol{x}}_{\text{A}} \leftarrow$ LDLT Solve	// Eq. 11
$\Delta\dot{\boldsymbol{x}}_c \leftarrow$ Contact Solve	// Eqs. 14-15
Update velocities and positions	

Yang et al. 2010], we introduce a relative deformation metric which allows rigidification of regions of an object that are deformed but moving rigidly, as well as an efficient way to elastify local parts of rigid sections in response to deformation caused by external loads including contact and friction. These contributions, taken together result in an adaptive scheme for on-demand input-sensitive elasticity that is both more flexible, and more performant than prior art.

3 ELASTIC AND RIGID SIMULATION

We will start with a brief review, describing how we set up our elastic finite element model simulation. We use tetrahedral meshes with linear shape functions and the standard semi-implicit backward Euler approach for numerical integration [Baraff and Witkin 1998]. Following this, we will introduce how the formulation changes when portions of the elastic solid are rigid, and how we handle contact. This will lead us to the problem of identifying what parts of the mesh should be rigid, and when rigid parts should become elastic again, which we will describe in Sections 3.4 and 3.5. An overview of the main loop of our method is shown in Algorithm 1.

3.1 Simulating Finite Elements

We define matrix B such that the deformation gradient may be computed as $F = B\,\boldsymbol{x}$, where $\boldsymbol{x} \in \mathbb{R}^{3n_v}$ contains the positions of the n_v vertices of the finite element model. While the deformation gradient of each element is a matrix, we pack column vector $F \in \mathbb{R}^{9n_e}$ by stacking vertically the deformation gradients of all n_e elements in column order. Using tetrahedral elements and barycentric interpolation as a shape function, the deformation gradient is constant across each element (otherwise, B can be seen as computing F at quadrature points). We compute the infinitesimal elastic energy ψ as a function of the deformation gradient F. For a given element i with rest volume V_i and deformation gradient F_i we compute element-integrated PK1 stress and stiffness

$$P_i = -V_i \frac{\partial \psi}{\partial F_i}^T, \tag{1}$$

$$C_i = -V_i \frac{\partial^2 \psi}{\partial F_i^2}^T. \tag{2}$$

We assemble the vector $\boldsymbol{P} \in \mathbb{R}^{9n_e}$ containing the element-integrated stress of all n_e elements in column order, and the sparse block matrix C with the 9-by-9 blocks C_i, which permits us to write the nodal forces and sparse stiffness matrix,

$$\boldsymbol{f} = B^T \boldsymbol{P}, \tag{3}$$

$$K = B^T C B, \tag{4}$$

where $\boldsymbol{f} \in \mathbb{R}^{3n_v}$ and $K \in \mathbb{R}^{3n_v \times 3n_v}$.

Thus, the system we solve for semi-implicit backward Euler integration is

$$\underbrace{(M - hD - h^2K)}_{A}\,\Delta\dot{\boldsymbol{x}} = h(D\dot{\boldsymbol{x}} + \boldsymbol{f} + hK\dot{\boldsymbol{x}} + \boldsymbol{f}_{\text{ext}}), \tag{5}$$

where we use a lumped mass matrix M, Rayleigh damping $D = \alpha_0 M + \alpha_1 K$, and $\boldsymbol{f}_{\text{ext}}$ are external forces such as gravity and user interaction forces (contact forces are discussed in Section 3.3). We use an LDLT factorization of matrix A to solve for the change in velocities $\Delta\dot{\boldsymbol{x}}$ and then update the velocities and subsequently the positions with the updated velocities.

For heterogeneous materials, we build the damping matrix D as a sum of a mass damping matrix M_d and stiffness damping matrix K_d. We build the stiffness damping matrix with α_1 weighted diagonal blocks, that is, $K_d = B^T C_d B$ with each block of C_d being $\alpha_{1i}\, C_i$. For the Rayleigh mass damping matrix we lump the mass damping property in the same way that we lump the mass. That is, for element i with density ρ_i we distribute $1/4\,\alpha_{0i}\, V_i\, \rho_i$ to each vertex making up the tetrahedral element.

3.2 Mixing Rigid and Elastic DOFs

During the simulation of an elastic solid, there may be extended periods of time where portions of the model are moving rigidly. This occurs trivially when an object is at rest in static equilibrium, but can also happen with non-zero linear and rotational velocity during flight or sliding contact. We can treat as rigid the regions of a model that have a zero strain rate for a period of time. While we discuss the rigidification process and related issues in Section 3.4, we will first present how we set up the equations of motion for a mixed elastic and rigid simulation.

For simplicity, let us consider the case with a single rigid body. Let $\mathcal{R}$ be the set of vertex indices that make up the rigid body. The simulation state of the body will consist of a position p and an orientation $R \in SO(3)$ along with a linear and angular velocity $\phi = (v^T \omega^T)^T$. Following the form of Equation 5 we can write the rigid body equation of motion as

$$M_{\text{R}} \Delta\phi = h(c(\phi) + w_{\text{ext}}), \tag{6}$$

where M_{R} is the 6-by-6 mass matrix with rotational inertia submatrix rotated into the world aligned frame given the body's current orientation, $c(\phi)$ are the velocity-depended rigid body torques, and w_{ext} are external forces such as gravity and user interaction.

When a portion of the elastic mesh is made rigid, we store the positions of vertices making up the rigid body in the rigid body frame. Letting r_i for $i \in \mathcal{R}$ be the rigid vertex positions in coordinates of the rigid body frame, we can compute the positions and velocities of these vertices in the world frame as $x_i = R\,r_i + p$ and $\dot{x}_i = -(Rr_i) \times \omega + v$. This second expression can be written instead as a

matrix product,

$$\dot{x}_i = \underbrace{\begin{bmatrix} I & -(R\,r_i)^\times \end{bmatrix}}_{\Gamma_i} \begin{bmatrix} v \\ \omega \end{bmatrix}, \tag{7}$$

where $(\cdot)^\times$ denotes construction of the 3-by-3 cross product operator. Furthermore, we can now write the velocity of all vertices in the finite element model as a product of a matrix G with velocities of our active (elastic and rigid) degrees of freedom, $\dot{\boldsymbol{x}}_\mathrm{A}$,

$$\dot{\boldsymbol{x}} = \underbrace{\begin{bmatrix} I & 0 \\ 0 & \Gamma \end{bmatrix}}_{G} \underbrace{\begin{bmatrix} \dot{\boldsymbol{x}}_\mathcal{A} \\ \dot{\phi} \end{bmatrix}}_{\dot{x}_\mathrm{A}}. \tag{8}$$

Here, $x_\mathcal{A}$ are the active elastic vertices (i.e., those which are not part of a rigid body), and for simplicity we assume that these vertices have lower indices and are copied by the identity block of G, while the Γ block is a stack of all Γ_i for $i \in \mathcal{R}$. When there are many rigid bodies, the lower part of matrix G and vector $\dot{\boldsymbol{x}}_\mathrm{A}$ grow accordingly.

With some of the elements being simulated as a rigid body, we only need to do the elastic solve for the set of elements which are still elastic. Let $\mathcal{E}$ be the set of elastic element degrees of freedom indices, and notice that we can compute the deformation gradient of only these elements as the product $F_\mathcal{E} = B_\mathcal{E} G \boldsymbol{x}_\mathrm{A}$, where matrix $B_\mathcal{E}$ consists of only those rows of B that correspond to the elastic elements. The elastic force and stiffness for active degrees of freedom can therefore be written

$$\boldsymbol{f}_\mathrm{A} = G^T B_\mathcal{E}^T \boldsymbol{P}_\mathcal{E}, \tag{9}$$

$$K_\mathrm{A} = G^T B_\mathcal{E}^T C_\mathcal{E} B_\mathcal{E} G, \tag{10}$$

where we only need to use the elastic element subset $\mathcal{E}$ of element-integrated stress and stiffness. Notice that the force $\boldsymbol{f}_\mathrm{A}$ and stiffness K_A include the rigid degrees of freedom because G provides the kinematic mapping for velocities of vertices that are on the boundary between a rigid body and the elastic elements.

With careful bookkeeping, the Γ block in G need only deal with those rigid nodes that are on the boundary of an elastic element. In practice, we find it simpler to compute the product $B_\mathcal{E}\, G$ without removing those rigid vertices that are not on the boundary with an elastic element.

Before we assemble the mixed elastic and rigid equations of motion, observe that the mass term in the Rayleigh damping must be included as a damping force on the rigid degrees of freedom. The rigid body damping force is computed as $\Gamma^T M_{d\mathcal{R}} \Gamma\, \dot{\phi}$, and we note that this will damp both linear and rotational motion of the rigid body. The Rayleigh damping on both elastic and rigid degrees of freedom will therefore have a mass damping component $M_{d\mathrm{A}} = G^T M_d G$. Rigid motions are in the null space of the stiffness matrix, thus the stiffness term in Rayleigh damping does not contribute to damping of the rigid degrees of freedom. The active elastic degrees of freedom will be damped with the stiffness damping matrix $K_{d\mathrm{A}} = G^T K_d G$. This allows the application of stiffness damping specifically to the deformable elements.

Thus, the system we solve for semi-implicit backward Euler integration of the mixed elastic rigid system is

$$A_\mathrm{A} \Delta\dot{\boldsymbol{x}}_\mathrm{A} = h\left(D_\mathrm{A}\dot{\boldsymbol{x}}_\mathrm{A} + \boldsymbol{f}_\mathrm{A} + hK_\mathrm{A}\dot{\boldsymbol{x}}_\mathrm{A} + \begin{pmatrix} \boldsymbol{f}_{\mathcal{A}\mathrm{ext}} \\ c(\phi) + w_\mathrm{ext} \end{pmatrix}\right), \tag{11}$$

where $A_\mathrm{A} = M_\mathrm{A} - hD_\mathrm{A} - h^2 K_\mathrm{A}$, with M_A being block diagonal containing $M_\mathcal{A}$ and M_R. Just as before, we use LDLT factorization of A_A to solve $\Delta\dot{\boldsymbol{x}}_\mathrm{A}$. However, this system can be *much* smaller than the original fully elastic system and consequently much faster to solve. Furthermore, in the case of a system in elastic static equilibrium (whether the solid is stationary or moving rigidly), it all reduces to solving Equation 6 for the rigid body motion alone.

Once we have a solution for $\Delta\dot{\boldsymbol{x}}_\mathrm{A}$, we update the elastic and rigid position level variables with the updated velocities, where we use Rodrigues's formula [Murray et al. 1994] to turn angular velocities over a time step into an incremental rotation matrix R to update rigid body positions.

3.3 Contact Handling

We use signed distance computations for collision detection and generate a contact at every boundary vertex of one mesh that ends up inside another mesh (we do not process self-contact). The contact is defined by the interpenetrating vertex and the location of the closest point on the surface of the other mesh. Using the barycentric coordinates of the closest point, we create three rows in the contact Jacobian matrix J_c for each contact (one for the contact normal and two tangent directions).

We use projected Gauss-Seidel (PGS) to solve for contact impulses λ, and in turn, a velocity update due to contact forces to compute the velocity at the next time step. Let us consider the update for a fully elastic system:

$$\dot{\boldsymbol{x}}^+ = \dot{\boldsymbol{x}} + \Delta\dot{\boldsymbol{x}} + \underbrace{A^{-1} J_c^T \lambda}_{\Delta\dot{\boldsymbol{x}}_c}. \tag{12}$$

We multiply by J_c to compute the slip and separation of contacts at the next time step, and rearrange to form the system

$$\underbrace{J_c A^{-1} J_c^T}_{H}\,\lambda = -\underbrace{J_c(\dot{\boldsymbol{x}} + \Delta\dot{\boldsymbol{x}})}_{b}. \tag{13}$$

Solving this system with PGS, clamping the normal and tangential components of λ to their respective bounds at each iteration, provides a solution to the frictional contact problem:

$$\lambda_i^+ \leftarrow \lambda_i - (b + H_i\,\lambda)/H_{ii}, \tag{14}$$

$$\lambda_i^+ \leftarrow \max\left(\min\left(\lambda_i^+, \lambda_{i\mathrm{MAX}}\right), \lambda_{i\mathrm{MIN}}\right), \tag{15}$$

Bounds $\lambda_{i\mathrm{MIN}}$ and $\lambda_{i\mathrm{MAX}}$ are zero and positive infinity for normal direction constraints, while for tangent directions the bounds are set based on the current normal force and Coulomb coefficient of friction. While the PGS inner loop is fast, and depends only on the number of contacts, there is a cost to assembling H. We use the LDLT factorization of A to assemble H, and that cost is greatly reduced as larger portions of the elastic solid get mapped to rigid bodies. In a mixed elastic and rigid system, the velocity update $\Delta\dot{\boldsymbol{x}}_{\mathrm{A}c}$ for active degrees of freedom due to frictional contact uses the smaller system matrix A_A and a smaller Jacobian $J_{\mathrm{A}c} = J_c G$ relating contact

velocities with only the active degrees of freedom. As such, the contact solve greatly benefits from the smaller matrices that arise when large portions of the elastic solid are rigidified.

We use Baumgarte stabilization [Baumgarte 1972] to deal with interpenetration that arises on collision and at resting contacts. We do this by setting $b = J_c(\dot{\boldsymbol{x}} + \Delta\dot{\boldsymbol{x}}) + k_b\Phi$ in Equation 13 where k_b is the Baumgarte feedback coefficient and Φ contains the constraint violations (penetration at each contact). We include a small amount of compliance to maintain an invisible amount of interpenetration, which is helpful for warm startsing the PGS solve. We do this by modifying the Lagrange multiplier solve in Equation 14 to be $\lambda_i^+ \leftarrow (\lambda_i H_{ii} - b + J_{ci}\,\Delta\dot{\boldsymbol{x}}_c)/(H_{ii} + \gamma)$, for compliance γ. We typically set $\gamma = 10^{-3}$ and $k_b = 0.2/h$ for rapid convergence of interpenetration values (see also [Smith et al. 2005]).

We always warm-start the PGS solve with λ values of contacts that existed at the previous time step. This is important for more than PGS convergence because it plays an important role in the elastification process. When an elastic object is in static resting contact, we require the contact solve on the rigid system to produce the same forces as the contact solve with elastic elements. While many different contact solutions are feasible for a rigid body, most of these, if applied to the elastic system, would lead to a dynamic deformation and confuse our elastification oracle (see Section 3.5). At rest, the compliant contact will have normal interpenetration proportional to the normal contact force, $\lambda_i = \frac{k_b}{\gamma}\Phi_i$ for constraint i in the normal direction, regardless if solving for a fully elastic or fully rigid object.

Other contact solvers can also be used, for instance, penalty based methods would also respect our requirement of matching rigid and elastic contact forces. We also note that other constraint stabilization approaches can be used, for instance the post-step of Cline and Pai [2003], provided compliance is still included as a regularization to ensure that the interpenetration records the force distribution desired by the full elastic solve.

3.4 Rigidification

Identifying the portions of an elastic solid that can be simulated as a rigid body is relatively easy. If the rotationally invariant Green strain tensor $E = \frac{1}{2}(F^TF - I)$ remains constant for a period of time, then we allow the element to become rigid. In our method, we set a threshold τ_R based on the squared Frobenius norm of the strain rate, which is much like setting a speed limit on the slowest elastic deformation that we will simulate. The threshold is meaningful, easy to set, and works well for lively elastic systems, but may require low values for highly overdamped systems that only move slowly to static equilibrium. We flag elements as ready to become rigid if the norm is less than the set threshold for a given number of simulation steps (3 to 5 in our examples). Requiring the strain rate to stay below for a number of simulation steps prevents premature rigidification (for example, at the moment of maximum deformation in a vibrating cantilever).

While we could use $F = B\,\boldsymbol{x}$ and $\dot{F} = B\,\dot{\boldsymbol{x}}$ to compute

$$\dot{E} = \frac{1}{2}\left(\dot{F}^TF + F^T\dot{F}\right), \tag{16}$$

Algorithm 2: Identify a new rigid body by greedy connected component BFS. Called for all elements flagged as rigid.

Data: element e flagged rigid, rigid body index j to assign.
Result: number of elements in the new rigid component connected to e, or zero if no rigid body can be formed.

```
if e.visited then
    return 0                    // already in a rigid body
end
Q.enqueue(e)
c ← 0                           // initialize count
while |Q| > 0 do
    e ← Q.dequeue
    e.visited ← true
    if any vertex of e already assigned to a body then
        continue                // avoid hinges
    end
    e.rigidID ← j               // assign element to rigid body
    c ← c + 1
    for vertex v in element e do
        v.rigidID ← j           // assign vertex to rigid body
    end
    S ← unvisited & flagged-rigid face-neighbors of e
    Q.enqueue(S)
end
return c
```

this does not produce zero strain rate for rotational motion. The velocity used to step elastic node positions to a rotated position does not correspond to an instantaneous rigid rotational velocity, and $\dot{E}$ will contain non-zero eigenvalues indicating instantaneous compression. Instead, for time step k, we compute the finite difference

$$\dot{E}_k = \frac{E_k - E_{k-1}}{h}, \tag{17}$$

ignoring rotation by comparing strain in material space.

Each element of the mesh has a Boolean *rigid* property that we set when it is flagged for possible rigidification, that is, based on having $\|\dot{E}_k\|^2_{\text{fro}}$ below threshold for several time steps. Likewise, elements that are already rigid will also have the rigid property set (unless it is flagged for elastification, see Section 3.5). If the rigid properties are unchanged from the previous time step (and none have been identified for elastification), then the current set of rigid bodies is left unchanged. Otherwise, we must recompute the set of rigid bodies from connected components of elements flagged rigid.

Computing connected components and rigid bodies has linear time complexity. Each element and each vertex has a *rigidID* property to identify which rigid body it is part of (if any), and all are initially assigned -1 (i.e., none). We then use a breadth first search (BFS) to construct a connected rigid component for every element which is flagged for rigidification. We use an adjacency graph for tetrahedral elements with shared faces (or with shared edges for triangle elements in 2D simulations). While each connected component forms a rigid body in our simulation, we must be careful

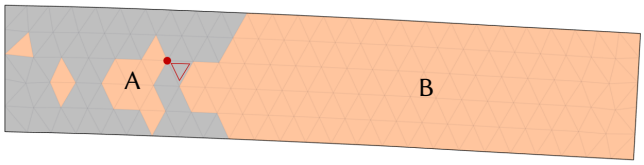


Fig. 2. Orange regions show rigid bodies. If the red triangle is a rigid candidate, it must not become part of body B as the red adjacent vertex already belongs to body A. Otherwise, bodies A and B would share the red vertex and require a hinge constraint for correct motion.

that different components do not share vertices. A shared vertex between two rigid components require a spherical joint constraint (similarly a shared edge would require a hinge constraint). Figure 2 shows an example of this in a 2D simulation, where a shared vertex would require a hinge constraint to keep the shared vertex at the same location. Thus, the BFS in Algorithm 2 includes a greedy vertex assignment and a vertex check to avoid hinge creation.

Once we have identified the connected components and the total number of rigid bodies, we then do a final linear pass over all vertices to compute the properties and state of each rigid body: center of mass p, linear and rotational mass matrix M_{R}, linear and angular velocity ϕ. The orientation R of the bodies are set to be the identity, and the linear and angular velocity are computed such that the linear and angular momentum about the center of mass, $M_{\mathrm{R}}\phi$, matches that of elastic degrees of freedom. If an object is moving rigidly, rigidification exactly preserves momentum. We lose momentum associated with non-rigid motion, but this is small due to our thresholds being conservative.

3.5 Elastification

The rigid parts of an elastic solid must be able to become elastic again when necessary. Tractions on the surfaces of the rigid regions may hold the elastic material within a rigid region in static equilibrium, or may not change enough (be large enough) to produce a noticeable deformation in the case of a very stiff elastic region. However, tractions can change, for instance, with the arrival of a traveling elastic wave, and this should cause rigid elements to become elastic again. Similarly, changing contact forces or new collisions should also cause elastification.

Ideally we would like to have an oracle that has low computational cost and can exactly identify only those elements that need to become elastic. However, the true solution requires a full solve to compute the strain rate of the full elastic system given the current state (position, velocity, contacts). Instead we propose a *quick solve* to inexpensively compute an approximate change in the velocities of all vertices, $\Delta\dot{\boldsymbol{x}}_{\mathrm{approx}}$, from which we can identify elements to make elastic *before* solving the system at a given time step. The key observation here is that the quick solve does not need to provide an accurate solution; it only needs to identify when rigid elements should become elastic.

We choose conjugate gradient for the quick solve. While the residual does not decrease monotonically, every iteration of conjugate gradient does reduce the error, and we can avoid the costly step of assembling the matrix A for the full elastic system. Preconditioning is essential, otherwise each iteration only propagates information between neighbours following the sparsity structure of A (for instance, an impulse at one vertex will only be able to influence the $\Delta\dot{\boldsymbol{x}}_{\mathrm{approx}}$ of that vertex and adjacent vertices with only one multiplication by A). Diagonal Jacobi conditioning, while meeting our requirement of low computational cost, does not alleviate this problem. In contrast, an incomplete Cholesky factorization is a good choice because the forward and backward substitution provides an excellent opportunity for an impulse at one vertex to influence $\Delta\dot{\boldsymbol{x}}_{\mathrm{approx}}$ at distant vertices, even with only one iteration of conjugate gradient (see Figure 8 in Section 4).

The system matrix in Equation 5 changes on each time step because the stiffness is dependent on the current state. Recomputing the preconditioner, even periodically [Courtecuisse et al. 2010], is costly. We neither want to incur the cost of the incomplete factorization on each quick solve, nor the cost to assemble the system matrix for just one multiplication. Instead we precompute the incomplete factorization of the constant Hessian approximation based on the mesh Laplacian as proposed by Liu et al. [2017]. With a suitable drop tolerance, we find that the incomplete factorization is both inexpensive, having a number of non-zeros similar to A, and performs very well in our quick solve (i.e., predicting elastification), even with only one iteration of conjugate gradient. Furthermore, we do not observe any noticeable difference in the predictive power when using the full Cholesky decomposition of the fixed preconditioner. In contrast, while similar in cost to using a drop tolerance, we do not see the same performance with no-fill incomplete Cholesky or no-fill modified incomplete Cholesky, regardless the permutation of variables (alternative minimum degree, reverse Cuthill-McKee, or nested dissection).

The quick solve in all our examples uses a drop tolerance of 10^{-6}, a nested dissection permutation, and a single iteration of conjugate gradient without assembly of A to solve the full system in Equation 5. The quick solve is done independently on each distinct elastic objects, irrespective of contacts. For gravity forces, We note that splitting is generally beneficial, especially in scenarios involving contact. That is, we update velocities based on gravity prior to the quick solve, rather than asking our single step preconditioned conjugate gradient to deal with these forces on the right hand side of the equation.

We have considered alternatives for the quick solve oracle. For instance, Gauss-Seidel iteration is effective for merging and splitting rigid bodies [Coevoet et al. 2020], and a careful ordering provides good propagation of information for correct treatment of impacts. But Gauss-Seidel is a poor choice for elastic material without additional mechanisms for long range information exchange [Kim et al. 2012]. While other alternatives may be an interesting avenue for future work, we believe we have found a good balance of simplicity and speed with our current solution.

3.5.1 Contacts. We take contact forces into account in the quick solve by using the contact forces from the previous time step. Recall that we use a warm start for the contact solve in Section 3.3. We include these warm start contact forces as explicit external forces in the quick solve by adding $J_c^T\lambda$ as a known quantity to the right hand side of Equation 5.

Table 1. Speedup for different examples, and the parameters used: rigidification and elastification thresholds τ_R and τ_E, time step h, Rayleigh parameters α_0 and α_1, Young's modulus E, Poisson's ratio ν, and number of tetrahedra $\#T$. All material energies in these specific scenes are neo-Hookean. Comparing the mean of per-frame speedup to the total speedup, we can see if the improvement in computation time is localized to specific frames or distributed over the full scene. For instance, the blob has big speedups at the start and end of the simulation, while the forest has a constant speedup. No-contact (NC) total and mean speedups show values computed without counting the contact solve time and demonstrate that our choice of contact solver does not exaggerate the benefit of adaptive rigidification. In all examples, we see a significant increase in performances, even in highly deforming scenes with non-localized deformations.

Sim	Time (s) Adaptive	Time (s) Default	Total Speedup	Mean Speedup	NC Total Speedup	NC Mean Speedup	τR	τE	h	Objects	α_0	α_1	ν	E	$\#T$
octopus	175.57	754.03	4.30	5.81	4.29	7.00	1e-5	1e-4	1e-2	octopus	1e-4	1e-1	0.30	5e3	6170
blob	1876.40	3915.30	2.09	9.10	1.81	5.74	1e-5	1e-4	1e-2	blob	1e-3	1e-2	0.22	5e3	6386
wheel	258.21	2966.80	11.49	16.43	9.74	12.39	5e-4	5e-3	1e-3	tire	1e-5	5e-2	0.30	3e3	15797
										rim	1e-5	2e-1	0.40	1e6	
forest	96.56	800.82	8.29	9.24	8.27	9.21	5e-3	5e-2	5e-3	pine tree	1e-5	1e-2	0.37	2e5	56818
										other tree	1e-5	7e-2	0.37	7e5	
										baseball bat	1e-4	1e-1	0.37	5e4	
										leaves	1e-4	1e-4	0.37	7e5	
pachinko	347.18	3803.20	10.96	13.24	5.23	5.63	7e-3	7e-2	1e-2	red pills	1e-5	5e-2	0.35	5e4	39808
										white pills	1e-5	1e-1	0.35	5e3	

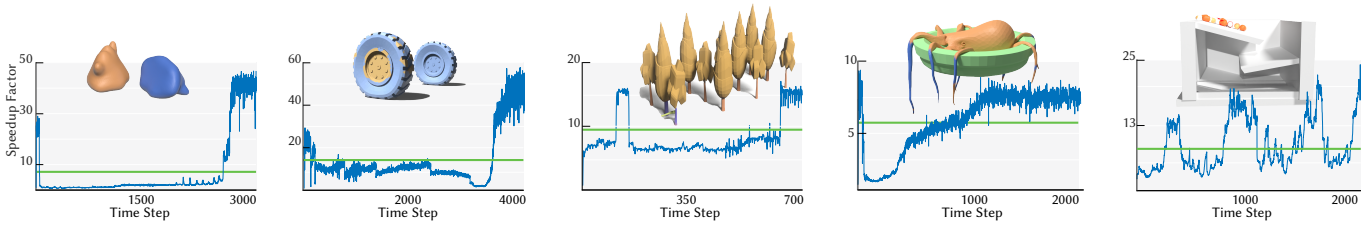


Fig. 3. Per-frame speedup factors for blob, wheel, forest, octopus, and pachinko. Using conservative elastification and rigidification thresholds, we can obtain very accurate simulations in reduced computation time. Adaptive rigidification works for collision, rotation, frictional contact, and proves a large benefit in big scenes with local deformations, such as forest, where it is possible to elastify individual trees as needed. The green line in each plot shows mean speedup.

New contacts pose a slightly harder problem. We handle new contacts by conservatively approximating the forces necessary to resolve these contacts. Just like warm started contacts, we include approximate contact forces for these new contacts on the right hand side of Equation 5. We treat new contacts as bilateral constraints, thus, with the new-contact Jacobian J_{cn}, we can write the KKT system

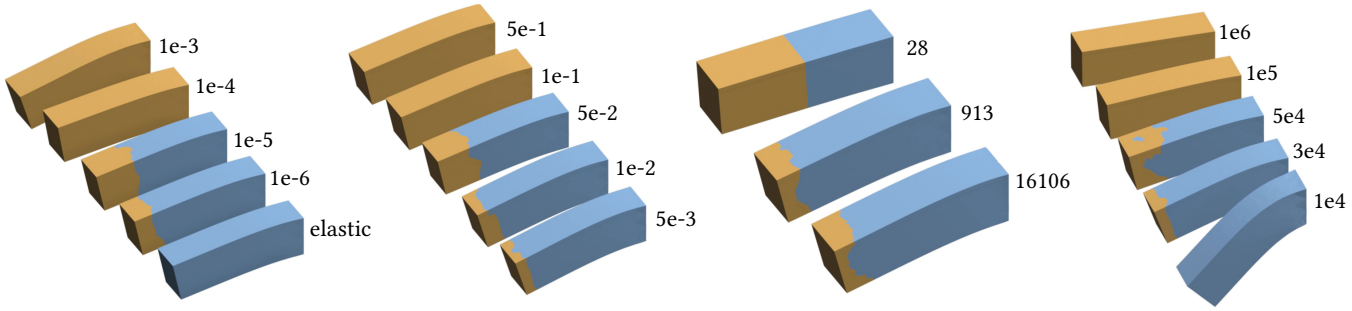
$$\begin{pmatrix} A & J_{cn}^T \\ J_{cn} & 0 \end{pmatrix} \begin{pmatrix} \Delta\dot{\boldsymbol{x}}_{\text{approx}} \\ \lambda_n \end{pmatrix} = \begin{pmatrix} z \\ -J_{cn}\dot{\boldsymbol{x}} \end{pmatrix}, \tag{18}$$

where z is the right hand size of Equation 5 along with explicit warm started contacts. Forming the Schur complement gives

$$J_{cn}A^{-1}J_{cn}^T\lambda_n = J_{cn}A^{-1}z + J_{cn}\dot{\boldsymbol{x}}, \tag{19}$$

which will be a small system for a small number of contacts. The complication here is A^{-1}, and recall that the main simulation loop will only compute the LDLT factorization of the elastic-rigid system matrix A_{A}, as opposed to the full elastic A. In the interest of having the fastest possible conservative solution, we assume the new vertices are isolated and uncoupled, and solve for their contact forces independently using an approximation of A^{-1} consisting of precomputed diagonal blocks (i.e., we disregard the implicit stiffness coupling), computed for the rest configuration. When many new contacts form simultaneously, for instance, a large contact patch forming on impact, we will overestimate the contact force due to missing coupling terms. But this will simply leads to a larger quick solve $\Delta\dot{\boldsymbol{x}}_{\text{approx}}$ solution, and in turn a larger region of elements will be conservatively converted to elastic for solving the next time step. In contrast, when new contacts form in isolation, for instance when a contact patch grows to include a new vertex, we will obtain an accurate estimate of the contact force.

Finally, with old warm-started and new approximate contact forces in account, a single iteration preconditioned conjugate gradient solve provides $\Delta\dot{\boldsymbol{x}}_{\text{approx}}$, which we combine with the current state $\boldsymbol{x}$ to compute $F = B\,x$ and $\dot{F} = B\,(\dot{x} + \Delta\dot{\boldsymbol{x}}_{\text{approx}})$, and in turn, the approximate strain rate $\dot{E}_{\text{approx}}$ using Equation 17. Every element that has $\|\dot{E}_{\text{approx}}\|_{\text{fro}}^2$ exceeding a elastification threshold τ_E is flagged for elastification. During the BFS described in Section 3.4, this can cause the outer layers of a rigid body to become elastic again, or can even break a rigid body into multiple components. It is the difficulty of tracking fragmentation of rigid bodies during elastification that motivates our strategy of recomputing the rigid bodies when there is a change. We briefly discuss incremental approaches in future work (see Section 5).



(a) Elastification threshold (b) Damping α_1 (c) Resolution vertices (d) Young's modulus

Fig. 4. (a) The rigidification threshold decides when rigid bodies are created, but it is the elastification threshold which is critical for deciding what regions will stay rigid. Lower values provide simulations that more faithfully reproduce the fully elastic behaviour, but at greater cost. (b) Rigidification takes place faster when there is higher damping in the simulation scenario, leading to greater speedups. (c) We observe very similar rigidification patters independent of the resolution of the mesh, while very coarse meshes rigidify more quickly because of resolution dependent stiffness and numerical damping. (d) Lower stiffness leads to larger and longer lived oscillations, while the damping in stiffer examples has them come to rest and rigidify earlier.

4 RESULTS

Table 1 shows parameters and performance measurements for all examples in the paper. Simulations were carried out on a Windows 10 PC using an Intel Core I7-6700K processor, with 64 GB of DDR3 RAM. Both the adaptive and non-adaptive simulations are primarily implemented in MATLAB, with performance critical components implemented in C++. We use GPToolbox [Jacobson et al. 2021] for some geometry processing and simulation specific functionality.

Our adaptive method is faster than its non-adaptive counterpart in all cases, both in terms of total simulation time and mean improvement in per-timestep computation time. The maximum performance improvement is approximately one order of magnitude in both cases (Table 1). Figure 3 shows detailed per-timestep speedups for many of our simulations which demonstrates that even in the worst-case, the adaptive code offers equivalent performance to the non-adaptive setup, and is often many times faster.

Our method is compatible with standard hyperelastic material models. We use Saint-Venant Kirchoff, neo-Hookean, and corotational energies in our simulator. The examples we show in this paper use StVK for 2D scenes and neo-Hookean for 3D scenes. While all of our examples use semi-implicit backward Euler integration (i.e., backward Euler on the linearized system), it is straightforward to use a full Newton solve with line search.

4.1 Threshold Selection

Selecting a threshold for rigidification is not difficult, and is largely a question choosing a trade-off between error and speed. However, a large threshold will lead to large errors; a good choice is crucial so as not to generate visual artifacts.

The square of the Frobenius norm provides an upper bound on the sum of squared eigenvalues of the strain rate. The eigenvalues correspond to stretch rates, which provides intuition. For example, a rigidification threshold of τ_R =1e-4, can be thought of as letting material become rigid if it is deforming at slower than 1% per second. This can be a good threshold for many scenes involving damped oscillations, and recall that making a portion of the mesh rigid does not prevent it from quickly become elastic again.

Figure 5 shows a clear relationship between between error and speed in the simulation of a 2D cantilever for different choices of τ_E with $\tau_R = 10^{-1}\tau_E$. The test scenario involves the under damped cantilever falling under gravity, visually coming to rest after a period of damped oscillation, and then reacting to a scripted force applied to its free end (see video). The error is computed as the magnitude of the maximum displacement of any vertex at any time from the fully elastic simulated trajectory, and is normalized by the length of the cantilever. We observe a maximum speedup with 10% max error above τ_E equal to 1, but this threshold setting is a poor choice because the cantilever simply remains rigid for the full simulation. In contrast, setting $\tau_E = 10^{-5}$ leads to a maximum error of approximately 0.1% in the simulation trajectory with a speedup of 1.7 times. This modest speedup comes from the fact that this is a simple 2D example with only 2739 triangular elements, while fine meshes in 3D lead to the more impressive speedups seen with other examples in this paper. A relationship between error and speedup is also observable in richer scenes. Figure 7 shows that the wheels using more conservative thresholds fall on the same side at approximately the same time while featuring good rigidification behavior.

While the rigidificaiton threshold needs to be high enough to let elastic elements become rigid, it is ultimately the elastification threshold that determines the behavior. If the elastification threshold τ_E is set too low then it will prevent the formation of any rigid bodies, while if it is set too high then the mesh can lock in a state far from a static equilibrium. This can be seen at top left of Figure 4 for the cantilever with τ_R =1e-3 which stops oscillating too early (see also the supplementary video).

Recall that the quick solve provides only an approximate prediction of the strain rate, and in our experience the solutions are always an overestimate. Intuition can come from observing the evolution of $\dot{E}$ and $\dot{E}_{\text{approx}}$ histograms during a simulation (see Figure 6). To account for the quick solve error, we typically set the elastification threshold to be one or two orders of magnitude higher than the rigidification threshold.

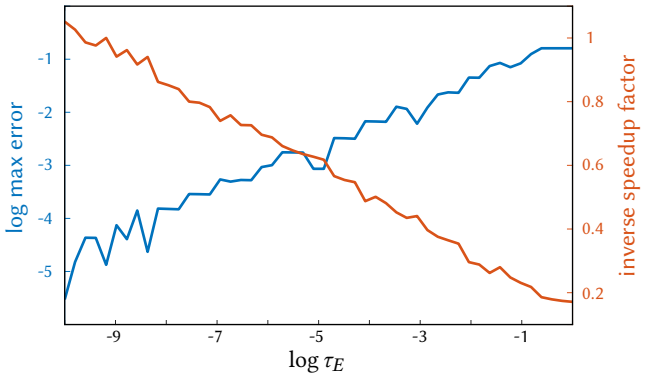


Fig. 5. Cantilever max vertex error (measured relative to cantilever length), and inverse speedup factor (lower is better), for a varying range of elastification thresholds and $\tau_R = 10^{-1}\tau_E$.

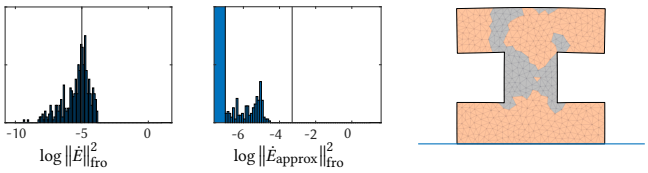


Fig. 6. Animated histograms (see video) of elastic element strain rate (left) and rigid element approximate strain rate from the quick solve (middle) provide intuition about simulation behavior for given thresholds.

Figure 4a shows effect of different elastification thresholds with $\tau_R = 10^{-2}\tau_E$. High thresholds give premature rigidification and a large error, while $\tau_E = 10^{-5}$ and lower match the fully elastic simulation. Our method accounts for varying material parameters because rigidification only depends on the strain rate (it is agnostic to the selection of parameters, as well as the choice of elastic energy and damping model). Increasing the damping (Figure 4b) or stiffness (Figure 4d), causes the simulated beam to rigidify more quickly. Finally, notice that similar rigidification regions form in meshes at different resolutions (Figure 4c), while an extremely low resolution mesh undergo rapid rigidification due to discretization-based numerical stiffening.

We observe that geometry can influence our threshold selection. For instance, geometry with long thin parts may benefit from a lower threshold to better capture global behavior (e.g., octopus, τ_R = 5e-7). Likewise, we may choose a lower threshold when small local details are important (e.g., forest, τ_R = 5e-5).

4.2 Example Simulations and Features

Our method supports local elastification and rigidification in response to external forces such as those arising from contact. Figure 8 shows two bowling balls dropped onto a mattress from different heights. Each impact elastifies a different local patch of the mattress mesh, with size proportional to the total force at impact. This conservative estimate of which elements need elastification is thanks to the approximation of a contact response in combination with the single iteration preconditioned conjugate gradients solve.

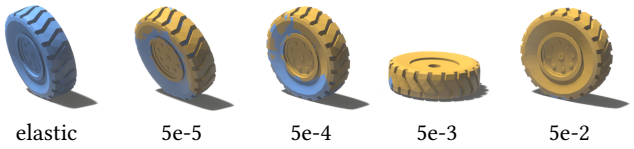


Fig. 7. Rolling wheels with different elastification thresholds τ_E fall at approximately the same time matching the fully elastic simulation. Lower thresholds lead to more accurate simulations. Here, adaptive simulations share a common rigidification threshold, $\tau_R = 10^{-1}\tau_E$.

Taken together, the expressivity of the elastification threshold parameter, along with the spatially varying behavior of the adaptive scheme allow us to find suitable settings which yield both a performance improvement and good visual agreement with non-adaptive simulations. For example, the elastic blobs (seen at left in Figure 3) are 2.6 times faster in terms of wall clock time, but visually indistinguishable from the standard finite element result.

This performance advantage persists even in more complicated scenes involving objects undergoing frictional contact. For instance, in the pachinko example in Figure 3, objects both become partially elastic and rigid as they navigate their way down multiple platforms. This ability for sliding and rotating objects to be rigid while in a deformed state is an important feature that improves runtime performance, in contrast to previous approaches which require the object to be in an undeformed state to be simulated as rigid [Chen et al. 2017].

Local elastification and rigidication also extend to more involved geometries and interactions. This can be seen in several of our examples, but the octopus scene provides a good example. As various parts of the octopus come to rest they rigidify while allowing other pieces to keep moving. Pulling on a tentacle causes only a small local portion of the mesh to become elastic, while the rest is simulated as a single rigid body.

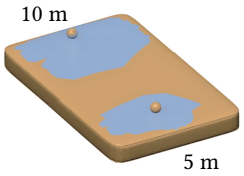


Fig. 8. The higher ball drop elastifies a larger region.

A key contribution of our method is that adaptivity is material-independent and so objects with heterogeneous material properties behave accordingly. Equivalent forces cause less strain on stiffer materials, which correctly leads to more aggressive rigidification. Our method is likewise compatible with arbitrary hyperelastic material models. This is a significant advantage over requiring a user to apriori intuit how an object will interact with a scene. Figure 1 demonstrates this with a rolling wheel that has a rubber tire and steel hub. The hub remains rigid for much of the simulation, simulated as a single rigid body, while the tire can elastify and rigidify on demand. The adaptive simulation retains excellent agreement with the non-adaptive simulation and is 5 times faster to simulate overall.

Finally, the forest example shows a particularly compelling use case of our method. In this scene, every tree is a finite element object and we move an axe through the scene, chopping at the trees. Our method correctly activates only parts of the trees required to capture the deformation, the rest are quickly simulated as rigid bodies. This

leads to a 10 times performance improvement over a standard FEM simulation. Such scenes, in which interaction in a large world is focused on a single hero character, are common in video games and movies.

All our examples feature objects which partially rigidify while in motion, and while deformed, and unfreeze (elastify) correctly in response to contact, avoiding common artifacts such as erroneous floating bodies. To our knowledge none of these examples are compatible with previous freezing approaches. Our method yields significant, often order of magnitude, speedups and is also compatible with other approaches to accelerate deformable object simulation. For instance, we could leverage Updated Sparse Cholesky Factors [Hecht et al. 2012] to reduce factorization costs. Our method is complementary to Subspace Condensation [Teng et al. 2015], which lists as its limitations a difficulty in handling scenarios where contacts can cause large changes in global motions, something our method excels at. We avoid the stated limitation of requiring careful construction of a reduced basis.

5 DISCUSSION AND LIMITATIONS

We currently recompute rigid body properties on each step only when there is a change in the elements making up the rigid bodies. However, if there is minimal change, such as just a small number of elements added or removed from a rigid body, there is an opportunity to do inexpensive incremental updates to the mass properties and rigid state. Currently the main challenge is when a rigid collection of elements splits apart into multiple rigid bodies, and designing efficient algorithms for this case is an interesting direction for future work. While the current linear algorithm to generate connected component is not a bottleneck, such incremental approach could further improve the efficiency of adaptive rigidification.

It would be interesting to consider how to make rigidification work for hexahedra with trilinear shape functions, or likewise any model with higher order shape functions. We currently monitor a single strain rate $\dot{E}$ for each element to identify if it should be rigid or not, but it could be possible to identify collections of vertices (control points) that could move rigidly based on monitoring their motion rather than the strain rate of quadrature points.

We currently only merge adjacent elements to form rigid bodies, while it could be interesting to also merge elements that are in contact. Our example simulations include cases where multiple elastic bodies stack and become rigid, and these examples could be further speed by following an approach similar to that of Coevoet et al. [2020], which merges rigid bodies and thus reduces the cost of collision detection and contact force computation.

6 CONCLUSIONS

We have presented a new adaptive method that uses on-the-fly rigidification to accelerate deformable object simulation. Our method is faster than standard, non-adaptive methods in all the examples presented and in many cases dramatically so (up to an order of magnitude wall clock time improvement). This is accomplished without sacrificing visual agreement with fully deformable methods. We believe our method to be a first and significant step in the grand unification of rigid body and deformable simulations. Up until now, whether to simulate rigidly or with deformation was a high-level modelling choice, made prior to ever running the simulation – a decision made for the purposes of improving performance. We imagine a world where users are free from this choice. Rather, algorithms will correctly adapt their chosen model based on the emergent physics of the simulated system. Our work shows that, not only is this possible, but can be of immense practical value. In order to spur future work in this important direction, code and data will be made available. We look forward to the more flexible future that awaits.

REFERENCES

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

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

J. Barbič and D. L. James. 2005. Real-Time Subspace Integration for St. Venant-Kirchhoff Deformable Models. *ACM Trans. Graph.* 24, 3 (July 2005), 982–990. https://doi.org/10.1145/1073204.1073300

J. Baumgarte. 1972. Stabilization of constraints and integrals of motion in dynamical systems. *Computer methods in applied mechanics and engineering* 1, 1 (1972), 1–16.

D. Chen, D. I. W. Levin, W. Matusik, and D. M. Kaufman. 2017. Dynamics-Aware Numerical Coarsening for Fabrication Design. *ACM Trans. Graph.* 36, 4, Article 84 (July 2017), 15 pages.

M. Cline and D. Pai. 2003. Post-stabilization for rigid body simulation with contact and constraints. In *2003 IEEE International Conference on Robotics and Automation*, Vol. 3. 3744–3751 vol.3. https://doi.org/10.1109/ROBOT.2003.1242171

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

H. Courtecuisse, J. Allard, C. Duriez, and S. Cotin. 2010. Asynchronous Preconditioners for Efficient Solving of Non-linear Deformations. In *Workshop in Virtual Reality Interactions and Physical Simulation "VRIPHYS" (2010)*. https://doi.org/10.2312/PE/vriphys/vriphys10/059-068

G. Debunne, M. Desbrun, M.-P. Cani, and A. H. Barr. 2001. Dynamic Real-time Deformations Using Space & Amp; Time Adaptive Sampling. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. ACM, New York, NY, USA, 31–36.

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

B. Gilles, G. Bousquet, F. Faure, and D. K. Pai. 2011. Frame-Based Elastic Models. *ACM Trans. Graph.* 30, 2, Article 15 (April 2011), 12 pages. https://doi.org/10.1145/1944846.1944855

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

F. Hecht, Y. J. Lee, J. R. Shewchuk, and J. F. O'Brien. 2012. Updated Sparse Cholesky Factors for Corotational Elastodynamics. *ACM Trans. Graph.* 31, 5, Article 123 (sep 2012), 13 pages. https://doi.org/10.1145/2231816.2231821

A. Jacobson et al. 2021. gptoolbox: Geometry Processing Toolbox. http://github.com/alecjacobson/gptoolbox.

J. Jansson and J. S. M. Vergeest. 2003. Combining Deformable- and Rigid-Body Mechanics Simulation. *Vis. Comput.* 19, 5 (aug 2003), 280–290. https://doi.org/10.1007/s00371-002-0187-6

L. Kharevych, P. Mullen, H. Owhadi, and M. Desbrun. 2009. Numerical Coarsening of Inhomogeneous Elastic Materials. *ACM Trans. Graph.* 28, 3, Article 51 (July 2009), 8 pages. https://doi.org/10.1145/1531326.1531357

T. Kim and D. L. James. 2009. Skipping Steps in Deformable Simulation with Online Model Reduction. *ACM Trans. Graph.* (2009).

T.-Y. Kim, N. Chentanez, and M. Müller-Fischer. 2012. Long Range Attachments - a Method to Simulate Inextensible Clothing in Computer Games. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '12)*. Eurographics Association, Goslar, DEU, 305–310.

J. Lenoir and S. Fonteneau. 2004. Mixing deformable and rigid-body mechanics simulation. In *Proceedings Computer Graphics International, 2004.* 327–334. https://doi.org/10.1109/CGI.2004.1309229

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

P.-L. Manteaux, F. Faure, S. Redon, and M.-P. Cani. 2013. Exploring the Use of Adaptively Restrained Particles for Graphics Simulations. In *VRIPHYS 2013 - 10th Workshop on Virtual Reality Interaction and Physical Simulation.* 17–24.

R. M. Murray, S. S. Sastry, and L. Zexiang. 1994. *A Mathematical Introduction to Robotic Manipulation* (1st ed.). CRC Press, Inc., USA.

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

M. Nesme, P. G. Kry, L. Jeřábková, and F. Faure. 2009. Preserving Topology and Elasticity for Embedded Deformable Models. *ACM Trans. Graph.* 28, 3, Article 52 (July 2009), 9 pages. https://doi.org/10.1145/1531326.1531358

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.

C. Schreck, D. Rohmer, S. Hahmann, M.-P. Cani, S. Jin, C. C. L. Wang, and J.-F. Bloch. 2016. Nonsmooth Developable Geometry for Interactively Animating Paper Crumpling. *ACM Trans. Graph.* 35, 1, Article 10 (Dec. 2016), 18 pages. https://doi.org/10.1145/2829948

R. Smith et al. 2005. Open dynamics engine user manual. (2005).

Y. Teng, M. Meyer, T. DeRose, and T. Kim. 2015. Subspace Condensation: Full Space Adaptivity for Subspace Deformations. *ACM Trans. Graph.* 34, 4, Article 76 (July 2015), 9 pages. https://doi.org/10.1145/2766904

D. Terzopoulos and A. Witkin. 1988. Physically based models with rigid and deformable components. *IEEE Computer Graphics and Applications* (1988).

M. Tournier, M. Nesme, F. Faure, and B. Gilles. 2014. Seamless adaptivity of elastic models. In *Proceedings of Graphics Interface 2014 (GI 2014)*. 17–24.

Y. Yang, G. Rong, L. Torres, and X. Guo. 2010. Real-time hybrid solid simulation: spectral unification of deformable and rigid materials. *Computer Animation and Virtual Worlds* 21, 3-4 (2010), 151–159. https://doi.org/10.1002/cav.373 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/cav.373