Affine Body Dynamics: Fast, Stable and Intersection-free Simulation of Stiff Materials

LEI LAN, Clemson University & University of Utah, USA
DANNY M. KAUFMAN, Adobe Research, USA
MINCHEN LI, University of California, Los Angeles & TimeStep Inc., USA
CHENFANFU JIANG, University of California, Los Angeles & TimeStep Inc., USA
YIN YANG, Clemson University, University of Utah & TimeStep Inc., USA

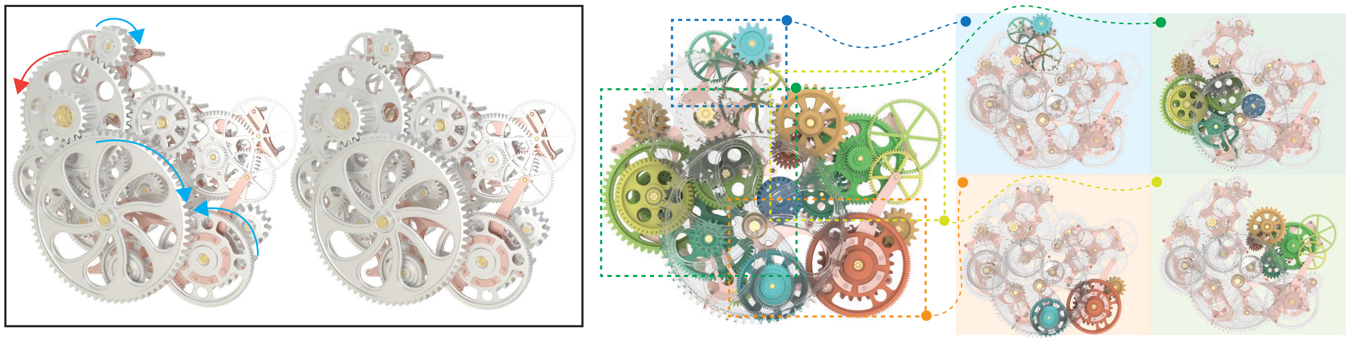

Fig. 1. **Geared system.** We propose an affine body dynamics (ABD) framework to efficiently and robustly simulate close-to-rigid contacting objects. ABD eases collision and contact processing costs over rigid body modeling while obtaining high-quality, intersection-free trajectories leveraging barrier-based frictional contact modeling. In this challenging stress-test benchmark we simulate a complex geared system composed of 28 toothed gears with frictional contact resolving all interactions. The combined gear set mesh is comprised of over 2.45M surface triangles. A torque applied to the actuated gear (red arrow) drives the motion of the entire system via contact, with over a quarter of all surface elements in active contact in every time step. Here we find that *all the existing rigid body simulation algorithms, including rigid-IPC, fail to make progress*, while ABD robustly simulates the example to completion. ABD enables large time step sizes (e.g., with $\Delta t = 1/50$ sec) even for such challenging, large displacement (rotation) contact processing. For a time step size of $1/100$ sec, ABD simulates each step in less than 12 sec on an `intel i9` CPU (multi-threaded), while the simulation runs interactively on a `3090` GPU (5-10 steps per second).

Simulating stiff materials in applications where deformations are either not significant or else can safely be ignored is a fundamental task across fields. Rigid body modeling has thus long remained a critical tool and is, by far, the most popular simulation strategy currently employed for modeling stiff solids. At the same time, rigid body methods continue to pose a number of well known challenges and trade-offs including intersections, instabilities, inaccuracies, and/or slow performances that grow with contact-problem complexity. In this paper we revisit the stiff body problem and present ABD, a simple and highly effective affine body dynamics framework, which significantly improves state-of-the-art for simulating stiff-body dynamics. We trace the challenges in rigid-body methods to the necessity of linearizing piecewise-rigid trajectories and subsequent constraints. ABD instead relaxes the unnecessary (and unrealistic) constraint that each body's motion be exactly rigid with a stiff orthogonality potential, while preserving the rigid body model's key feature of a small coordinate representation. In doing so ABD replaces piecewise *linearization* with piecewise *linear* trajectories. This, in turn, combines the best of both worlds: compact coordinates ensure small, sparse system solves, while piecewise-linear trajectories enable efficient and accurate constraint (contact and joint) evaluations. Beginning with this simple foundation, ABD preserves all guarantees of the underlying IPC model we build it upon, e.g., solution convergence, guaranteed non-intersection, and accurate frictional contact. Over a wide range and scale of simulation problems we demonstrate that ABD brings orders of magnitude performance gains (two- to three-orders on the CPU and an order more when utilizing the GPU, obtaining $10,000\times$ speedups) over prior IPC-based methods, while maintaining simulation quality and nonintersection of trajectories. At the same time ABD has comparable or faster timings when compared to state-of-the-art rigid body libraries optimized for performance without guarantees, and successfully and efficiently solves challenging simulation problems where both classes of prior rigid body simulation methods fail altogether.

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

Additional Key Words and Phrases: Rigid body dynamics, Reduced model, CCD, Barrier function

ACM Reference Format:
Lei Lan, Danny M. Kaufman, Minchen Li, Chenfanfu Jiang, and Yin Yang. 2022. Affine Body Dynamics: Fast, Stable and Intersection-free Simulation

Authors' addresses: Lei Lan, Clemson University & University of Utah, USA, lan6@clemson.edu; Danny M. Kaufman, Adobe Research, USA, dannykaufman@gmail.com; Minchen Li, University of California, Los Angeles & TimeStep Inc., USA, minchernl@gmail.com; Chenfanfu Jiang, University of California, Los Angeles & TimeStep Inc., USA, chenfanfu.jiang@gmail.com; Yin Yang, Clemson University, University of Utah & TimeStep Inc., USA, yin5@clemson.edu.

0730-0301/2022/7-ART67 $15.00
https://doi.org/10.1145/3528223.3530064

of Stiff Materials. *ACM Trans. Graph.* 41, 4, Article 67 (July 2022), 14 pages. https://doi.org/10.1145/3528223.3530064

1 INTRODUCTION

The simulation of highly stiff and so close-to-rigid materials remains a critical task in diverse applications ranging from animation and computer vision to robotics and geomechanics. A long-standing and natural strategy then is to model these bodies at the limit of stiffness and so treat them as exactly rigid. Equipped with just rotational and translational degrees of freedom (DOFs) this simplification enables the computational efficiency of rigid body methods as they utilize orders of magnitude fewer DOFs when deformations can be excluded.

At the same time this minimal representation for rigid bodies poses several fundamental challenges of its own in exchange for small system size. The first being that moving from a flat continuum model to rigid (SE(3)) coordinates introduces significant nonlinearities that must be resolved. The second being that infinite stiffness implies applied forces, and especially contact responses, are communicated instantaneously across the material domain. In combination these two issues have long challenged both rigid body models (e.g., Painlevé's paradox) and the downstream simulation methods derived from them. State-of-the-art rigid body methods have long focused on efficient velocity (twist) level solutions. However, in doing so, these methods tend to generate undesired positional errors in the form of intersections and instabilities.

To address these issues a *position-level* barrier method for rigid bodies was recently introduced by Ferguson and colleagues [2021]. Applying the incremental potential contact (IPC) model [Li et al. 2020] to the rigid body model, the resulting rigid-IPC method provides robust intersection-free simulation of rigid solids with frictional contact. However, here a third challenge posed by rigid body models limits the efficiency and applicability of rigid-IPC: accurately tracing a piecewise-rigid trajectory is much more difficult than for a piecewise-linear trajectory. This is required across numerous continuous-collision detection (CCD) operations throughout computation in order to ensure non-intersection. Rigid-IPC addresses this third challenge by conservatively subdividing rigid transformations into piecewise-linear subsequences to solve the curved CCD. While viable, the expense for this curved CCD remains substantial – especially as CCD is invoked heavily throughout each simulation step. As a result, the overall performance of rigid-IPC is close to (and occasionally slower than) comparable (appropriately stiffened) full-space finite element method (FEM) IPC simulations [Li et al. 2020]. Here the curved CCD in rigid-IPC severely undermines the advantage of its small DOF representation.

Beginning with this analysis, our takeaway is that the key advantage of rigid body models is *not* the rigidity assumption but rather the compact representation. Indeed, the rigidity assumption is neither necessary, efficient, nor particularly accurate since no material is perfectly rigid. Following this reasoning, we approach the stiff-body problem from a different perspective: we remove the constraint (and resultant limitations) that the motion be exactly rigid. Specifically, we construct an affine-body dynamics (ABD) model, directly stiffened to obtain close-to-rigid trajectories and augmented with an IPC-type barrier. ABD preserves all guarantees of the IPC model including solution convergence, guaranteed non-intersection, and accurate frictional contact. However, discrete steps in ABD are now, as in the FEM case, piecewise linear, enabling us to utilize efficient linear CCD routines. Likewise ABD remains compact, utilizing 12 DOFs per body – a bit more than rigid bodies but still compact enough for efficient system solves. In turn the relaxation from the rigidity constraint allows ABD to significantly outperform the rigid-IPC method across all benchmarks (ranging from two- to three-order speedups for side-by-side comparison on the CPU, and an order-of-magnitude further improvement enabled by our GPU implementation), and to likewise successfully simulate problems where rigid-IPC fails.

At the same time, by combining compact representation and efficient collision processing, ABD also exhibits clear advantages in quality, reliability and even performance when compared to off-the-shelf rigid body simulation libraries. Here these libraries (we use `Bullet` [Coumans 2015] as our baseline for comparison; see Section 5 for a discussion of this choice) are often optimized for speed over robustness and guarantees. Nevertheless, without requiring the precomputation of collision proxies (e.g., the convex decompositions required by `Bullet`, `Mujoco` [Todorov et al. 2012], and `PhysX` [Nvidia 2011]), ABD remains closely competitive in performance on small-scale examples, while obtaining significantly faster simulations on larger and/or more challenging scenarios.

ABD is also able to simulate a wide range of challenging modeling problems where existing rigid body methods and libraries fail altogether. As an example, in Fig. 1 we demonstrate a driven mechanical system with all gear-to-gear interactions processed directly via frictional contact. As we apply an external torque to the driving gear, the entire mechanism moves with well over a quarter of the mesh's 2.45M triangles actively in contact during each time step. Here we find rigid-IPC (curved CCD failures) and `Bullet` (severe intersections) are both unable to simulate this mechanism even as we adjust algorithm settings and time step sizes conservatively. ABD simulates the scene robustly without algorithm tuning. Under a time step of $\Delta t = 1/100$ sec, ABD simulates each step/frame in 12 sec on the CPU (multi-threaded) and reaches an interactive speed on the GPU ranging from 5 to 10 FPS.

In a nutshell, ABD provides a new stiff-body simulation framework suitable for all rigid-body-type modeling problems that offers similar (or improved) performance when compared to existing rigid body libraries (optimized for performance), while providing guarantees of non-intersection, accurate contact, and convergent implicit solves that they do not. ABD does not require pre-computed convex part proxies for simulation geometries; frees users from time-consuming per-scene parameter sweeps to find parameters that work; and ensures successful simulation completion in challenging cases where prior methods fail altogether.

2 RELATED WORK

Forces and impulses are propagated rapidly across highly stiff materials, largely eliminating relative deformations. Rigid body models are then an idealization simplifying stiff objects. Here extreme stiffness is handled *kinematically* by directly formulating rigid body models with SE(3) coordinates. Rigid body models have been extensively studied by the graphics community dating back to the pioneering work of Baraff [1989]. We refer the reader to the comprehensive survey from Bender an colleagues [2014], which covers the wide spectrum of rigid body simulation methods.

The primary focus of rigid body research is typically the resolution of frictional contact. Intuitively, objects should never intersect with each other during the simulation. Enforcing this requirement often leads to algorithms based on linear complementarity programming (LCP) [Baraff 1994, 1995; Stewart 2000; Trinkle et al. 2001]. One needs to carefully search within the combinatorial space for an approximation of a feasible configuration, while the interaction among bodies in contact is generally based on impulses rather than forces [Baraff 1989]. The LCP-based contact problem is known to be NP-Hard due to the indeterminacy of which contacting nodes are contributing the collision impulse [Baraff 1991]. Alternate solvers formulated on approximations of velocity-level LCPs have been popular [Anitescu and Potra 1997; Erleben 2007; Kaufman et al. 2005]. Yet, the resulting system remains nonconvex and challenging to solve for complex scenes. Irrespective of the accuracy of the solution, here the required constraint linearization means that intersections leading downstream to artifacts like drifting and tunneling can and will result. To reduce these artifacts, additional constraint stabilizations are often employed [Baumgarte 1972; Cline and Pai 2003; Moreau 1988] however this, in turn, can introduce new instability artifacts like popping and explosions. In addition to rigid bodies, LCP models are also widely used for contact modeling among non-rigid objects [Duriez et al. 2005; Hauser et al. 2003; Otaduy et al. 2007; Pauly et al. 2004; Song and Kumar 2003].

For small-scale problems, a direct LCP solver could be used. When the complexity and the dimensionality increase, iterative LCP solvers stand as a more efficient option. Here successful designs of various iterative methods for LCP-based contacts such as Gauss-Seidel [Erleben 2007], PROX (iterative proximal operator) [Erleben 2017], surrogate constraints [Kaufman et al. 2005], accelerated gradient descent [Mazhar et al. 2015], staggered projections [Kaufman et al. 2008], and adaptive merging [Coevoet et al. 2020] have all been applied.

Penalties are also another popular option widely used to process collisions [Cundall and Strack 1979; Terzopoulos et al. 1987; Teschner et al. 2005]. Instead of imposing inequality constraints, a penalty method often chooses a spring-like repulsion mechanism based on the penetration depth between two objects [Drumwright 2007; Fisher and Lin 2001; Hasegawa et al. 2004]. While computationally simple, the penalty method fails for fast-moving models or simulations under large time steps and often requires significant manual tuning of stiffness parameters per scene. Its stability can be significantly enhanced using implicit formulations coupled with CCD [Tang et al. 2012; Xu et al. 2014]. Nevertheless, interpenetration still can and will result. This defect limits its wider use beyond graphics, where visual plausibility is not the only concern. Recently, Müller and colleagues also proposed a position-based rigid body framework [Müller et al. 2020]. Unlike classic rigid body algorithms, this method uses PBD-like constraint projection [Macklin et al. 2016; Müller et al. 2007] to process multiple-body dynamics.

Collision detection is another important procedure for modeling rigid bodies with contacts. In general, a collision can occur between any triangle pair of two objects, and an exhaustive triangle-based collision detection is infeasible for high-resolution models. To this end, a commonly adopted method is to use some bounding volume hierarchy (BVH) [Zachmann and Langetepe 2003] to avoid excessive triangle-triangle intersection tests. This pre-screening procedure is known as collision culling. Different BV types have been explored including AABB [Bergen 1997], OBB [Gottschalk et al. 1996], bounding sphere [Hubbard 1995; James and Pai 2004], Boxtree [Zachmann 2002], spherical shell [Krishnan et al. 1998]. As the geometry of the model does not change in rigid body models, BVH updates become particularly convenient – the per-body rigid transformation can be directly applied to update the BVH instead of re-building it from scratch (as opposed to deformable objects). In some existing rigid body packages e.g., `Bullet` library [Coumans 2015], collision detection does not apply directly to surface meshes but rather to a volumetric proxy of the model, formed of convex components – most often obtained by via a convex decomposition. While methods like `Bullet` require these convex proxies for robust processing this proxy also helps as an acceleration for collision detection. In this paper, we provide a new culling method to accelerate collision detection for rigid bodies and ABD. Our method leverages the fact that the colliding region between two rigid bodies often constitutes a very small fraction of their surfaces. Based on this observation, we create a BVH only covering the overlapping region of two bodies for a more effective culling.

Discrete collision detection (DCD) checks for collisions or penetrations at a specific time instance. This method could miss interpenetrations if the detection is not performed frequently enough. Alternatively, CCD checks the possible overlap of the trajectories of the surface primitives and returns the first time of impact (TOI) [Bridson et al. 2002; Redon et al. 2005]. The overlap test for triangle-vertex and edge-edge becomes a cubic polynomial, and several root-finding algorithms are available for solving the TOI [Brochu et al. 2012; Redon et al. 2005; Tang et al. 2014]. In a recent contribution from Wang and colleagues [2021], a more stable root-finding algorithm was proposed based on an improved inclusion.

Lastly, the most closely related work to ABD is on rigid-IPC from Ferguson and colleagues [2021]. Rigid-IPC features a new rigid body formulation, where contact is modeled with a barrier-based potential i.e., the IPC [Li 2020; Li et al. 2020] model. Conceptually, IPC is similar to the implicit penalty method (e.g., as in [Tang et al. 2012]), which produces a repulsion force pushing apart two contacting objects. However, due to the dedicated design of the barrier function, IPC provides guaranteed intersection-free collision resolution when appropriately combined with a CCD-filtered line search. When compared to existing contact handling methods, IPC has demonstrated a superior performance – it is significantly faster than LCP-based solutions for complicated contacts where LCP-based methods often fail altogether and much more robust than regular penalty methods with user-specified accuracy bounds. This method has been broadly applied to elastodynamics simulation [Li et al. 2020], codimensional models [Li et al. 2021b], embedded interfaces [Zhao et al. 2022], FEM-MPM coupling [Li et al. 2021a], deformation processing [Fang et al. 2021], and reduced models [Lan et al. 2021]. Rigid-IPC extends rigid body modeling to IPC and so provides significantly improved reliability in contact processing. Unfortunately, this also comes at a cost – strictly rigid motion imposes significant computational challenges as the trajectory for CCD evaluation in rigid-IPC is curved. In order to successfully compute TOI in curved CCD, Ferguson and colleagues [2021] subdivide the rigid trajectory into piece-wise line segments, which becomes the new bottleneck of the simulation. We observe that IPC already demonstrates the advantages of smooth (albeit stiff) approximation in-place of hard constraints. Here we then consider another smooth approximation, this time for the rigidity constraint in rigid body modeling. ABD is then constructed following this intuition with a substantial performance gain. ABD provides stiff compliance (as in real-world stiff materials) enabling tight, almost rigid parts to robustly conform together in contact – even in cases where find that a strictly rigid modeling approach can fail. Finally, as in rigid-IPC, ABD provides user-controllable physical and geometric accuracies with solutions that reliably reach specified tolerances.

3 ABD KINEMATICS

We begin by constructing a flat kinematics via affine coordinates. We equip each simulated body b in our domain with a time-varying linear transform $\mathsf{A}_b(t) \in \mathbb{R}^{3\times3}$, and a translation $p_b(t) \in \mathbb{R}^3$. In the following we often store per-body configuration in vector form as: $q = (p^T, a_1^T, a_2^T, a_3^T)^T \in \mathbb{R}^{12}$, with the transform $\mathsf{A} = [a_1, a_2, a_3]^T$, then stored in row order.

Each material point k in body b has a body frame (equivalently rest) position $\bar{x}_k$ with its corresponding world frame coordinates given by the affine map:

$$x_k = \mathsf{A}_b \bar{x}_k + p_b = \mathsf{J}(\bar{x}_k) q, \tag{1}$$

and its velocity,

$$\dot{x}_k = \dot{\mathsf{A}}_b \bar{x}_k + \dot{p}_b = \mathsf{J}(\bar{x}_k) \dot{q}. \tag{2}$$

Here note that $\mathsf{J}(\bar{x}) = [\mathsf{I}_3, \mathsf{I}_3 \otimes \bar{x}]$ is *constant* across all configuration changes, where I_3 is a 3 by 3 identity matrix.

3.1 Kinetic Energy

Given a mass density distribution, ρ, over the body domain, Ω, the kinetic energy of each affine body is then

$$\begin{aligned} \frac{1}{2}\int_\Omega \rho \dot{x}^T \dot{x} \, \mathrm{d}\Omega &= \frac{1}{2}\int_\Omega \rho (\dot{\mathsf{A}}\bar{x} + \dot{p})^T (\dot{\mathsf{A}}\bar{x} + \dot{p}) \, \mathrm{d}\Omega \\ &= \frac{1}{2}\dot{q}^T \left(\int_\Omega \rho \, \mathsf{J}(\bar{x})^T \mathsf{J}(\bar{x}) \, \mathrm{d}\Omega \right) \dot{q}, \end{aligned} \tag{3}$$

with the generalized mass matrix for each affine body defined as:

$$\mathsf{M} = \int_\Omega \rho \, \mathsf{J}(\bar{x})^T \mathsf{J}(\bar{x}) \, \mathrm{d}\Omega. \tag{4}$$

Here we note an additional convenience of ABD: for flat affine coordinates we obtain a constant mass matrix. In turn this ensures that the equations of motion for affine bodies, unlike rigid bodies, *do not* add nonlinear Coriolis-type forces.

The equations of motion for ABD are then simply

$$\mathsf{M}\ddot{q} = -\nabla V(q) + f, \tag{5}$$

where V is the total potential energy, and external forces $f_k \in \mathbb{R}^3$, applied at material points k, are included as $f = \sum_k \mathsf{J}(\bar{x}_k)^T f_k$.

3.2 Orthogonality Potential

In place of SE(3) coordinates we rigidify each affine body with a stiff orthogonality potential

$$V_\perp(q) = \kappa v \|\mathsf{A}\mathsf{A}^T - \mathsf{I}_3\|_F^2, \tag{6}$$

scaled by the stiffness κ and the body's volume v. We apply a large stiffness ($\kappa > 100$GPa) to ensure the deformation on the body is sufficiently suppressed and negligible.

As we relax the rigidity constraint, we instead apply a highly stiff penalty term. Generally engineering rule of thumb suggests that stiff penalties should be avoided in simulation, as they tend to exacerbate the numerical difficulties of computing with nonlinear potentials. However, we observe that contact and collision forces (irrespective of whether they are treated via constraints, penalties springs, or barriers) introduce a much more dominant stiffness to the system. In our case this already requires handling system solves with a robust Newton-type algorithm so that the overhead of a single, additional stiff orthogonality potential per body is negligible. In practice, as we show in Section 5, directly stiffened affine systems significantly improve in performance over comparable systems that resolve rigid motion explicitly.

The stiff potential itself provides the effective constitutive model for the affine bodies – modulating their collision response upon impact (and so intrinsically handling restitution). While $V_\perp$ is a natural energy choice for close-to-rigid motion [Moser and Veselov 1991], many other alternatives are certainly reasonable. Indeed affine bodies could alternately be equipped with the ARAP energy [Alexa et al. 2000; Igarashi et al. 2005; Sorkine and Alexa 2007], i.e., $\|\mathsf{A} - \mathsf{R}(\mathsf{A})\|_F^2$ with polar decomposition $\mathsf{A} = \mathsf{RS}$, to enforce rigidity, or else neo-Hookean, or any number of other rotation-invariant hyperelastic energies [Bonet and Wood 1997]. However, we find that the orthogonality potential is both effective and efficient. Unlike ARAP, $V_\perp$ does not require an expensive decomposition – it can be computed as a polynomial

$$V_\perp = \kappa v \left(\sum (a_i \cdot a_i - 1)^2 + \sum_{i \neq j} (a_i \cdot a_j)^2 \right), \tag{7}$$

leading to more efficient evaluations of energy gradient and Hessian:

$$\begin{aligned} \frac{\partial V_\perp}{\partial a_i} &= 2\kappa v \left(2(a_i \cdot a_i - 1) a_i + 2 \sum (a_j \otimes a_j) a_i \right), \\ \frac{\partial^2 V_\perp}{\partial a_i^2} &= 2\kappa v \left(4 a_i \otimes a_i + 2(\|a_i\|^2 - 1)\mathsf{I}_3 + 2 \sum a_j \otimes a_j \right). \end{aligned} \tag{8}$$

In comparison, energy operations for affine bodies with $V_\perp$ are over 43% and 178% faster than applying ARAP and neo-Hookean models.

4 AFFINE IPC

We simulate systems of affine bodies with triangulated boundaries. For each affine body $b \in \mathcal{B}$, we construct a discrete incremental potential (IP) [Li et al. 2020], E_b, whose stationary points give the unconstrained time step update:

$$q_b^{t+1} = \arg\min_{q_b} E_b(q_b), \quad E_b = \frac{1}{2}\|q_b - \widetilde{q}_b\|_{\mathsf{M}}^2 + \Delta t^2 V_\perp(q_b), \tag{9}$$

Here Δt is the time step size, $\widetilde{q}_b = q_b^t + \Delta t \dot{q}_b^t + \Delta t^2 \mathsf{M}^{-1} f_b^{t+1}$ a known vector composed from the prior step's state, and f_b are per-body external forces (see Eq. (5)). Unlike rigid body dynamics[1], ABD's flat equations of motion allow us to directly construct IPs for a broad range of standard implicit time-integration methods [Li et al. 2020, 2021b].

Overloading $q = \left(q_1^T, \cdots, q_{|\mathcal{B}|}^T\right)^T$ as the stacked vector of all affine body DOF, we construct IPC potentials for contact, $V_C(q)$, and dissipative friction, $V_F(q)$, to model inter-body contact forces. The contact potential

$$V_C(q) = \kappa_c \sum_{i \in \mathcal{C}} B\left(d_i(q)\right), \tag{10}$$

(with contact stiffness κ_c) resolves contacts between all pairings $i \in \mathcal{C}$ of inter-body surface geometry primitives (e.g., edge-edge and vertex-face pairs between body meshes) while leaving out intra-body surface pairs as there is no self-contact (high-stiffness). The smoothly clamped logarithmic barrier function,

$$B(d, \hat{d}) = \begin{cases} -(d - \hat{d})^2 \ln\left(\dfrac{d}{\hat{d}}\right), & 0 < d < \hat{d} \\ 0 & d \geq \hat{d} \end{cases}, \tag{11}$$

evaluates unsigned distances, d for all primitive pairs in $\mathcal{C}$. Smooth clamping ensures that proximities beyond $\hat{d}$ can be safely culled from evaluation without harming convergence, while surface pairs within the small, prescribed contact accuracy, $\hat{d}$, generate contact forces.

The friction potential is then

$$V_F(q) = \sum_{j \in \mathcal{F}} \mu \lambda_j m_j(q), \tag{12}$$

where $\mathcal{F} \subseteq \mathcal{C}$ is the active subset of contact pairs with positive contact force magnitude $\lambda_j = -\kappa_c \nabla_q B\left(d_j(q)\right)$, μ is the coefficient of friction, and m_j returns a mollified norm of the relative sliding velocity, orthogonal to the distance vector, between geometric pairs j. See Li et al. [2020] for details and analysis demonstrating that V_F provides smooth approximation of the nonsmooth Coulomb friction model with controllable accuracy. In turn, this friction model effectively and accurately captures frictional stick and slip behaviors for both maximal and reduced body models [Ferguson et al. 2021; Li et al. 2020]. At each time step we then construct a global IP for the full contact-coupled system as $E(q) = V_C(q) + V_F(q) + \sum_{b \in \mathcal{B}} E_b(q_b)$. We then minimize, $q^{t+1} = \text{argmin}\, E(q)$, with line-search filtered Newton and update system configuration.

Restitution models are applied to impose collision-induced energy dissipation for physical systems that do not include internal forces. As ABD has both a constitutive model and local contact deformation (via contact barriers) its restitution behavior is implicitly controlled jointly by κ_c, $\hat{d}$, and Δt, as well as the time integration method applied. We can estimate how much work/energy the barrier-penalty will store and release within the $\hat{d}$ region to compute collision and time step parameters that target a desired restitution.

[1]Rigid body models require Poisson or constrained Lagrangian methods for numerical time integration [Hairer and Vilmart 2006].

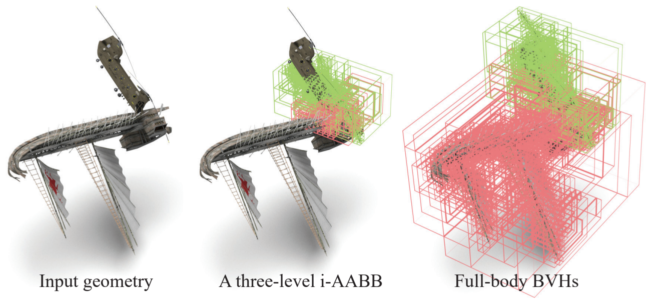

Fig. 2. **i-AABB.** Intersection-AABB (i-AABB) is a simple and more effective culling strategy than per-object BVHs often employed for rigid bodies. In this example, we collide helicopter and ship models with well over 343K surface triangles for the system. Culling with full-body BVHs leads to 1.3M AABB intersection tests. Instead, we build a shallow, three-level i-AABB based on the overlapping AABB volume, and the total number of intersection tests reduces to 120K, 90% fewer than regular BVH-based culling.

4.1 Affine CCD

To ensure that every Newton iterate maintains nonintersection, a significant (and for rigid bodies dominant) cost of each timestep's solve is the repeated CCD evaluation of surface primitive pairs in $\mathcal{C}$ during line-search filtering. Edge-edge and vertex-face pairs are defined on vertices $x_j \in \mathbb{R}^3$, $j = 1$ to 4, each belonging to one of two bodies in $\mathcal{B}$. At every iteration ℓ of each Newton solve new tentative positions for these vertices, $\mathsf{A}_b^\ell \bar{x}_j + p_b^\ell$, are proposed per vertex j and participating body b. We then perform CCD search along the directions formed by $\Delta\mathsf{A}_b^\ell = \mathsf{A}_b^\ell - \mathsf{A}_b^{\ell-1}$ and $\Delta p_b^\ell = p_b^\ell - p_b^{\ell-1}$. Equivalently, the corresponding trajectories to apply CCD to are then just $(\mathsf{A}_b^{\ell-1} + \alpha\Delta\mathsf{A}_b^\ell)\bar{x}_j + p_b^\ell + \alpha\Delta p_b^{\ell-1}$, with $\alpha \in [0, 1]$. In other words, we can simply test displacements (from 0 to 1) along $\Delta\mathsf{A}_b^\ell \bar{x}_j + \Delta p_b^\ell$ starting from $x_j^{\ell-1} = \mathsf{A}_b^{\ell-1}\bar{x}_j + p_b^{\ell-1}$, i.e., the previous iteration's position, and so can apply standard linear CCD rather than the expensive, curved CCD required for rigid-body trajectories. Throughout we employ the Additive CCD (ACCD) method [Li et al. 2021b] for all affine body CCD evaluations.

4.2 Contact Culling via i-AABB

The IPC framework inverts current contact-processing practices by integrating collision detection within every iterate, inside each nonlinear time step solve Above we have already seen the implications of this choice on CCD and the advantages for ABD. We next address culling of contact pair evaluations and then below, in the following section, an integrated approach to efficiently compute the local energy Hessian evaluations and their assembly.

To further cull inter-body surface pairs from downstream collision processing, rigid body methods commonly employ precomputed per-body (or per unique mesh instance) BVH which only requires a rigid (or, in our setting, affine) transform to evaluate at each configuration update. To further improve collision detection queries, most popular rigid body libraries (e.g., `Bullet`, `Mujoco`, `PhysX`) additionally require that all nonconvex surface meshes be replaced by approximations – convex decompositions [Mamou et al. 2016] so that evaluations can utilize convexity assumptions.

In the context of stiff-body models where self-collisions can be safely ignored, we propose a simple strategy that significantly enhances contact-pair culling. We start with the observation that contact-dense regions between close-to-rigid meshes are most often local. Unlike deforming meshes, we see that a large portion of each body's surface remains free of contacts. A BVH evaluation over each body's full surface is then a bit too aggressive (and a potentially unnecessarily expensive precomputation cost) – especially as we consider higher-resolution models and/or increasing numbers of bodies.

We instead begin with a coarse AABB per body, offset to account for the barrier's small $\hat{d}$ parameter. While clearly not providing a tight bound we only need to evaluate potential contacts in each pairwise overlap between these initial, *intersecting* AABB (i-AABB) volumes. Before the simulation, we organize these AABBs into a binary tree $\mathcal{T}$. Each leaf node of $\mathcal{T}$ hosts one body's AABB, and the geometry of the AABB is updated directly by applying the current affine transformation. The construction of $\mathcal{T}$ is in a bottom-up fashion. Two nearby bodies are merged into an upper-level tree node. Here we use a 30-bit Morton code [Morton 1966] to determine which two bodies are grouped as siblings. During the simulation, the i-AABB culling builds the list of overlapping pairs. Specifically, we would like to know if AABBs of two bodies intersect given their respective affine coordinates. This is done by performing a level-order traversal of $\mathcal{T}$, and it is parallelized at all the bodies. In our implementation, we require that the index of the first body is always bigger than the second one in an overlapping pair to avoid redundant intersection tests. With stiff affine transformations the overlap volume of each i-AABB is most often sufficiently small – the total number of surface primitives within our i-AABB is often one order smaller than that obtained from full-mesh BVH queries. Then, utilizing the guarantee that i-AABBs will not intersect one another, this already provides effective culling for us to directly proceed with parallel (multi-threaded) primitive-pair collision-check evaluations within each i-AABB.

In the most extreme, heavily entangled configurations, e.g., see Fig. 2, we find i-AABB volumes can be too conservative for CPU implementations. Here we simply build a shallow, three-level AABB hierarchy (a binary AABB tree) for each i-AABB. Our observation is that for GPU implementations and most (even contact-intense) CPU examples the single-level i-AABB is highly effective (and exceedingly simple to implement). We utilize the i-AABB hierarchies solely for CPU examples where dense, entangled contacts are consistently encountered, such as the geared system in Fig. 1 and the interlocked collision of complex models in Fig. 2. As a representative improvement we note that in the latter example the ship and the helicopter models have 227K and 116K surface triangles respectively; here the three-level hierarchical i-AABB is over 90% more efficient in culling surface pairs in comparison to per-body BVH.

4.3 Contact-Aware Hessian Construction

It is, by far, most efficient to simulate a set of noncontacting affine bodies. Along with reduced cost for collision detection, the global (system-wide) Hessian for E is then simply a $12|\mathcal{B}| \times 12|\mathcal{B}|$ block-diagonal matrix with just a *separable* 12×12 block for each affine body's mass and orthogonality energy ($V_{\perp}$) contributions. When the system includes contacts, however, off-diagonal terms from active, inter-body contact and friction potentials necessarily pollute the Hessian to account for contact coupling.

Standard FEM-type evaluation and assembly would suggest iterating across all active contact potentials. However, here the surface mesh resolution, and so the corresponding number of surface pairs forming contact potentials, is generally much larger than the number of affine bodies we simulate. Current assembly, in this contact-oblivious way, is neither necessary nor efficient. We instead integrate our Hessian evaluation and assembly with the i-AABB hierarchy for contact-aware parallelization in both multi-core CPU and GPGPU implementations.

We start with the easiest part. The global Hessian's default non-zero diagonal blocks are given by a constant mass term and the orthogonality potentials' Hessian (Eq. (8)). This can be computed trivially in parallel. Next comes the expensive part. The Hessian of the contact potential, V_C, then has an unpredictable pattern which varies widely with contact states. Recall that, when bodies i and j are sufficiently close ($\leq \hat{d}$), barrier and friction forces between them activate, resulting in non-zero contributions to both their respective i-th and j-th 12×12 diagonal blocks *and* to the off-diagonal blocks linking the corresponding body coordinates in the Hessian.

Here we observe that the configuration-varying sparsity of the global Hessian is effectively determined by our i-AABB culling. If the leaves of the i-AABB tree between bodies i and j are empty, they certainly do not contact, and the barrier and friction potentials make no contribution to the Hessian. Otherwise, we can conservatively allocate space for the corresponding 12×12 off-diagonal blocks to store all potential (active contacts are not yet certain) non-zero contact Hessian contributions between bodies i and j. Thus, utilizing the i-AABB structures we apply a two-pass strategy to compute and assemble barrier and friction terms for the global Hessian. The first pass iterates across all surface primitives pairs within each i-AABB; their local Hessians are computed in parallel and cached. Here we also account for the Hessian's symmetry to reduce total memory consumption. Our second pass then accumulates the local Hessians from the surface pairs. Here, as each culled i-AABB is independent and non-intersecting, accumulation is parallelized at each corresponding non-zero element of the global Hessian.

We find that i-AABB Hessian construction is significantly faster than default sequential parallelization of Hessian computation (e.g., as applied in rigid-IPC). We observe speedups of up to two orders, especially when the contacting system is composed of large numbers of bodies. Here, for example, the simulation in Fig. 3 provides a representative example, with a 188× speedup over contact-oblivious, sequential Hessian construction.

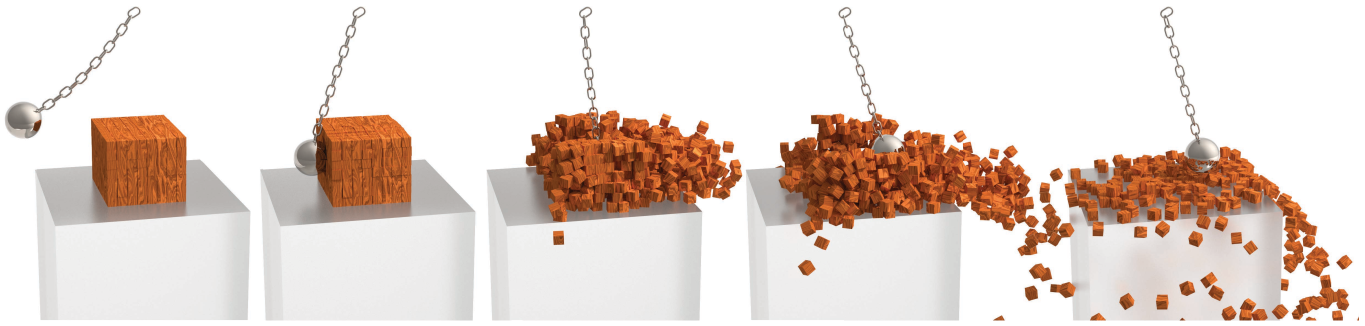

Fig. 3. **Wrecking ball.** A metal ball linked to a chain of rigid rings hits a stack of 560 wooden blocks. There are in total 575 bodies including the ball, rings on the chain, and blocks in this example. ABD and rigid-IPC yield nearly identical simulation results but ABD is 124× faster than rigid-IPC (both on CPU). The time step is $\Delta = 1/100$ sec. We also tested ABD using $\Delta = 1/50$ sec and $\Delta = 1/25$ sec respectively. The results are similar – all are free of inter-penetration.

5 EVALUATION

Our implementation platform is a desktop PC with an `intel i9 11900K` CPU (8 cores, 3.5GHZ), 64G memory and an `nVidia 3090` GPU. An overall flowchart of ABD is illustrated in Fig. 4. All the numerical methods were implemented using `C++` on CPU, and we chose `Eigen` [Guennebaud et al. 2010] as our primary linear algebra library, including all sparse linear system solves. Our CPU parallelization utilizes `intel TBB`. We understand that some other BLAS libraries may be better optimized for our hardware (e.g., `intel MKL` [Wang et al. 2014]). Our choice is to ensure an objective comparison with rigid-IPC, whose multi-thread CPU implementation is also based on `Eigen` and `TBB`.

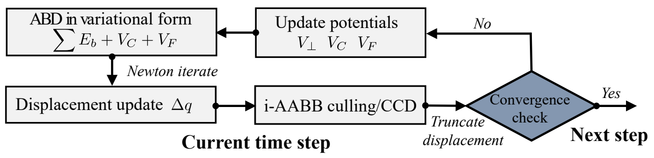

Fig. 4. **An overview of ABD pipeline.** The overall algorithmic flowchart of ABD is similar to the IPC framework. With the aim of optimizing the global variational function (Eq. (9)), each displacement update from a Newton solve undergoes the triangle-level CCD to ensure bodies are interpenetration-free.

For benchmarking we focus on rigid-IPC [Ferguson et al. 2021] for comparison to a "quality-oriented" rigid-body simulator with comparable guarantees to ABD and, as a representative baseline for comparison to state-of-the-art, optimized rigid-body libraries we use `Bullet` [Coumans 2015]. There are certainly numerous other, highly effective rigid-body libraries with varying capabilities. However, trade-offs with respect to `Bullet` among alternatives have been extensively documented by Ferguson et al. [2021]. In their comprehensive analysis and benchmark testing of rigid body libraries `Bullet` most consistently succeeds across challenging examples and, at the same time, we also note that `Bullet` is likely the most widely deployed rigid body simulation solution.

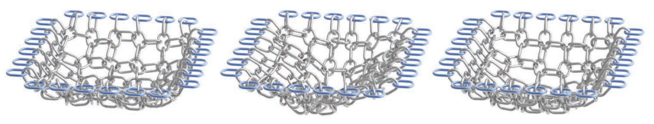

Fig. 5. **Small chain net.** We simulate a small-size chain net consisting of 8 by 8 rings using both ABD and rigid-IPC. The exterior rings are bound with a loop of fixed blue rings. In this simple test, ABD is 34× faster than rigid-IPC on the CPU. An increase of the time step size from 1/100 sec to 1/50 sec does not noticeably slow down our method (from 0.059 sec to 0.065 sec per frame), but rigid-IPC will be significantly slower (by 350%).

5.1 Comparison with Rigid-IPC

Both ABD and rigid-IPC [Ferguson et al. 2021] utilize the IPC [Li et al. 2020] model for contact processing and friction modeling. Here we carefully compare ABD with rigid-IPC across several representative simulation scenarios of varying complexities, as illustrated in Figs. 3. 5, 6, 8, and 7.

The snapshots of the first comparison are given in Fig. 3. In this test, a heavy ball linked by a chain of metal rings collides into a stack of wooden blocks, which are scattered by the collision. There are in total 575 bodies in this example. Both rigid-IPC and ABD produce high-quality simulation results without inter-penetration. However, ABD is 124× faster. The performance of ABD is not sensitive to time step size. Doubling or even quadrupling the time step size (from 1/100 sec to 1/50 sec and 1/25 sec) lead to similar results and performance with ABD. Detailed timing statistics can be found in Section 5.6. We observe similar behavior for the chain net examples. Finally, for the smallest chain net example (Fig. 5), we have few bodies, and here the relative velocities among bodies are small. In this "entry-level" test, ABD offers a 34× speedup.

We report an "upgraded" experiment in Fig. 6, where the resolution of the chain net is set to 16 × 16 and we drop a heavy ball, which bounces back and forth on the net triggering interesting dynamics. The collisions and contacts then become more complex than in the simple chain net test in Fig. 5. Here the difference between our ABD and rigid-IPC becomes more significant. At some instances, when sharp collisions occur under high relative velocities, rigid-IPC can take multiple hours to simulate a single frame ($\Delta t = 1/100$ sec). On the other hand, ABD requires seconds at most. Here, on average, ABD is over 1,200× faster than rigid-IPC. This speedup exceeds

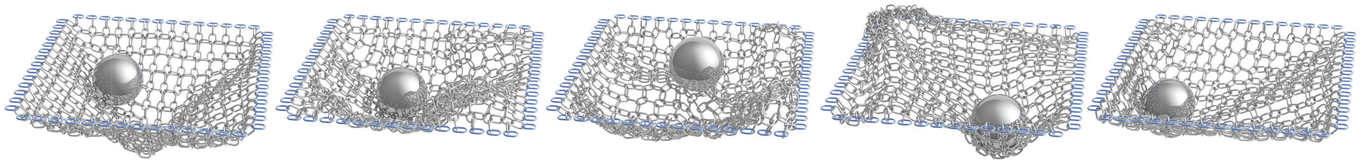

Fig. 6. **Big chain net.** We increase the resolution of the chain net to 16×16 and drop a ball to the net. The ball hits the net and bounces back, which leads to interesting dynamical responses of the net. While both ABD and rigid-IPC are able to handle this simulation robustly, our method is $1,200\times$ faster on average. The speedup could reach over $10,000\times$ on GPU with CUDA.

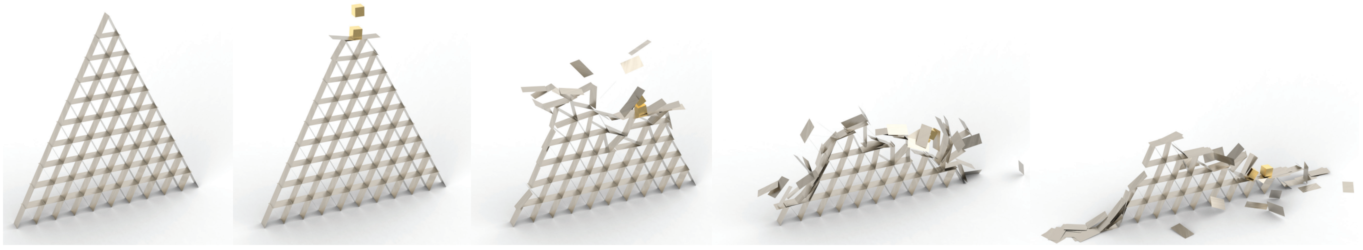

Fig. 7. **House of cards.** The 10-level stack of cards is initially balanced by frictions among cards. Two falling boxes break the balance and crash the stack. This experiment mixes static and dynamic frictions involving 158 bodies. Our method is $103\times$ faster than rigid-IPC.

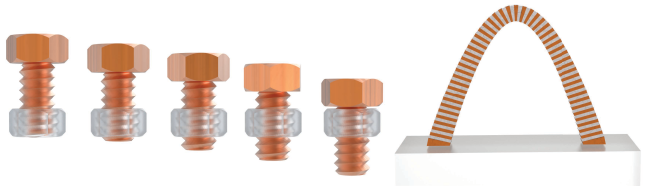

Fig. 8. **Friction test.** While ABD and rigid-IPC use the same barrier-based friction processing method, ABD is much faster than rigid-IPC under the same simulation settings. The screw example (left) is a representative demonstration of dynamic friction/contact. Our speedup is $30\times$. On the other hand, the arch example (right) is dominated by static frictions, and ABD is $77\times$ faster than rigid-IPC.

$4,000\times$ from time to time during the simulation. In ABD, affine CCD, i-AABB, as well as the integrated Hessian assembly are all parallelization-ready. Therefore, ABD typically receives one additional order of performance gain on CUDA, gaining a total $10,000\times$ speed-up over rigid-IPC (albeit CPU-based) in Fig. 6.

ABD and rigid-IPC both use the IPC-based variational friction model [Li et al. 2020]. Nevertheless, ABD still exhibits superior performance in simulations dominated by friction. We here consider three experiments with significant frictional contact in Figs. 8 and 7. In these tests, both ABD and rigid-IPC use the same simulation settings with a time step of 1/100 sec. The screw example (Fig. 8 left) is relatively simple – the surface geometry has just 7K triangles and 5K edges. Here the bolt rotates into the fixed nut. On average, rigid-IPC takes 2.59 sec to simulate one step, while ABD takes just 87 ms. The arch (Fig. 8 right) consists of 100 blocks. In this example, we mainly resolve static friction and here Rigid-IPC runs faster than the screw case, taking 0.67 sec to simulate one step. ABD, however, uses just 8.7 ms. Finally, the house of cards example starts with a 10-level stack of 155 cards. The stack is first stands stable with friction and is then collapsed by two falling boxes. In this example, rigid-IPC takes approximately 8.9 sec per step, while ABD uses 86 ms.

5.2 ABD under various stiffness

Our use of the potential, $V_\perp$, to maintain stiff material behavior is key for ABD. If $V_\perp$ is not strong enough, affine deformations become noticeable on the model. On the other hand, making $V_\perp$ unecessarily stiff certainly slows convergence. In our implementation we set κ in $V_\perp$ (Eq. (7)) between 100 − 200GPA based on common stiff materials like bronze and steel. Under this setting, affine deformation is negligible and well matches the behavior of real-world stiff objects. Nevertheless, we note that ABD is flexible in choice of κ, and the system convergence is relatively consistent across a wide range of stiff κ settings.

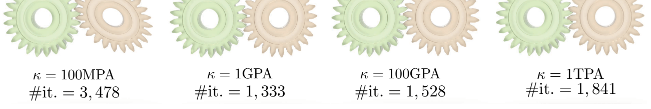

Fig. 9. **ABD with different κ.** We use ABD to simulate a pair of coupled gears under different κ values. The time step size is $\Delta t = 1/100$ sec, and $\hat{d} = 0.001$ m. A small κ leads to observable affine deformations. Increasing κ quickly eliminates this artifact. The convergence is lightly impacted by increased stiffness. Fortunately even if we set the κ $10\times$ bigger than the default value of 100GPA, the convergence slow remains reasonable.

In Fig. 9, we simulate a pair of gears coupled at their teeth with ABD. Both gears have a diameter of 0.5 meter with $\hat{d}$ and Δt being 1 millimeter and 1/100 sec. When κ is small (i.e., $\kappa = 100\text{MPA}$),

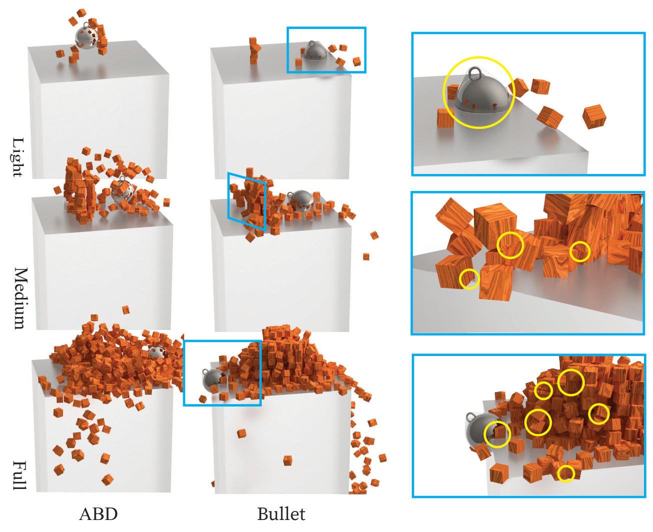

Fig. 10. **ABD vs. `Bullet`.** We compare simulation results using ABD and `Bullet` with varying numbers of bodies and time step sizes. `Bullet` often produces interpenetrations between bodies even under small time steps. Interpenetrations becomes increasingly severe with growing body counts and/or time step size. On the other hand, ABD remains intersection-free across changing parameters and scene complexities. All timing statistics are reported in Tab. 1.

we clearly see affine deformation in the orange gear. Such large deformations also contribute to a more expensive CCD and therefore a slower convergence. The total iteration count for simulating 500 time steps is 3, 478. Increasing κ to 1GPA effectively suppresses this artifact, and the total iteration count reduces to 1, 333, which is slightly more than two iterations per step. Under our default setting with κ = 100GPA, the total number of iterations increases to 1, 528 barely hurts the overall simulation performance. We further escalate the stiffness to κ = 1TPA i.e., an order larger than the default setting. ABD is still able to efficiently simulate such a highly stiff scene, and the total iteration count reaches 1, 841.

5.3 Comparison with `Bullet`

`Bullet` is a popular and optimized rigid-body engine based on velocity-level LCP modeling. `Bullet` primarily relies on contact-resolution via proxies of surface geometries formed by convex decomposition. It is then not surprising to see `Bullet` failures in simulations under persistent and/or intensive collisions and contacts. We have demonstrated that our method is orders of magnitude faster than rigid-IPC with further improved robustness. For smaller simulation problems, ABD is nearly as efficient as `Bullet` while preserving guarantees. As scene complexity increases in then quickly outperforms `Bullet` from almost all perspectives.

We compare ABD and `Bullet` using the wrecking ball setup (see Fig. 3) but under different block counts and time step sizes. In this set of comparisons the ball is no longer attached to the chain as in Fig. 3. This is because under high-velocity movements, collision resolution in `Bullet` frequently fails, with the rings on the chain breaking so that the ball can not hit the stack. Results are reported

Table 1. **ABD vs. `Bullet` timing.** We record timing information of simulating the wrecking ball scenarios under different block counts and time step sizes using ABD and `Bullet`. # **Bdy** is the number of bodies in the test. # **Tri./Edg.** gives the total numbers of triangles and edges on the surface of the models. **Δt** is the time step size. # **Iter.** is the average iteration counts in ABD simulation. **Time** reports the average computation time for each frame using **ABD** and `Bullet`. This timing comparison is reported on a single thread implementation.

Test	# Bdy	# Tri./Edg.	Δt (sec)	# Iter.	Time (ms)
Light	16	1.2K/796	1/100	1.9	**3** | 2
			1/240	1.5	**2.2** | 1.5
			1/1000	1.1	**2** | 3
Medium	142	3.5K/2.3K	1/100	7.6	**92** | 68
			1/240	2.9	**41** | 58
			1/1000	1.3	**19** | 82
Full	562	11K/7.3K	1/100	11.0	**657** | 629
			1/240	4.4	**328** | 809
			1/1000	1.8	**102** | 804

in Fig. 10, where we have three configurations: light, medium, and full. The light test only has 16 blocks on the stack; the medium test has 142 blocks; and the full test is the same as in Fig. 3 with 560 blocks.

A detailed timing comparison is listed in Tab. 1. In order to ensure the comparison is fair, we turn off multi-threading in ABD and `Bullet`. This is because our method processes collision on the surface triangles, which will receive more acceleration under parallelization. For the light test (Δt = 1/100 sec), `Bullet` is efficient and only needs 2 ms to simulate one frame. Here ABD is slower than `Bullet` and takes 3 ms to simulate one frame. This difference diminishes under a smaller time step. For instance, when the time step size is set as 1/240 sec, which is the default setting in `Bullet`, the difference of per-frame simulation time is less than one millisecond (2.2 ms for ABD and 1.5 ms for `Bullet`). If the time step size is more conservatively set to 1/1000 sec, ABD becomes faster than `Bullet` by a small margin (2 ms for ABD and 3 ms for `Bullet`). However, *simulations using* `Bullet` *in all time step sizes have intersections* (even with Δt = 1/1000 sec). Our method on the other hand is guaranteed to be free of inter-penetration. Another interesting observation is per-frame simulation using ABD becomes more efficient under smaller time steps, which is not the case for `Bullet`. This may be because a smaller time step could expose more collisions to the `Bullet` solver, which are otherwise missed under a bigger step. In addition, `Bullet` fails all the friction experiments including the screw, arch, and house of cards (Figs. 8 and 7) even with highly conservative time step size (Δt = 1/10000 sec).

The story is similar for the medium/full test: ABD (92 ms in medium and 657 ms in full) is slightly slower than `Bullet` (68 ms in medium and 629 ms in full) when Δt = 1/100 sec[2]. ABD then takes the lead under smaller time steps of 1/240 sec and 1/1000 sec. In addition, `Bullet` has significantly more inter-penetration instances in the medium/full tests than in the light test, which generate artifacts unsuitable for many simulation tasks.

As discussed, ABD's collision processing directly handles each body's input surface mesh boundary with all guarantees, including

[2]ABD and `Bullet` have essentially the same FPS in the full test with Δt = 1/100 sec

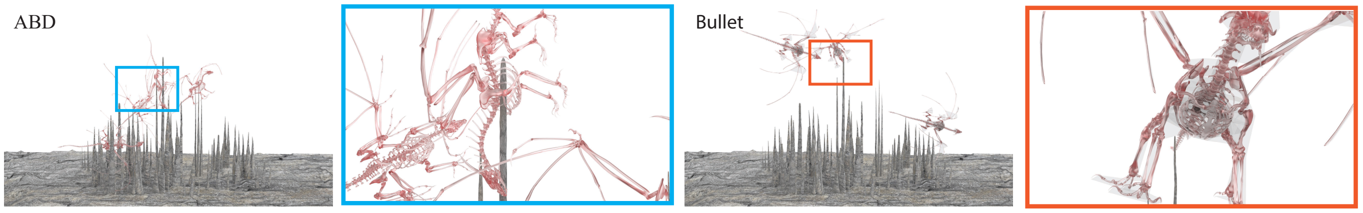

Fig. 11. **Bone dragon.** In addition to the simulation algorithm, another difference between ABD and `Bullet` is the collision detection. ABD allows triangle-level collision detection to capture local contacts between fine geometries. `Bullet` is mesh-based and relies on convex decomposition of the model. In some cases, the decomposition is not accurate enough to approximate concave shapes. This figure shows one of such examples. Both bone dragons and rock spikes are sharp and concave.

non-intersection, applying directly to those meshes. This means that control of resolution and so quality is available via standard, well optimized geometry pipelines, e.g., mesh decimation. `Bullet`, however, requires pre-processing to convert all input meshes into convex decompositions. In turn `Bullet` will then simulate these decomposed, simplified models as proxies, resolving collision handling on them, for each rigid body's surface rather than the original mesh. `Bullet` users often use V-HACD [Mamou et al. 2016], an automated library for computing convex decomposition, for this process. Utilizing libraries like V-HACD, can be a slow and often time-consuming iterative process to hand-tune quality parameters in order to obtain a reasonable geometric approximation and resolution. Even so important features can be lost while details and symmetries are often unnecessarily broken. Here, we demonstrate a comparative example of utilizing detailed, bone dragons in Fig 11 dropped on a highly featured geometry. After careful hand-tuning of V-HACD parameters, we present in Fig 11 right the resulting simulation using our best resulting V-HACD proxies. Here we see the convex decomposition geometry is still insufficiently detailed to capture local collision behavior between sharp asperities and convexities. In contrast, in Fig 11 left, ABD directly simulates the detailed geometry with tight tolerance so that all the original input meshes' detailed affordances are kept. In turn, we see that this allows the dragons' concavities to tightly entangle and also slide directly onto and be caught by the sharp points – all while remaining intersection free.

The `Bullet` library also incorporates a mesh-based contact-resolution collision-resolution (non-default) option with `GImpact`. In all the tests we performed, `GImpact` produces large interpenetrations and pass-throughs, even under a highly conservative time step sizes (e.g., $\Delta t = 1/1000$).

5.4 Joint Constraints are Linear for Affine Bodies

An additional benefit of ABD is convenient constraint handling – especially for the those prescribing rotational DOFs. For instance, it is common to require multiple objects to obey a given kinematic relation following the joint connecting them. Such constraints are often nonlinear as they are formulated from rigid body rotational DOF. Enforcing them would then require computing the derivatives of the rotational DOFs (either via constrained Lagrangian methods or generalized coordinates). Here relaxation from rigid to affine also eases the processing of these constraints.

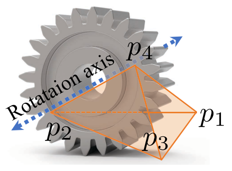

Fig. 12. **ABD constraint handling.** An axis constraint becomes linear and can be easily enforced in ABD framework.

It is known that a non-degenerate tetrahedron uniquely defines an affine transform. This suggests our generalized coordinate q can be mapped to *any* linear tetrahedron. From this perspective, an affine body simulation can also be viewed as a single-element FEM (with a simplified strain energy). The geometry of this element however, can be setup flexibly even if its position deviates away significantly from the object. Let ϕ denote the map between q and this virtual tetrahedron such that $q = \phi(\mathrm{P})$. $\mathrm{P} \in \mathbb{R}^{3\times4}$ stores the deformed vertex positions p_1, p_2, p_3, and p_4 of the element. Given its rest shape position $\bar{\mathrm{P}}$, one can easily verify that:

$$\phi(\mathrm{P}) = \left[\frac{1}{4}\sum_i (p_i - \bar{p}_i)^T, \mathrm{vec}^T\left(\mathrm{P}\bar{\mathrm{P}}^T(\bar{\mathrm{P}}\bar{\mathrm{P}}^T)^{-1}\right)\right]^T. \tag{13}$$

Here, vec$(\cdot)$ denotes the vectorization of a matrix. Since $\bar{\mathrm{P}}$ is given, we could use vec(P) as the new generalized coordinate of the system. Clearly ϕ is a linear function of P. $\partial\phi/\partial p_i$ only depends on $\bar{\mathrm{P}}$. Therefore, the Jacobi of the system remains constant:

$$\mathrm{J}_i = \frac{\partial x_i}{\partial q}\cdot\frac{\partial\phi}{\partial \mathrm{vec}(\mathrm{P})} \in \mathbb{R}^{3\times12}. \tag{14}$$

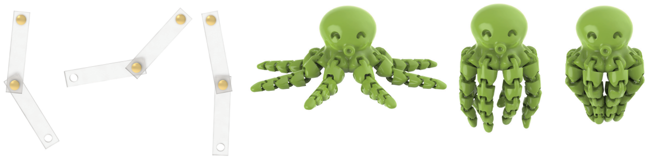

Fig. 13. **Constrained simulation.** Prescribing rotation freedom in ABD is convenient. We visualize the generalized coordinate as a virtual tetrahedron, and most rotational constraints are linearized. We show two examples of such simulation in this figure. Both pendulum and octopus tentacles are made of multiple bodies connected via hinge joints. ABD remains significantly faster than rigid-IPC: ABD achieves a 371× speedup for the pendulum and a 28× speedup for the octopus.

In practice, we exploit the fact that $\bar{\mathrm{P}}$ could be any tetrahedron to manipulate such constraints intuitively. For instance as shown

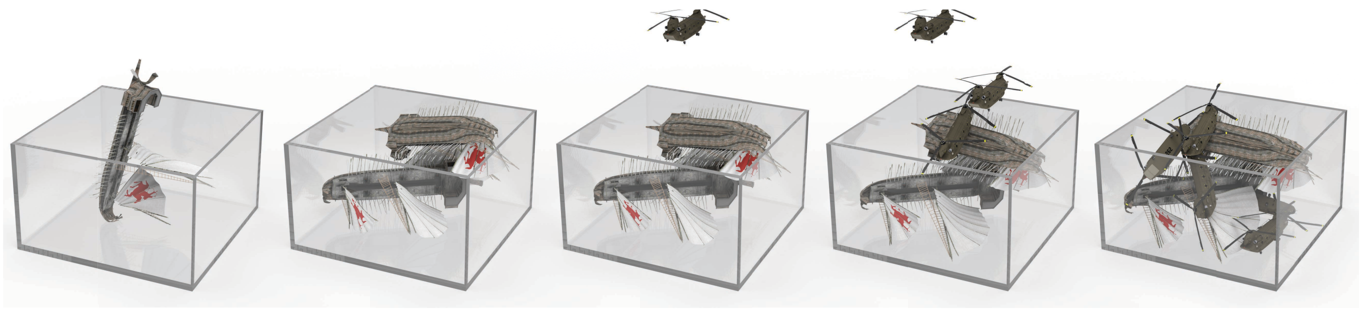
Fig. 14. **Hybrid simulation.** ABD is particularly convenient for simulating hybrid object with both rigid and deformable parts. Both the barbarian ship (225K triangles, 341K edges) and helicopter (116K triangles, 176K edges) are such hybrid objects with a rigid main body appended by several deformable parts (canvas, rotors, and wheels). We use standard neo-Hookean FEM to simulate deformable parts, which are constrained to the virtual tetrahedron corresponding to the rigid body. Due to the involvement of massive DOFs from deformable parts, the simulation uses about 16 sec for one frame under $\Delta t = 1/100$ sec.

in Fig. 12, the gear is constrained to rotate around a prescribed axis (blue dash line). In ABD, we map q to a tetrahedron and make sure that one of its edges is parallel to the rotation axis. The axis constraint can then be simply posed as a linear equality constraint fixing two vertices of the edge that is parallel to the axis (p_2 and p_4 in the figure). Indeed, such formulation is over-constraining as it eliminates six freedoms instead of two (we should also allow p_2 and p_4 to move along the edge). However, this hypothetical "over-constraint" still gives correct simulation because ABD has more DOFs to simulate than a rigid object. Other constraint types can be resolved similarly.

Here we demonstrate two applications with joint in Fig. 13. The pendulum is a simple mechanism with two rigid links constrained by a hinge. The octopus has eight tentacles, each composed with five joints. We also compared ABD with rigid-IPC with these models. ABD does not only provide a simple formulation but also runs significantly faster (371× and 28× speedups for the pendulum and octopus respectively). A more challenging experiment is reported in Fig. 1, where we simulate a more complex mechanism with a set of 28 gears. These gears are coupled by tooth contacts and shafts (see Fig. 16). There are $2,450$K triangles and $3,090$K edges on the surface (i.e., those that could participate in culling and CCD). A component-wise breakdown showing the inter-connectivity of gears is given in the right of the figure. As the driving torque is applied at one of the gear (the one with red arrow), the entire device moves forward. This simulation is a stress test for simulation algorithm robustness. Because gear teeth are close to each other, highly localized and detailed CCD is massively used. All rigid body simulation algorithms we have tested fail in this case including rigid-IPC, `MuJuCo`, and `Bullet`. As mentioned, `Bullet` requires building a convex proxy for collision processing, which does not capture the zigzag geometry at the gear tooth. `MuJuCo` fails the test even under highly conservative settings (e.g., very small time step size). Rigid-IPC also fails this experiment: after the first few steps the Newton iteration loops infinitely because the curved CCD is unable to find a usable TOI (time of impact). ABD simulates this scene without issues. Each frame takes about 12 sec on the CPU. As the majority of the computation is in CCD processing, we port our i-AABB algorithm to `CUDA`, and here the simulation of the gear set reaches an interactive rate (at 5 – 10 FPS).

5.5 Hybrid Simulation

Another benefit of ABD, when viewed as a reduced single-tetrahedron FEM, is to simulate models with both extremely stiff (close-to-rigid) and soft components. While existing rigid-body methods could be augmented to incorporate such hybrid simulations via coupling, it is particularly seamless and simple in the ABD framework. In Fig. we demonstrate a hybrid simulation of two barbarian ships each with a rigid ship body and two deformable canvas sails. Each ship has 225K surface triangles and 341K edges. The scene collides the ships with helicopters that are also hybrid with a rigid body, two soft rotors, and four soft wheels. There are 116K triangles and 176K edges on each helicopter. After two ships fall into the glass tank, five helicopters follow, producing interesting effects combining both rigid and deformable dynamics. Note that the collision between stiff and soft bodies can be handled uniformly using the barrier-based penalties. In these hybrid simulations, the Hessian of elastic potential on the deformable components are much smaller (by several orders) than the Hessian of the orthogonality potential ($V_\perp$), and the positive definiteness of the global system matrix can be numerically compromised. Therefore, LLT factorization (`SimplicialLLT`) may fail occasionally. In this case, we switch to LDLT Cholesky for each Newton solve. A possible remedy for increase numerically stability is to use Schur complement like formulation [Peiret et al. 2019] to potentially decouple DOFs from affine and deformable components.

5.6 Timing and Breakdown

Detailed timing statistics are reported in Tab. 2. In all experiments, we uniformly scale the scene to a $1 \times 1 \times 1$ box and set $\hat{d}$ as $1/1000$. In other words, if the size of the model is around one meter, the contact accuracy is guaranteed to be less than one millimeter, while all model trajectories remain intersection free throughout simulations. We report the comparative timing benchmark of ABD and rigid-IPC under $\Delta t = 1/100$ in most experiments. Normally, rigid-IPC remains numerically robust under larger time steps. However, we see its performance is highly sensitive to a larger Δt. This is because the underlying curved CCD quickly becomes prohibitive

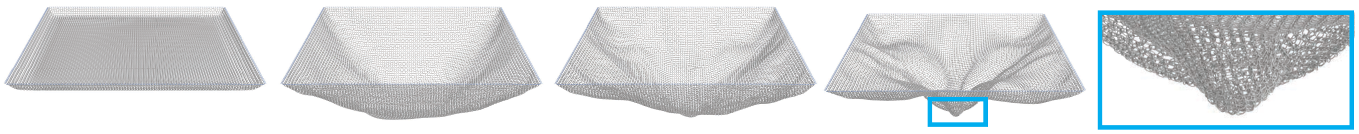

Fig. 15. **Huge chain net.** When the body count increases, the disadvantage of using redundant DOFs in the simulation may become more obvious. Following this thought, we up scale the chain net to 27,645 bodies. In this stress test, A-IPC takes about 5 min to simulate one frame on average. The entire simulation takes about five *days* with A-IPC. We are never able to finish this test with rigid-IPC. Based on our observation, A-IPC is at least 2,000× faster than rigid-IPC. This means it will need several *years* for rigid-IPC to finish this experiment.

Table 2. **Time statistic.** Detailed time statistics of our experiments. In most experiments, we tested ABD under three different Δt settings namely 1/100, 1/50, and 1/25. # **Bdy** (number of bodies), # **Tri./Edg.** (numbers of triangles and edges), $\boldsymbol{\Delta t}$ (time step size), # **Iter.** (average per-frame iteration counts), and **Time** (total time for each frame) are the same measures as in Tab. 1. This table also reports computation time used for Hessian assembly (**Hess.**), total time used for solving the linear system in Newton method at each frame (**Sol.**), time used for triangle-level CCD (**CCD**), time used for building the collision pairs (**Cons.**), and other computations (**Misc.**) e.g., variables initialization, convergence check etc. In many examples (except for the last four examples), we give comparative timing information of both **ABD** and rigid-IPC. The **GPU FPS** is also reported for $\Delta t = 1/100$ ABD simulations.

Test	# Bdy	# Tri./Edg.	Δt (sec)	# Iter.	Hess. (sec)	Sol. (ms)	CCD (sec)	Cons. (sec)	Misc. (ms)	Time (sec)
Wrecking ball (Fig. 3)	575	14K/20K	1/100	**8.8** | 17.1	**0.023** | 10.3	**53** | 47	**0.028** | 4.9	**0.031** | 2.3	**8** | 53	**0.14** | 17.6 | **18** FPS
			1/50	**19.4**	**0.067**	**160**	**0.078**	**0.092**	**18**	**0.41**
			1/25	**42.6**	**0.18**	**444**	**0.23**	**0.027**	**43**	**1.1**
Small chain net (Fig. 5)	144	63K/95K	1/100	**2.1** | 4.0	**0.011** | 0.18	**6** | 2	**0.016** | 1.1	**0.017** | 0.73	**9** | 6	**0.059** | 2.1 | **143** FPS
			1/50	**2.4**	**0.014**	**7**	**0.023**	**0.023**	**10**	**0.07**
			1/25	**2.8**	**0.018**	**9**	**0.29**	**0.032**	**10**	**0.08**
Big chain net (Fig. 6)	673	445K/297K	1/100	**13.5** | 169	**0.13** | 109	**241** | 437	**0.41** | 305	**0.43** | 388	**62** | 1336	**0.78** | 944 | **5** FPS
			1/50	**24.9**	**0.25**	**440**	**0.82**	**0.87**	**83**	**2.4**
			1/25	**26.3**	**0.27**	**464**	**0.93**	**0.96**	**86**	**2.7**
House of cards (Fig. 7)	158	336/816	1/100	**9.8** | 66.5	**0.014** | 2.49	**24** | 41	**0.009** | 5.1	**0.035** | 1.1	**4** | 46	**0.086** | 8.9 | **42** FPS
			1/50	**13.3**	**0.024**	**37**	**0.014**	**0.043**	**20**	**0.13**
			1/25	**25.2**	**0.045**	**64**	**0.035**	**0.074**	**9**	**0.24**
Screw (Fig. 8, left)	2	7.9K/5.2K	1/100	**4.1** | 8.2	**0.0004** | 0.028	**0.1** | 0.1	**0.039** | 2.5	**0.039** | 0.096	**9** | 1	**0.087** | 2.6 | **1K** FPS
			1/50	**3.2**	**0.0007**	**0.1**	**0.068**	**0.068**	**10**	**0.14**
			1/25	**5.7**	**0.001**	**0.1**	**0.11**	**0.12**	**10**	**0.23**
Arch (Fig. 8, right)	101	1.2K/1.8K	1/100	**1.9** | 6.2	**0.002** | 0.13	**2** | 44	**0.001** | 0.44	**0.002** | 0.099	**2** | 3	**0.0087** | 0.67 | **500** FPS
			1/50	**3.2**	**0.003**	**2**	**0.001**	**0.003**	**2**	**0.013**
			1/25	**5.7**	**0.005**	**3**	**0.003**	**0.006**	**3**	**0.022**
Pendulum (Fig. 13, left)	4	1.3K/2.0K	1/100	**4.1** | 4.3	**0.001** | 0.018	**0.1** | 0.1	**0.003** | 2.5	**0.003** | 0.03	**1** | 1	**0.007** | 2.6 | **1K** FPS
			1/50	**4.3**	**0.001**	**0.1**	**0.004**	**0.003**	**1**	**0.009**
			1/25	**4.7**	**0.001**	**0.1**	**0.005**	**0.004**	**1**	**0.01**
Octopus (Fig. 13, right)	41	33K/49K	1/100	**4.6** | 4.3	**0.012** | 0.071	**2** | 0.5	**0.02** | 0.97	**0.019** | 0.34	**5** | 3	**0.05** | 1.4 | **333** FPS
			1/50	**5.6**	**0.014**	**0.2**	**0.03**	**0.025**	**6**	**0.07**
			1/25	**6.7**	**0.017**	**0.2**	**0.04**	**0.03**	**6**	**0.09**
Hybrid sim. (Fig. 5.5)	19+8	1.1M/1.6M	1/100	**16.7** | –	**4.3** | –	**4600** | –	**3.5** | –	**3.4** | –	**490** | –	**16.4** | – | **0.4** FPS
			1/50	**19.3**	**5.8**	**5311**	**4.8**	**4.5**	**562**	**21.3**
			1/25	**25.9**	**7.9**	**7083**	**7.2**	**6.9**	**836**	**30.0**
Gear set (Fig. 1)	28	2.5M/3.1M	1/100	**16.7** | –	**4.3** | –	**4** | –	**5.4** | –	**6.0** | –	**211** | –	**11.7** | – | **7.3** FPS
			1/50	**19.3**	**5.8**	**4**	**6.3**	**7.9**	**219**	**14.5**
			1/25	**25.9**	**7.9**	**6**	**9.3**	**11.2**	**43**	**20.1**
Bone dragon (Fig. 11)	29	1.2M/1.7M	1/100	**6.3**	**0.003**	**0.2**	**2.3**	**0.56**	**74**	**2.8**
Huge chain net (Fig. 15)	27,645	12M/18M	1/100	**14.2**	**37.2**	**87 sec**	**95**	**94**	**7.2 sec**	**310**

when searching over wider trajectory gaps. Therefore, we do not report our speedup over rigid-IPC for any time steps larger than $\Delta t = 1/100$ sec. For the chain net example in Fig. 6, the literal speedup exceeds four-orders in general if we set $\Delta t = 1/25$. We do not push to this comparison by unnecessary tweaking of the time step size to these extremes.

The primary portion of our performance improvement is gained by relaxing the rigidity constraint. This can be observed in two important metrics in Tab. 2: the iteration count (# **Iter.**) and the time needed for CCD processing (**CCD**). Here we see that as long as the contact frequency in the simulation is intense, ABD always requires much fewer iterations to converge than rigid-IPC does. This difference is "extremized" in the gear set example (Fig. 1), where ABD needs 17 iterations for each time step, and rigid-IPC needs does not converge. CCD processing is another major game changer. As mentioned, rigid-IPC strictly follows the rigidity constraint making per-step trajectory curved. This curved trajectory is split and converted back to piece-wise line segments again during the CCD. This conversion is fully avoided in ABD, and one can use any existing CCD algorithm to detect potential intersection between primitives and to compute TOI. Recall that CCD must be applied at every Newton iteration in order to ensure line search does not introduce

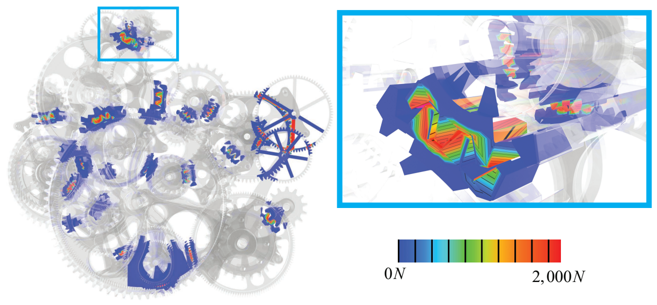

Fig. 16. **Contact forces at teeth.** With the assistance of i-AABB, ABD efficiently and accurately model the contact forces among gear teeth. This figure visualizes the force distribution over the gear teeth.

inter-penetrations. Therefore, the performance gap between A-IPC and rigid-IPC is further scaled by iteration count. Together, these two factors contribute to 90% of our speedup in complex simulations such as the big chain net (Fig. 6), house of cards (Fig. 7) and wrecking ball (Fig. 3). In other, relatively lighter simulation experiments, our improved culling and Hessian assembly become more profitable. Our culling is up to to 90% more effective than conventional BVH-based strategy for models with complex geometry. It is on average 30% more effective for simpler shapes (e.g., the chain net or the octopus). In terms of system solve, ABD is typically slower than existing rigid body methods due to inflated DOFs. However, this disadvantage is invisible because ABD always enjoys much fewer iterations and much faster CCD processing. This is our core inspiration in designing ABD.

Given the above analysis it is reasonable to wonder if increasing numbers of bodies in a scene would grant a lead to rigid-IPC at a certain point. To this question we simulate a very large chain net model with 27, 645 bodies (Fig. 15). Here the result is opposite – we estimate ABD obtains a $\sim 5,000\times$ speedup (on CPU). This is just a rough assessment as we are unable to finish rigid-IPC simulation in this stress test, which would require *several years* to complete currently while ABD finishes the simulation in *five days*.

6 CONCLUSION

We have introduced a new, simple affine dynamics model and a carefully customized, easy-to-implement affine IPC algorithm for the simulation of extremely stiff materials with fidelity, convergence and reliability. The resulting method is highly suited for simulating all scenarios and applications where currently rigid body methods are now popularly employed *without*, as we have shown, the current limitations that rigid body models impose. Here we have demonstrated that ABD obtains orders of magnitude speedup over state-of-the-art rigid body simulation with comparable guarantees of non-intersection and convergence. At the same time we have also shown that ABD obtains both comparable (for easy examples) and improved (as scene complexity grows) speeds when compared with highly optimized rigid body libraries that do not have guarantees and so suffer from artifacts and all-out failures that limit their automated use. While ABD has demonstrated superior robustness and efficiency, it is, of course, based on numerical discretization of underlying differential equations. Therefore, ABD could and certainly fail for extreme simulation conditions. For instance, affine CCD certainly becomes inaccurate if one rotates a rotor blade for 180 degrees within a single time step.

ABD is custom-suited for parallelization, is also differentiable, and automatically and directly simulates all input triangulated geometries. We have shown that, when leveraging the GPU, ABD can simulate complex contacting systems at interactive rates. With these combined properties it is then exciting to consider future applications where ABD's automation, reliability, and differentiability can be utilized for computational design, machine learning, and robotics. In these cases consistent, artifact-free simulation behavior across shape, material and contact variations, without algorithm parameter tuning, should accelerate development. We have also shown a few initial, proofs-of-concept for extensions of ABD to both complex, jointed stiff multibody systems, and to hybrid stiff/flexible multibody systems. Here there also clearly remains significant opportunities for further development and application of ABD for jointed and hybrid systems. Finally, looking ahead, with the popularity and diverse applications of physical modeling we hope that ABD will provide the rapidly growing and diverse community of simulation users with a reliable, differentiable and exceedingly efficient framework suitable to swap in for all rigid body-type applications.

ACKNOWLEDGMENTS

We thank anonymous reviewers for their constructive comments. Yin Yang and Lei Lan are partially supported by National Science Foundation (No. 2011471, 2016414). Chenfanfu Jiang and Minchen Li are are partially supported by National Science Foundation (No. 2153851, 2153863, 2023780) and Department of Energy (No. ORNL 4000171342).

REFERENCES

Marc Alexa, Daniel Cohen-Or, and David Levin. 2000. As-rigid-as-possible shape interpolation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. 157–164.

Mihai Anitescu and Florian A Potra. 1997. Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. *Nonlinear Dynamics* 14, 3 (1997), 231–247.

David Baraff. 1989. Analytical methods for dynamic simulation of non-penetrating rigid bodies. In *Proceedings of the 16th annual conference on Computer graphics and interactive techniques*. 223–232.

David Baraff. 1991. Coping with friction for non-penetrating rigid body simulation. *ACM SIGGRAPH computer graphics* 25, 4 (1991), 31–41.

David Baraff. 1994. Fast contact force computation for nonpenetrating rigid bodies. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*. 23–34.

David Baraff. 1995. Interactive simulation of solid rigid bodies. *IEEE Computer Graphics and Applications* 15, 3 (1995), 63–75.

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

Jan Bender, Kenny Erleben, and Jeff Trinkle. 2014. Interactive simulation of rigid body dynamics in computer graphics. In *Computer Graphics Forum*, Vol. 33. Wiley Online Library, 246–270.

Gino van den Bergen. 1997. Efficient collision detection of complex deformable models using AABB trees. *Journal of graphics tools* 2, 4 (1997), 1–13.

Javier Bonet and Richard D Wood. 1997. *Nonlinear continuum mechanics for finite element analysis*. Cambridge university press.

Robert Bridson, Ronald Fedkiw, and John Anderson. 2002. Robust treatment of collisions, contact and friction for cloth animation. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*. 594–603.

Tyson Brochu, Essex Edwards, and Robert Bridson. 2012. Efficient geometrically exact continuous collision detection. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 1–7.

Michael B Cline and Dinesh K Pai. 2003. Post-stabilization for rigid body simulation with contact and constraints. In *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, Vol. 3. IEEE, 3744–3751.

Eulalie Coevoet, Otman Benchekroun, and Paul G Kry. 2020. Adaptive merging for rigid body simulation. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 35–1.

Erwin Coumans. 2015. Bullet physics simulation. In *ACM SIGGRAPH 2015 Courses*. 1.

Peter A Cundall and Otto DL Strack. 1979. A discrete numerical model for granular assemblies. *geotechnique* 29, 1 (1979), 47–65.

Evan Drumwright. 2007. A fast and stable penalty method for rigid body simulation. *IEEE transactions on visualization and computer graphics* 14, 1 (2007), 231–240.

Christian Duriez, Frederic Dubois, Abderrahmane Kheddar, and Claude Andriot. 2005. Realistic haptic rendering of interacting deformable objects in virtual environments. *IEEE transactions on visualization and computer graphics* 12, 1 (2005), 36–47.

Kenny Erleben. 2007. Velocity-based shock propagation for multibody dynamics animation. *ACM Transactions on Graphics (TOG)* 26, 2 (2007), 12–es.

Kenny Erleben. 2017. Rigid body contact problems using proximal operators. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 1–12.

Yu Fang, Minchen Li, Chenfanfu Jiang, and Danny M. Kaufman. 2021. Guaranteed Globally Injective 3D Deformation Processing. *ACM Trans. Graph. (SIGGRAPH)* 40, 4, Article 75 (2021).

Zachary Ferguson, Minchen Li, Teseo Schneider, Francisca Gil-Ureta, Timothy Langlois, Chenfanfu Jiang, Denis Zorin, Danny M Kaufman, and Daniele Panozzo. 2021. Intersection-free rigid body dynamics. *ACM Transactions on Graphics* 40, 4 (2021), 183.

Susan Fisher and Ming C Lin. 2001. Fast penetration depth estimation for elastic bodies using deformed distance fields. In *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, Vol. 1. IEEE, 330–336.

Stefan Gottschalk, Ming C Lin, and Dinesh Manocha. 1996. OBBTree: A hierarchical structure for rapid interference detection. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM, 171–180.

Gaël Guennebaud, Benoit Jacob, et al. 2010. Eigen. *URl: http://eigen. tuxfamily. org* 3 (2010).

Ernst Hairer and Gilles Vilmart. 2006. Preprocessed discrete Moser–Veselov algorithm for the full dynamics of a rigid body. *Journal of Physics A: Mathematical and General* 39, 42 (2006), 13225.

Shoichi Hasegawa, Nobuaki Fujii, Katsuhito Akahane, Yasuharu Koike, and Makoto Sato. 2004. Real-time rigid body simulation for haptic interactions based on contact volume of polygonal objects. *Transactions of the Society of Instrument and Control Engineers* 40, 2 (2004), 122–131.

Kris K Hauser, Chen Shen, and James F O'Brien. 2003. Interactive Deformation Using Modal Analysis with Constraints.. In *Graphics Interface*, Vol. 3. 16–17.

Philip Martyn Hubbard. 1995. Collision detection for interactive graphics applications. *IEEE Trans. on Visualization and Computer Graphics* 1, 3 (1995), 218–230.

Takeo Igarashi, Tomer Moscovich, and John F Hughes. 2005. As-rigid-as-possible shape manipulation. *ACM transactions on Graphics (TOG)* 24, 3 (2005), 1134–1141.

Doug L James and Dinesh K Pai. 2004. BD-tree: output-sensitive collision detection for reduced deformable models. *ACM Trans. Graph. (TOG)* 23, 3 (2004), 393–398.

Danny M Kaufman, Timothy Edmunds, and Dinesh K Pai. 2005. Fast frictional dynamics for rigid bodies. In *ACM SIGGRAPH 2005 Papers*. 946–956.

Danny M Kaufman, Shinjiro Sueda, Doug L James, and Dinesh K Pai. 2008. Staggered projections for frictional contact in multibody systems. In *ACM SIGGRAPH Asia 2008 papers*. 1–11.

Shankar Krishnan, M Gopi, M Lin, Dinesh Manocha, and A Pattekar. 1998. Rapid and accurate contact determination between spline models using ShellTrees. *Computer Graphics Forum* 17, 3 (1998), 315–326.

Lei Lan, Yin Yang, Danny Kaufman, Junfeng Yao, Minchen Li, and Chenfanfu Jiang. 2021. Medial IPC: accelerated incremental potential contact with medial elastics. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–16.

Minchen Li. 2020. *Robust and Accurate Simulation of Elastodynamics and Contact*. Ph.D. Dissertation. University of Pennsylvania.

Minchen Li, Zachary Ferguson, Teseo Schneider, Timothy Langlois, Denis Zorin, Daniele Panozzo, Chenfanfu Jiang, and Danny M Kaufman. 2020. Incremental potential contact: Intersection-and inversion-free, large-deformation dynamics. *ACM transactions on graphics* (2020).

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

Xuan Li, Yu Fang, Minchen Li, and Chenfanfu Jiang. 2021a. BFEMP: Interpenetration-free MPM–FEM coupling with barrier contact. *Computer Methods in Applied Mechanics and Engineering* (2021), 114350.

Miles Macklin, Matthias Müller, and Nuttapong Chentanez. 2016. XPBD: position-based simulation of compliant constrained dynamics. In *Proceedings of the 9th International Conference on Motion in Games*. 49–54.

Khaled Mamou, E Lengyel, and AK Peters. 2016. Volumetric hierarchical approximate convex decomposition. In *Game Engine Gems 3*. AK Peters, 141–158.

Hammad Mazhar, Toby Heyn, Dan Negrut, and Alessandro Tasora. 2015. Using Nesterov's method to accelerate multibody dynamics with friction and contact. *ACM Transactions on Graphics (TOG)* 34, 3 (2015), 1–14.

Jean J Moreau. 1988. Unilateral contact and dry friction in finite freedom dynamics. In *Nonsmooth mechanics and Applications*. Springer, 1–82.

Guy M Morton. 1966. A computer oriented geodetic data base and a new technique in file sequencing. (1966).

Jürgen Moser and Alexander P Veselov. 1991. Discrete versions of some classical integrable systems and factorization of matrix polynomials. *Communications in Mathematical Physics* 139, 2 (1991), 217–243.

Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. 2007. Position based dynamics. *Journal of Visual Communication and Image Representation* 18, 2 (2007), 109–118.

Matthias Müller, Miles Macklin, Nuttapong Chentanez, Stefan Jeschke, and Tae-Yong Kim. 2020. Detailed rigid body simulation with extended position based dynamics. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 101–112.

Developer Z Nvidia. 2011. Physx sdk. *URL Httpdeveloper Nvidia Comphysx-Downloads* (2011).

Miguel A Otaduy, Daniel Germann, Stephane Redon, and Markus Gross. 2007. Adaptive deformations with fast tight bounds. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 181–190.

Mark Pauly, Dinesh K Pai, and Leonidas J Guibas. 2004. Quasi-rigid objects in contact. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 109–119.

Albert Peiret, Sheldon Andrews, József Kövecses, Paul G Kry, and Marek Teichmann. 2019. Schur complement-based substructuring of stiff multibody systems with contact. *ACM Transactions on Graphics (TOG)* 38, 5 (2019), 1–17.

Stephane Redon, Ming C Lin, Dinesh Manocha, and Young J Kim. 2005. Fast continuous collision detection for articulated models. (2005).

Peng Song and Vijay Kumar. 2003. Distributed compliant model for efficient dynamic simulation of systems with frictional contacts. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Vol. 37009. 1009–1018.

Olga Sorkine and Marc Alexa. 2007. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, Vol. 4. 109–116.

David E Stewart. 2000. Rigid-body dynamics with friction and impact. *SIAM review* 42, 1 (2000), 3–39.

Min Tang, Dinesh Manocha, Miguel A Otaduy, and Ruofeng Tong. 2012. Continuous penalty forces. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 1–9.

Min Tang, Ruofeng Tong, Zhendong Wang, and Dinesh Manocha. 2014. Fast and exact continuous collision detection with bernstein sign classification. *ACM Transactions on Graphics (TOG)* 33, 6 (2014), 1–8.

Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. 1987. Elastically deformable models. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*. 205–214.

Matthias Teschner, Stefan Kimmerle, Bruno Heidelberger, Gabriel Zachmann, Laks Raghupathi, Arnulph Fuhrmann, M-P Cani, François Faure, Nadia Magnenat-Thalmann, Wolfgang Strasser, et al. 2005. Collision detection for deformable objects. In *Computer graphics forum*, Vol. 24. Wiley Online Library, 61–81.

Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 5026–5033.

Jeffrey C Trinkle, JA Tzitzouris, and Jong-Shi Pang. 2001. Dynamic multi-rigid-body systems with concurrent distributed contacts. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 359, 1789 (2001), 2575–2593.

Bolun Wang, Zachary Ferguson, Teseo Schneider, Xin Jiang, Marco Attene, and Daniele Panozzo. 2021. A Large-scale Benchmark and an Inclusion-based Algorithm for Continuous Collision Detection. *ACM Transactions on Graphics (TOG)* 40, 5 (2021), 1–16.

Endong Wang, Qing Zhang, Bo Shen, Guangyong Zhang, Xiaowei Lu, Qing Wu, and Yajuan Wang. 2014. Intel math kernel library. In *High-Performance Computing on the Intel® Xeon Phi™*. Springer, 167–188.

Hongyi Xu, Yili Zhao, and Jernej Barbič. 2014. Implicit multibody penalty-baseddistributed contact. *IEEE transactions on visualization and computer graphics* 20, 9 (2014), 1266–1279.

Gabriel Zachmann. 2002. Minimal hierarchical collision detection. In *ACM symposium on Virtual reality software and technology*. ACM, 121–128.

Gabriel Zachmann and Elmar Langetepe. 2003. *Geometric data structures for computer graphics*. Eurographics Assoc.

Yidong Zhao, Jinhyun Choo, Yupeng Jiang, Minchen Li, Chenfanfu Jiang, and Kenichi Soga. 2022. A barrier method for frictional contact on embedded interfaces. *Computer Methods in Applied Mechanics and Engineering* 393 (2022), 114820.