

Hierarchical Optimization Time Integration for CFL-Rate MPM Stepping

XINLEI WANG, Zhejiang University & University of Pennsylvania

MINCHEN LI, University of Pennsylvania & Adobe Research

YU FANG, University of Pennsylvania

XINXIN ZHANG, Tencent

MING GAO, Tencent & University of Pennsylvania

MIN TANG, Zhejiang University

DANNY M. KAUFMAN, Adobe Research

CHENFANFU JIANG, University of Pennsylvania

We propose Hierarchical Optimization Time Integration (HOT) for efficient implicit timestepping of the material point method (MPM) irrespective of simulated materials and conditions. HOT is an MPM-specialized hierarchical optimization algorithm that solves nonlinear timestep problems for large-scale MPM systems near the CFL limit. HOT provides convergent simulations out of the box across widely varying materials and computational resolutions without parameter tuning. As an implicit MPM timestepper accelerated by a custom-designed Galerkin multigrid wrapped in a quasi-Newton solver, HOT is both highly parallelizable and robustly convergent. As we show in our analysis, HOT maintains consistent and efficient performance even as we grow stiffness, increase deformation, and vary materials over a wide range of finite strain, elastodynamic, and plastic examples. Through careful benchmark ablation studies, we compare the effectiveness of HOT against seemingly plausible alternative combinations of MPM with standard multigrid and other Newton-Krylov models. We show how these alternative designs result in severe issues and poor performance. In contrast, HOT outperforms existing state-of-the-art, heavily optimized implicit MPM codes with an up to 10× performance speedup across a wide range of challenging benchmark test simulations.

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

Additional Key Words and Phrases: Material point method (MPM), optimization integrator, quasi-Newton, multigrid

X. Wang and M. Li contributed equally to this article.

This work was supported in part by NSFC (61972341, 61972342, 61732015, 61572423), NSF grants IIS-1755544 and CCF-1813624, DOE ORNL subcontract 4000171342, a gift from Adobe Inc., NVIDIA GPU grants, and Houdini licenses from SideFX.

Authors' addresses: X. Wang, M. Li, Y. Fang, M. Gao, and C. Jiang, Zhejiang University and University of Pennsylvania, 3330 Walnut Street Philadelphia, PA, USA 19104; emails: wxlwxl1993@zju.edu.cn, {minchenli, squarefk, ming.gao07}@gmail.com, cf-jiang@seas.upenn.edu; X. Zhang, Tencent, Keji Middle 1st Rd, Nanshan, Shenzhen, Guangdong Province, China, 518057; email: zhangshinshin@gmail.com; M. Tang, College of Computer Science and Technology, Zhejiang University, Hangzhou, Zhejiang, P. R. China, 310027; email: tang_m@zju.edu.cn; D. M. Kaufman, Adobe Research, 801 N 34th St, Seattle, WA, USA 98103; email: kaufman@adobe.com.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](https://permissions.acm.org).

© 2020 Association for Computing Machinery.

0730-0301/2020/04-ART21 \$15.00

<https://doi.org/10.1145/3386760>

ACM Reference format:

Xinlei Wang, Minchen Li, Yu Fang, Xinxin Zhang, Ming Gao, Min Tang, Danny M. Kaufman, and Chenfanfu Jiang. 2020. Hierarchical Optimization Time Integration for CFL-Rate MPM Stepping. *ACM Trans. Graph.* 39, 3, Article 21 (April 2020), 16 pages.

<https://doi.org/10.1145/3386760>

1 INTRODUCTION

The material point method (MPM) is a versatile and highly effective approach for simulating widely varying material behaviors ranging from stiff elastodynamics to viscous flows (e.g., see Figures 12 and 14) in a common framework. As such, MPM offers the promise of a single unified, consistent, and predictive solver for simulating continuum dynamics across diverse and potentially heterogeneous materials. However, to reach this promise, significant hurdles remain. Most significantly, obtaining accurate, consistent, and robust solutions within a practical time budget is severely challenged by small timestep restrictions. This is most evidenced as we vary material properties, amounts of deformation, and/or simulate heterogeneous systems (Table 1).

Although MPM's Eulerian grid resolution limits timestep sizes to the CFL limit¹ [Fang et al. 2018], the explicit time integration methods commonly employed for MPM often require much smaller timesteps. In particular, the stable timestep sizes of explicit MPM time integration methods remain several orders of magnitude below the CFL limit when simulating stiff materials like metal (see Table 1) and snow [Fang et al. 2018; Stomakhin et al. 2013]. A natural solution then is to apply implicit numerical time integration methods, such as implicit Euler, which can enable larger stable timestep sizes for MPM [Fang et al. 2019; Gast et al. 2015]. However, doing so requires solving challenging and potentially expensive nonlinear systems at every timestep.

1.1 Challenges to Implicit MPM Timestepping

Although implicit MPM timestepping methods in engineering provide larger step sizes [Cummins and Brackbill 2002; Guilkey and Weiss 2001, 2003; Nair and Roy 2012], they do not target CFL-rate timestep sizes as is typically desired in graphics applications. Thus,

¹A particle cannot travel more than one grid cell per timestep, although in practice, a CFL number of 0.6 is often used [Gast et al. 2015].

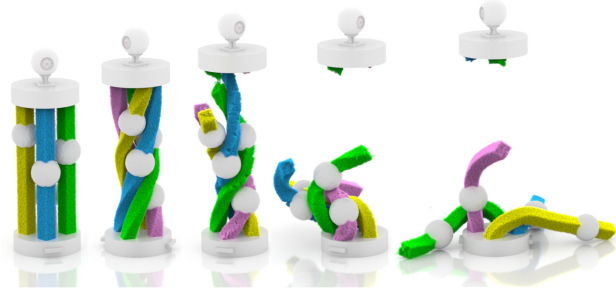


Fig. 1. Hierarchical Optimization Time Integration (HOT) is naturally suited for simulating dynamic contact of heterogeneous solid materials with substantial stiffness discrepancy. In this bar twisting example, compared across all available state-of-the-art, heavily optimized implicit MPM codes, HOT achieves more than $4\times$ speedup overall and up to $10\times$ per-frame. HOT obtains rapid convergence without need for per-example hand tuning of either outer nonlinear solver nor inner linear solver parameters.

in engineering, the standard Newton method is often directly applied without globalizations to solve the nonlinear timestepping problem [Charlton et al. 2017; Nair and Roy 2012]; with larger timestep sizes near the CFL limit, the nonlinearity grows and the first-order Taylor expansion becomes less accurate, which can then make the Newton method unstable and even explode. More recently, state-of-the-art implicit MPM methods in graphics have been introduced that enable timesteps closer to the CFL limit. Gast et al. [2015] introduced a globalized Newton-Krylov method for MPM, whereas Fang et al. [2019] extended ADMM to solve implicit MPM timesteps. However, their convergence and performance are limited for simulations involving heterogeneous and/or stiff materials, leading to slow computations and inconsistent, unpredictable, and even unstable results. Although ADMM [Fang et al. 2019] for MPM is attractively efficient, the underlying ADMM algorithm has no guarantee of convergence for nonconvex and nonlinear continua problems. In practice, it can at best achieve linear convergence. As we show in Section 7, when able to converge, the ADMM solver is thus exceedingly slow to reach reasonable solutions.

However, inexact Newton-Krylov methods exemplified by Gast et al. [2015] are seemingly ideal for solving implicit MPM problems where the sparsity structure of the Hessian can change at every timestep. Key to the efficiency and stability of these methods are the *inexact* iterative linear solve of each inner Newton iterate. In turn, this requires setting a tolerance to terminate each such inner loop. However, no single tolerance setting works across examples. Instead, suitable tolerances can and will vary over many orders of magnitude per example and so must be experimentally determined as we change setups over many expensive, successive simulation trials. Otherwise, as we demonstrate in Section 7 and in our supplemental material, a tolerance suitable for one simulated scene will generate extremely slow solves, nonphysical artifacts, instabilities, and even explosions in other simulations.

Next, we observe that Newton-Krylov methods employing Jacobi or Gauss-Seidel preconditioned CG solvers significantly lose efficiency from deteriorating convergence as material stiffnesses increase (Table 3). In such cases, multigrid preconditioners

Table 1. Parameters for Solid Materials Studied in This Article

	Density (kg/m^3)	Young's Modulus (Pa)	Poisson's Ratio	Yield Stress (Pa)
Tissue	300–1,000	$1 \times 10^2 - 1 \times 10^6$	0.4–0.5	—
Rubber	1,000–2,500	$1 \times 10^6 - 5 \times 10^7$	0.3–0.5	—
Bone	800–2,000	$7 \times 10^7 - 3 \times 10^{10}$	0.1–0.4	—
PVC	1,000–2,000	$2 \times 10^9 - 4 \times 10^9$	0.3–0.4	$1 \times 10^7 - 4 \times 10^7$
Metal	5,00–20,000	$1 \times 10^{10} - 4 \times 10^{11}$	0.2–0.4	$2 \times 10^8 - 2 \times 10^9$
Ceramic	2,000–6,000	$1 \times 10^{11} - 4 \times 10^{11}$	0.2–0.4	—

[Tamstorf et al. 2015; Wang et al. 2018; Zhu et al. 2010] are often effective solutions, as the underlying hierarchy allows aggregation of multiple approximations of the system matrix inverse across a range of resolutions. This accelerates information propagation across the simulation domain and thereby improves convergence.

We focus on *h*-multigrid that coarsens the degrees of freedom for coarser levels to reduce the computational cost. *H*-multigrid has been investigated for MPM by Cummings and Brackbill [2002] via merging particles level by level. However, they conclude that it performs similarly to Jacobi preconditioners, which indicates that building a multigrid hierarchy for MPM is challenging. This may be because merging particles lacks error bounds and can potentially make DOF coarsening inconsistent. However, as we discuss in Section 7.3, although building each coarser-level system using the original particles without merging improves convergence of inner linear solves, the computational overhead of this seemingly reasonable hierarchy still does not reduce the overall cost of MPM simulations. This is because (1) construction and evaluation of system matrices in each coarser level can be as expensive as the fine-level computation, and (2) at the domain boundaries, the coarsening of DOFs may not be consistently defined between matrices and right-hand side vectors of the coarse-level systems.

1.2 Hierarchical Optimization Time Integration

We propose the Hierarchical Optimization Time Integration (HOT) algorithm to address these existing limitations and so provide “out-of-the-box” efficient MPM simulation. To enable consistent, automatic termination of both outer Newton iterations and inner inexact linear solves across simulation types, we extend the characteristic norm (CN) [Li et al. 2019; Zhu et al. 2018] to inhomogeneous MPM materials. As we show in Section 5.2 and Table 1 in our supplemental material, this produces consistent, automatic, high-quality results for inexact Newton-Krylov simulations that match the quality and timing of the best hand-tuned results of Gast et al. [2015].

Next, to obtain both improved convergence and performance for MPM systems with multigrid, we develop a new, MPM-customized hierarchy. We begin by embedding progressively finer-level grid nodes into coarser-level nodes via the MPM kernel. We then construct coarse-level matrices directly from their immediate finer-level matrix entries. This avoids computation and storage of each coarse level’s geometric information automatically handles boundary conditions and enables sparsity by our choice of MPM embedding kernel. This resulting multigrid hierarchy then retains improved convergence while significantly improving performance (see Figure 19).

While offering a significant gain, our MPM-customized multigrid still requires explicit matrix construction. In many elastodynamic simulation codes, such matrix construction costs are alleviated by applying just a fixed number of Newton iterations irrespective of convergence. However, this strategy is neither suitable for artistic control nor engineering, as it sacrifices consistency and accuracy for efficiency—for example, it can produce artificially softened materials, numerically damped dynamics, and inaccurate predictions. Following recent developments in mesh-based elasticity methods [Li et al. 2019], we instead alleviate matrix construction costs by constructing our hierarchy just once per timestep but then apply it as an efficient, second-order initializer (with one V-cycle per iteration) inside a quasi-Newton solver.

1.3 Contributions

HOT's inner multigrid provides efficient second-order information, whereas its outer quasi-Newton low-rank updates provide efficient curvature updates. This enables HOT to maintain consistent, robust output with a significant speedup in performance—even as we grow stiffness, increase deformation, and widely vary materials across the simulation domain. The combined application of node-embedding multigrid, automatic termination, and customized integration of multigrid V-cycle into the quasi-Newton loop jointly enables HOT's significant and consistent performance gains. In summary, our contributions are the following:

- We derive a novel MPM-specific multigrid model exploiting the regularity of the background grid and construct a Galerkin coarsening operator consistent with rediscritization via particle quadrature. To our knowledge, this is the first time Galerkin h -multigrid is applied for the MPM discretization of nonlinear elasticity with significant performance gain.
- We develop a new, node-wise CN [Li et al. 2019; Zhu et al. 2018] measure for MPM. Node-wise CN enables unified tolerancing across varying simulation resolutions, material parameters, and heterogeneous systems for both termination of inner solves in inexact Newton and convergence determination across methods. CN likewise ensures a fair comparison across all solvers in our experiments.
- We construct HOT—an *out-of-the-box* implicit MPM time integrator—by employing our multigrid as an efficient inner initializer inside a performant quasi-Newton MPM timestep solve. A carefully designed set of algorithmic choices customized for MPM then achieves both efficiency and accuracy that we demonstrate on a diverse range of numerically challenging simulations.
- We perform and analyze extensive benchmark studies on challenging, industrial-scale simulations to determine the best data structure and algorithmic choices for MPM numerical time integration. Across these simulation examples, we compare HOT against a wide range of alternative, seemingly reasonable algorithmic choices to demonstrate their pitfalls and the carefully designed advantages of HOT.

Across a wide range of challenging elastodynamic and plastic test simulations, we show (see Section 7) that HOT *without the need of any parameter tuning* outperforms existing state-of-the-art,

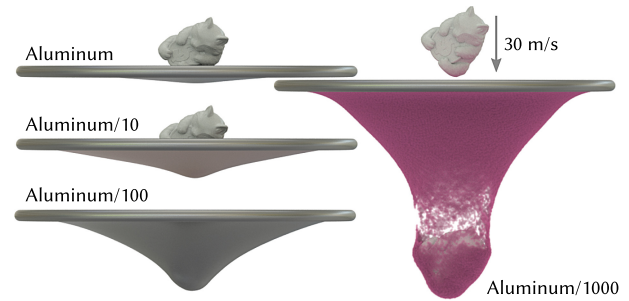


Fig. 2. Stiffness comparisons. A stiff lucky cat is smashed onto sheets with different Young's moduli starting from aluminum (6.9 GPa, with yield stress 240 Mpa) and then scaled down by 10, 100, and 1,000. Different stiffness gives drastically different behavior for elastoplastic materials.

heavily optimized implicit MPM codes. All alternative methods either exhibit significantly slower performance or else suffer from large variations across simulated examples. Our study then suggests HOT as a robust, unified MPM time integrator with fast convergence and outstanding efficiency of up to 10 $\times$ speedup to best alternatives across a wide range of possible simulation input.

2 RELATED WORK

2.1 Material Point Method

MPM was introduced by Sulsky et al. [1994] as a generalization of FLIP [Brackbill et al. 1988; Zhu and Bridson 2005] to solid mechanics. MPM's convergence was demonstrated computationally and explained theoretically by Steffen et al. [2008] with a smooth (e.g., quadratic B-spline) basis for grid solutions. This was further verified by Wallstedt [2009] with manufactured solutions.

In graphics, MPM has been applied to model a diverse array of materials and behaviors. These include the modeling of snow [Stomakhin et al. 2013], sand [Daviet and Bertails-Descoubes 2016; Gao et al. 2018b; Klár et al. 2016; Yue et al. 2018], foam [Fang et al. 2019; Ram et al. 2015; Yue et al. 2015], cloth [Guo et al. 2018; Jiang et al. 2017a], rods [Fei et al. 2019; Han et al. 2019], mixtures [Nagasawa et al. 2019], fracture [Wang et al. 2019; Wolper et al. 2019; Wretborn et al. 2017], multiphase flow [Gao et al. 2018a; Pradhana et al. 2017; Stomakhin et al. 2014], and even baking [Ding et al. 2019]. The coupling between softer MPM materials and rigid bodies has also been explored in both explicit [Hu et al. 2018] and implicit [Ding and Schroeder 2019] settings. Although rigid body dynamics provides an efficient approximation of extremely stiff materials for many applications, it is not suitable for capturing elastoplastic yielding nor for computing accurate mechanical responses.

Implicit time integration, such as via implicit Euler, is often the preferred choice for timestepping stiff materials and large deformations due to explicit integration's often unacceptable sound-speed CFL restriction [Fang et al. 2018]. Early implicit MPM [Guilkey and Weiss 2001, 2003] solutions applied Newmark time integration, demonstrating improved stability and solution accuracy over explicit MPM when compared to validated finite element solutions. More recently, Nair and Roy [2012] and Charlton et al. [2017] further investigated implicit generalized interpolation MPM for hyperelasticity and elastoplasticity, respectively.



Fig. 3. ArmaCat. A soft armadillo and a stiff lucky cat are both dropped onto an elastic trampoline, producing interesting interactions between them.

However, research in graphics has explored force linearization [Stomakhin et al. 2013] and optimization-stabilized Newton-Raphson solutions for both implicit Euler [Gast et al. 2015] and implicit midpoint [Jiang et al. 2017b] to achieve larger timestep sizes.

2.2 Optimization and Nonlinear Integrators

Numerical integration of differential systems is often reformulated variationally to be solved as a minimization problem. This allows methods to often achieve improved accuracy, robustness, and performance by taking advantage of available numerical optimization techniques. In computer graphics, simulation methods are increasingly applying this strategy to simulate both fluid [Batty et al. 2007; Weiler et al. 2016] and solid [Bouaziz et al. 2014; Dinev et al. 2018a, 2018b; Gast et al. 2015; Overby et al. 2017; Wang and Yang 2016] dynamics. For optimizations originating from nonlinear problems, Newton-type methods are generally the standard mechanism, delivering quadratic convergence near solutions. However, when the initial guess is far from a solution, Newton’s method may fail to provide a reasonable search direction, as the Hessian can be indefinite [Li et al. 2019; Liu et al. 2017; Smith et al. 2018]. Teran et al. [2005] propose a positive definite fix to project the Hessian to a symmetric positive definite form to guarantee a descent direction can be found; we compare with this method and further augment it with a backtracking line search to ensure energy decrease. We refer to this method as projected Newton (PN). Since PN requires a potential energy to be defined for the timestepping, we lag the plasticity update and only perform it once per timestep so that our system stays integrable. This is unlike Charlton et al. [2017], Klár et al. [2016], and Fang et al. [2019], who handle plasticity fully implicitly.

In each PN iteration, a linear system is solved. For MPM simulations that generally involve a large number of nodes and can have changing sparsity, Krylov iterative linear solvers such as conjugate gradient (CG) are generally preferred over direct factorization. To improve CG convergence, different preconditioning options exist. We apply the most efficient and straightforward Jacobi (diagonal) preconditioner as our baseline PN method, which we refer as PN-PCG. To further minimize memory consumption and access cost, existing implicit MPM methods in graphics apply matrix-free PN-PCG (PN-PCG(MF)) without explicitly constructing system matrices. However, when many CG iterations are required, such as for large timestep sizes and/or stiff materials, matrix free is no longer

necessarily a better option than matrix construction (Section 7). This is because the cost of recomputing the intermediate variables becomes more dominant, although such a cost could be significantly reduced if the matrix is explicitly constructed only once at the beginning of the timestep. Convergence can then be further improved with multigrid preconditioning. However, doing so for MPM presents new challenges.

2.3 Multigrid Methods

Multigrid methods [Briggs et al. 2000] are widely employed to accelerate both solid [McAdams et al. 2011; Tamstorf et al. 2015; Tielen et al. 2019; Wang et al. 2018; Xian et al. 2019; Zhu et al. 2010] and fluid [Aanjaneya et al. 2017; Fidkowski et al. 2005; Gao et al. 2018a; McAdams et al. 2010; Setaluri et al. 2014; Zhang and Bridson 2014; Zhang et al. 2015, 2016] dynamics simulations. Here, multi-level structures allow information of computational cells to better propagate, making multigrid methods highly efficient for systems with long-range energy responses and/or high stiffnesses. Unlike p -multigrid [Fidkowski et al. 2005; Tielen et al. 2019] methods that apply higher-order shape functions with same DOFs to improve convergence, h -multigrid methods construct hierarchies of coarser DOF models with potentially lower computational cost.

H -multigrid is generally categorized as geometric or algebraic. Unlike algebraic multigrid, geometric multigrid constructs coarse-level system matrices from coarsened grids or meshes [Stüben 2001]. However, the mismatch at the irregular boundaries due to geometric coarsening can require special treatment to ensure convergence improvement (e.g., extra smoothing at boundaries as in McAdam et al. [2010]). Alternately, Chentanez and Muller [2011] demonstrate that with a volume weighted discretization, robust results can be obtained without additional smoothing at boundaries. However, Ando et al. [2015] derive a multiresolution pressure solver from a variational framework that handles boundaries using signed-distance functions. Cummings and Brackbill [2002] propose a geometric multigrid preconditioned Newton-Krylov implicit MPM method that resamples particles for coarser levels. However, they conclude that such multigrid preconditioning performs similarly to Jacobi preconditioning; this is consistent with our analysis of geometric multigrid in Section 7.

However, Galerkin multigrid [Strang and Aarikka 1986] automatically handles boundary conditions by projection. Yet smooth projection matrices often deteriorate sparsity with large increases in the nonzero entries in coarse-level systems. Xian et al. [2019] designed their special Galerkin projection criterion based on skinning space coordinates with piecewise constant weights to maintain sparsity, but their projection could potentially lead to singular coarser-level matrices, and thus extra care needs to be taken. In our work, we derive prolongation and restriction operators via node embedding. Our resulting model is then consistent with an MPM-customized Galerkin multigrid while, due to the regularity of the MPM grid, our resulting coarse-level matrices both maintain sparsity via an appropriate choice of kernel and are full rank.

As in Ferstl et al. [2014], McAdams et al. [2011], and Zhang et al. [2016], a natural approach would then be to apply our multigrid as a preconditioner in a Krylov solver. However, as demonstrated in our benchmark experiments, this straightforward application

would not outperform existing diagonally preconditioned alternatives (PN-PCG) because of the repeated expense of hierarchy reconstruction at each Newton iterate. Instead, we develop HOT by applying our multigrid model as an efficient inner initializer within a quasi-Newton solver.

2.4 Quasi-Newton Methods

Quasi-Newton methods such as L-BFGS have long been applied for simulating elastica [Deuffhard 2011]. L-BFGS can be highly effective for minimizing potentials. However, an especially good choice of initializer is required and makes an enormous difference in convergence and efficiency [Nocedal and Wright 2006]. Directly applying a lagged Hessian at the beginning of each timestep is of course the most straightforward option that effectively introduces second-order information [Brown and Brune 2013]; unfortunately, it is generally a too costly option with limitations in terms of scalability. Liu et al. [2017] proposed to instead invert the Laplacian matrix that approximates the rest-shape Hessian as the initializer. This provides better scalability and more efficient evaluations, but convergence speed drops quickly in nonuniform deformation cases [Li et al. 2019]. Most recently Li et al. [2019] proposed a highly efficient domain-decomposed initializer for mesh-based FE that leverage start of timestep Hessians—providing both scalability and fast convergence in challenging elastodynamic simulations. For the MPM setting, inexact rather than direct methods are required to approximate the system Hessian given the scale and changing sparsity patterns of MPM simulations. Following Li et al. [2019], HOT applies our new multigrid as an inner initializer for L-BFGS to build an efficient method that outperforms or closely matches best-per-example prior methods across all tested cases on state-of-the-art, heavily optimized implicit MPM codes. Unlike in the work of Wen and Goldfarb [2009] that requires many configuration parameters to alternate between multigrid and single-level solves and uses L-BFGS as the solver for certain multigrid levels, HOT consistently applies V-cycles on our node-embedding multigrid constructed from the projected Hessian [Teran et al. 2005] as the inner initializer, without the need of any parameter tuning.

3 PROBLEM STATEMENT AND PRELIMINARIES

3.1 Optimization-Based Implicit MPM

MPM assembles a hybrid Lagrangian-Eulerian discretization of dynamics. A background Cartesian grid acts as the computational mesh while material states are tracked on particles. In the following, we apply subscripts p, q for particles and i, j, k for grid quantities, respectively. We then remove subscripts entirely, as in ζ , to denote vectors constructed by concatenating nodal quantities ζ_i over all grid nodes. Superscripts n , and $n + 1$ distinguish quantities at timesteps t^n and t^{n+1} . An implicit MPM timestep with implicit Euler from t^n to t^{n+1} is performed by applying the following operation sequence:

Particles-to-grid (P2G) projection: Particle masses m_p^n and velocities $\mathbf{v}_p^n$ are transferred to the grid’s nodal masses m_i^n and velocities $\mathbf{v}_i^n$ by APIC [Jiang et al. 2015].

Grid time step: Nodal velocity increments, $\Delta \mathbf{v}_i$, are computed by minimizing implicit Euler’s incremental potential in (1)

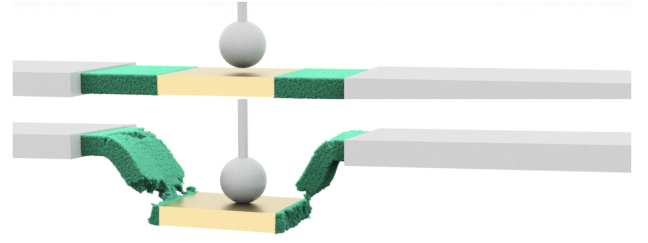


Fig. 4. Boxes. A metal box is concatenated with two elastic boxes on both sides. As the sphere keeps pushing the metal box downward, the elastic boxes end up being torn apart.

and are then applied to update nodal velocities by $\mathbf{v}_i^{n+1} = \mathbf{v}_i^n + \Delta \mathbf{v}_i$.

Grid-to-particles (G2P) interpolation: Particle velocities $\mathbf{v}_p^{n+1}$ are interpolated from $\mathbf{v}_i^{n+1}$ by APIC.

Particle strain-stress update: Particle strains (e.g., deformation gradients $\mathbf{F}_p$) are updated by the velocity gradient $\nabla \mathbf{v}$ via the updated Lagrangian. Where appropriate, inelasticity is likewise enforced through per-particle strain modification [Gao et al. 2017; Stomakhin et al. 2013].

Particle advection: Particle positions are advected by $\mathbf{v}_p^{n+1}$.

Here we focus on developing an efficient and robust nonlinear solver for the preceding MPM Grid timestep operation. All other operations are standard for MPM (see Jiang et al. [2016]).

Assuming an MPM nodal-position-dependent potential energy $\Phi(\mathbf{x})$ (e.g., a hyperelastic energy), Gast et al. [2015] observe that minimization of

$$E(\Delta \mathbf{v}) = \sum_i \frac{1}{2} m_i^n \|\Delta \mathbf{v}_i\|^2 + \Phi(\mathbf{x}^n + \Delta t(\mathbf{v}^n + \Delta \mathbf{v})) \quad (1)$$

subject to proper boundary conditions is equivalent to solving the MPM implicit Euler update $\mathbf{f}_i(\mathbf{x}_i^n + \Delta t \mathbf{v}_i^{n+1}) = (\mathbf{v}_i^{n+1} - \mathbf{v}_i^n) m_i^n / \Delta t$, where $\mathbf{f}_i$ is the implicit nodal force. Minimization of a corresponding incremental potential for the mesh-based elasticity has been widely explored for stable implicit Euler timestepping [Bouaziz et al. 2014; Li et al. 2019; Liu et al. 2017; Overby et al. 2017]. For MPM, however, a critical difference is that nodal positions $\mathbf{x}_i$ are virtually displaced from the Eulerian grid during the implicit solve and are then reset to an empty Cartesian scratchpad. Significantly, across timesteps, the system matrix can change sparsity pattern. This changing sparsity, together with large MPM system sizes (where more than 100K DOFs are common), generally motivate the application of matrix-free Newton-Krylov methods rather than direct factorization in existing MPM codes.

3.2 Inexact Newton-Krylov Methods

To minimize (1) with Newton-Krylov methods, further computational savings can be achieved by employing inexact Newton where computational effort in early Newton iterations can be saved by inexact solving the linear system. For example, Gast et al. [2015] apply the L2 norm of the incremental potential’s gradient to adaptively terminate Krylov iterations. However, Gast et al. [2015] mainly target softer materials. However, more generally, materials often have large material stiffnesses, such as Young’s at

ALGORITHM 1: Inexact Newton-Krylov Method**Given:** E, ϵ **Output:** $\Delta \mathbf{v}^n$ **Initialize and Precompute:** $i \leftarrow 1, \Delta \mathbf{v}^1 \leftarrow \mathbf{0}$ $\mathbf{g}^1 \leftarrow \nabla E(\Delta \mathbf{v}^1)$ // E is defined in Equation (1)**while** scaledL2norm($\mathbf{g}^i$) $> \epsilon \sqrt{n_{\text{node}}}$ **do** // termination criteria (Section 5.2) $\mathbf{P}^i \leftarrow \text{projectHessian}(\nabla^2 E(\Delta \mathbf{v}^i))$ // [Teran et al. 2005] $k \leftarrow \min(0.5, \sqrt{\max(\sqrt{g_i^T \mathbf{P} \mathbf{g}_i}, \tau)})$ // adaptive inexactness (Section 5.3) $\mathbf{p}^i \leftarrow \text{ConjugateGradient}(\mathbf{P}^i, \mathbf{0}, -\mathbf{g}^i, k)$ // k as relative tolerance $\alpha \leftarrow \text{LineSearch}(\Delta \mathbf{v}^i, 1, \mathbf{p}^i, E)$ // backtracking line search $\Delta \mathbf{v}^{i+1} \leftarrow \Delta \mathbf{v}^i + \alpha \mathbf{p}^i$ $\mathbf{g}^{i+1} \leftarrow \nabla E(\Delta \mathbf{v}^{i+1})$ $i \leftarrow i + 1$ **end while** $\Delta \mathbf{v}^n \leftarrow \Delta \mathbf{v}^i$

10^9 for the metal wheel in Figure 12. It becomes even more challenging when materials with widely varying stiffnesses interact with each other. In these cases, the inexact Newton strategy in Gast et al. [2015] can simply fail to converge in practical time, such as in our experiments for the scenes in Figures 1 and 10.

This observation has motivated the question as to whether an early termination criterion for Newton-type iterations can be computed to obtain visually consistent and stable results across varying simulation inputs. Li et al. [2019] extend the CN from distortion optimization [Zhu et al. 2018] to elastodynamics and demonstrate its capability to obtain consistent, relative tolerance settings across a wide set of elastic simulation examples over a range of material moduli and mesh resolutions. However, for a scene with materials with drastically different stiffness parameters, the averaging L2 measure will not suffice to capture the multiscale incremental potential gradient in a balanced manner.

We thus propose an extended scaled CN to support multimaterial applications in MPM. Incremental potential gradients are nonuniformly scaled so that multiscale residuals can be effectively resolved. We apply this new CN to both terminate outer Newton iterations and improve the inexact Newton iterations in our baseline PN solver. See Algorithm 1 for our inexact Newton; details are in Section 5.

With extended CN and improved inexact Newton, iterative methods can still suffer from ill conditioning from stiff materials and so we require preconditioning. Unfortunately, incomplete Cholesky is not suitable, as elastodynamic system Hessians are not M-matrices [Kershaw 1978], leading us to multigrid strategies. However, multigrid construction costs may not be well compensated by the resulting convergence improvement with Newton-Krylov. We thus apply our custom MPM multigrid, next constructed in Section 4, as an inner initializer inside our quasi-Newton loop (see Section 5).

4 MPM MULTIGRID

We propose to construct our hierarchy by embedding finer-level grids into the coarser-level grids analogously to MPM's

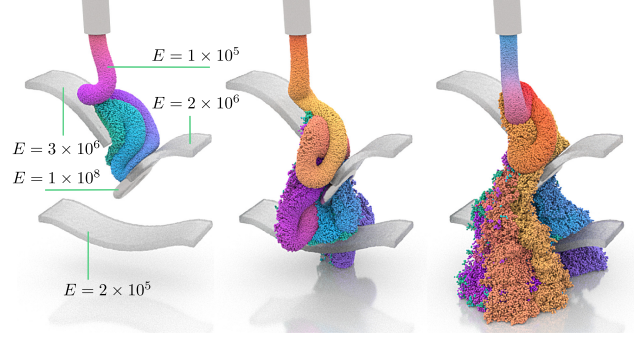


Fig. 5. Boards. A granular flow is dropped onto boards with varying Young's moduli, generating coupled dynamics.

embedding of particles into grid nodes. Then by explicitly storing the system matrix, we progressively constructing coarser-level matrices directly from the adjacent finer-level matrix entries, avoiding the need to compute or store any coarser-level geometric information. We next show that our multigrid is consistent with Galerkin multigrid where boundary conditions are automatically handled, and by selecting different node-embedding kernels, we support flexible control on sparsity.

4.1 Node-Embedding Multigrid Derivation

We begin with an M -level multigrid hierarchy. We denote level 0 and level $M - 1$ as the finest and coarsest levels, respectively. System matrices are constructed at each level with prolongation, $\mathcal{P}_m^{m+1}$, and restriction operators, $\mathcal{R}_m^{m+1}$, between adjacent levels m and $m + 1$.

We illustrate the construction of our restriction and prolongation operators by considering operations between levels 0 and 1. Nodal forces in the finest level are

$$\mathbf{f}_i^0 = - \sum_p V_p \frac{\partial \phi(\mathbf{x}_p)}{\partial \mathbf{x}_i^0} = - \sum_p V_p \mathbf{P}_p \mathbf{F}_p^T \nabla \omega_{ip}^0. \quad (2)$$

Here, V_p is the initial particle volume, ϕ is the energy density function, $\mathbf{P}_p$ is the first Piola-Kirchhoff stress, $\mathbf{F}_p$ is the deformation gradient, and ω_{ip} are corresponding particle-grid interaction weights.

In the multigrid hierarchy, residuals, following forces, are restricted from finer to coarser levels. Forces at nodes j in the next level are then $\mathbf{f}_j^1 = - \sum_p V_p \frac{\partial \phi(\mathbf{x}_p)}{\partial \mathbf{x}_j^1}$. Embedding finer-level nodes to coarser-level nodes, we then can simply apply the chain rule, converting derivatives evaluated at a coarse node to those already available at the finer level:

$$\mathbf{f}_j^1 = - \sum_i \sum_p V_p \left(\frac{\partial \mathbf{x}_i^0}{\partial \mathbf{x}_j^1} \right)^T \frac{\partial \phi(\mathbf{x}_p)}{\partial \mathbf{x}_i^0} = \sum_i \left(\frac{\partial \mathbf{x}_i^0}{\partial \mathbf{x}_j^1} \right)^T \mathbf{f}_i^0. \quad (3)$$

This gives our restriction operation as $\mathbf{f}^1 = \mathcal{R}_0^1 \mathbf{f}^0$ with $\mathcal{R}_0^1 = \left(\frac{\partial \mathbf{x}^0}{\partial \mathbf{x}^1} \right)^T$.

Prolongation is correspondingly given by the transpose $\mathcal{P}_0^1 = (\mathcal{R}_0^1)^T$. Recalling that MPM particle velocities $\mathbf{v}_p$ are interpolated

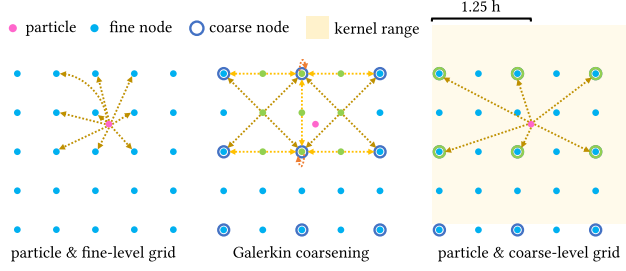


Fig. 6. Geometric equivalence. Left: In the finest level, particles' properties are transferred to the grid nodes via the B-spline quadratic weighting function. Middle: Then the finer nodes transfer information to coarser nodes via linear embedding relationships—based on which we perform Galerkin coarsening. Right: Galerkin coarsening can then be reinterpreted as a new weighting function, with a smaller kernel width, connecting coarser nodes directly with the particles.

from grid node velocities $\mathbf{v}_i$ as $\mathbf{v}_p = \sum_i w_{ip} \mathbf{v}_i$, we have

$$\mathbf{v}_j^0 = \sum_i \frac{\partial \mathbf{x}_i^0}{\partial \mathbf{x}_j^1} \mathbf{v}_i^1 = \sum_i (\mathcal{R}_0^1)_{ji}^T \mathbf{v}_i^1, \quad (4)$$

giving us $\mathbf{v}^0 = \mathcal{P}_0^1 \mathbf{v}^1 = (\mathcal{R}_0^1)^T \mathbf{v}^1$.

For matrix coarsening, we similarly can compute the second-order derivative of (1) w.r.t. $\mathbf{x}^1$. Applying the chain rule, with $\mathbf{x}^0$ as intermediate variable, we obtain

$$\begin{aligned} (\mathbf{H}^1)_{jk} &= \frac{\partial \mathbf{f}_j^1}{\partial \mathbf{x}_k^1} = \sum_l \frac{\partial \sum_i \left(\frac{\partial \mathbf{x}_i^0}{\partial \mathbf{x}_j^1} \right)^T \mathbf{f}_i^0}{\partial \mathbf{x}_l^0} \frac{\partial \mathbf{x}_l^0}{\partial \mathbf{x}_k^1} \\ &= \sum_i \sum_l \left(\frac{\partial \mathbf{x}_i^0}{\partial \mathbf{x}_j^1} \right)^T (\mathbf{H}^0)_{il} \frac{\partial \mathbf{x}_l^0}{\partial \mathbf{x}_k^1}. \end{aligned} \quad (5)$$

Here, $\mathbf{H}^0$ is the Hessian of (1) w.r.t. $\mathbf{x}^0$. We then have the Galerkin operator

$$\mathbf{H}^1 = \mathcal{R}_0^1 \mathbf{H}^0 \mathcal{P}_0^1, \quad (6)$$

confirming our construction is consistent with Galerkin multigrid. Dirichlet boundary conditions are then resolved at all levels by projection of the corresponding rows and columns of the system matrix and entries in the right-hand side.

4.2 Geometric Multigrid Perspective and Kernel Selection

Our multigrid is now complete up to our choice of embedding kernel, $\frac{\partial \mathbf{x}^0}{\partial \mathbf{x}^1}$, and MPM particle-grid kernel, ω_{ip} . A careful choice of kernels enables us to maintain sparsity as we coarsen. This allows us to improve both convergence and cost. We apply MPM kernels for our node embedding. Convolution with the particle-grid kernel can then be viewed as a direct embedding of our finest-level *particles* into coarser-level *grids*. This provides a geometric multigrid perspective where we can consider coarse grid matrices as constructed from fine particle quadratures. We next apply this perspective to select our multigrid kernels.

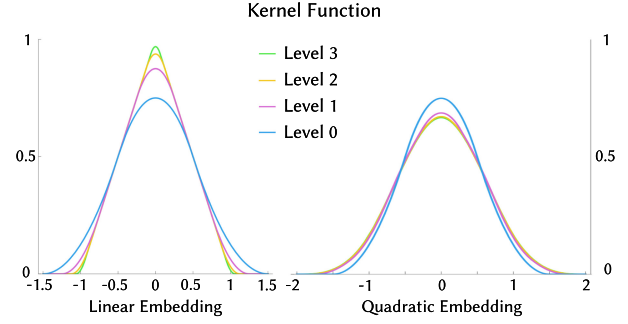


Fig. 7. Kernel width. The width of our geometric weighting function, equivalently in our algebraic derivation, changes with level increase. For linear embedding (our choice for HOT), width becomes smaller with coarsening, whereas for quadratic embedding, width becomes larger but with an upper bound at 2.

We start with a direct MPM derivation defining nodal forces at level 1:

$$\mathbf{f}_j^1 = - \sum_p V_p \mathbf{P}_p \mathbf{F}_p^T \nabla \omega_{jp}^1. \quad (7)$$

Compare this with a reformulation of (3) where we apply our newly defined restriction operator:

$$\mathbf{f}_j^1 = - \sum_p V_p \mathbf{P}_p \mathbf{F}_p^T \left(\sum_i (\mathcal{R}_0^1)_{ji} \nabla \omega_{ip}^0 \right). \quad (8)$$

Here, particle-grid weight gradients between level 1 and particles are now given by $\sum_i (\mathcal{R}_0^1)_{ji} \nabla \omega_{ip}^0$, and our multigrid obtains a simple geometric interpretation as illustrated in Figure 6. As a geometric multigrid, this provides a weighting function directly bridging between particles and coarse grid nodes. The grid itself can be generated by traversing all particles to find occupied coarse nodes. Similarly, a concatenation of prolongation operators for each coarse level, right multiplied by the original weight gradient, gives us the new weight gradients required in each successive level. In turn, with this weight gradient, the Hessian matrix can be defined to complete the geometric multigrid model. We use the corresponding weighting function to plot curves in Figure 7.

For HOT, we apply B-spline quadratic weighting for our base particle-grid kernel and choose the linear kernel for our embedding. The latter defines our prolongation and restriction operators between adjacent levels in the hierarchy. With this choice, the stencil size of our coarser-level systems become progressively smaller, providing better sparsity. As shown in Figure 7 (left), kernel width reduces from $3\Delta x$ to $2\Delta x$ as levels increase. An alternative would be to uniformly apply the B-spline *quadratic* weighting for all kernels. However, stencil size would then grow as we coarsen (c.f. Figure 7 (right)), making it computationally less attractive (see Table 2 in our supplemental material for the comparison). Likewise, direct geometric multigrid, where particles are directly coarsened, also exhibits impractical fill-in as stencil sizes grow with coarsening. For example, see Figure 8, where we compare the matrix sparsity patterns for the ArmaCat simulation in Figure 3.

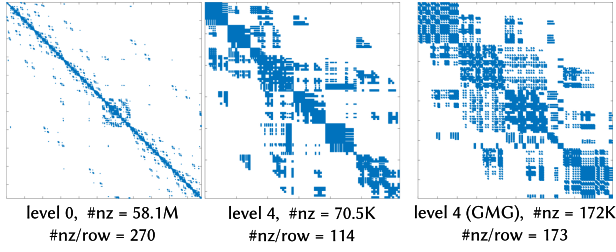


Fig. 8. Sparsity pattern. Our MPM multigrid system matrices gain better sparsity as levels increase because stencil sizes decrease. Left and middle: Our level-0 and level-4 matrices for the ArmaCat simulation in Figure 3. Right: Direct geometric multigrid generates denser matrices for increasingly coarse levels, here shown at level 4 for the same simulation. Note that although in simulation we only used a three-level multigrid, here we illustrate the sparsity patterns with five levels for visual clarity.

5 HIERARCHICAL OPTIMIZATION TIME INTEGRATION

Our newly constructed MPM multigrid structure can be used as a preconditioner by applying one V-cycle (Algorithm 2) per iteration for a CG solver to achieve superior convergence in a positive-definite fixed, inexact Newton method. In the following, we denote this approach as “projected Newton, multigrid preconditioned conjugate gradient,” or PN-MGPCG. However, in practice, the cost of reconstructing the multigrid hierarchy at each Newton iteration of PN-MGPCG is not well compensated by the convergence improvement, providing only little or moderate speedup compared to a baseline projected Newton PCG solver (PN-PCG) (see Figures 17 and 20) where a simple diagonal preconditioner is applied to CG. This is because in PN-MGPCG, each outer iteration (PN) requires reconstructing the multigrid matrices, and each inner iteration (CG) performs one V-cycle. One reconstruction of the multigrid matrices would take around 4× the time for one V-cycle and over 20× the time for one Jacobi preconditioned PCG iteration. Unlike the Poisson system in Eulerian fluids simulation, the stiffnesses of elastodynamic systems are often not predictable, as it varies a lot under different timestep sizes, deformation, and dynamics. Therefore, it is hard for PN-MGPCG to consistently well accelerate performance in all timesteps.

5.1 Multigrid Initialized Quasi-Newton Method

Rather than applying MPM multigrid as a *preconditioner* for a Krylov method (which can still both be slow and increasingly expensive as we grow stiffness; see Figure 20), inspired by Li et al. [2019], we apply our MPM multigrid as an *inner initializer* for a modified L-BFGS solver. In the resulting hierarchical method, multigrid then provides efficient second-order information, whereas our outer quasi-Newton low-rank updates [Li et al. 2019] provide efficient curvature updates to maintain consistent performance for timesteps with widely varying stiffness, deformations, and conditions. In turn, following recent developments, we choose a start of timestep lagged model update [Brown and Brune 2013; Li et al. 2019]. We reconstruct our multigrid structure once at the start of each timestep solve. This enables local second-order information to efficiently bootstrap curvature updates from the successive light-weight, low-rank quasi-Newton iterations.

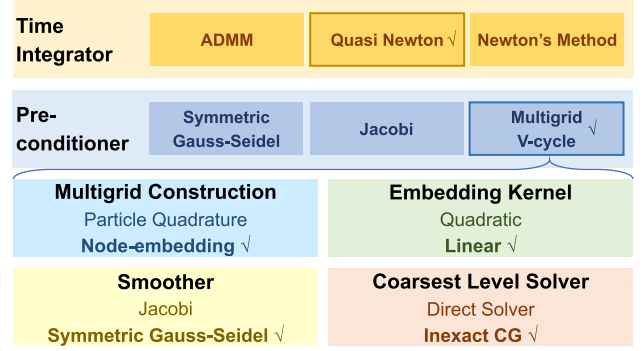


Fig. 9. Summary of HOT's design choices.

ALGORITHM 2: Multigrid V-Cycle Preconditioner

Given: $\mathcal{R}, \mathcal{P}, \mathcal{M}$

Input: b^0, H

Output: u^0

for $m = 0, 1, \dots, \mathcal{M} - 2$

$u^m \leftarrow 0$

$u^m \leftarrow \text{SymmetricGaussSeidel}(H^m, u^m, b^m)$

$b^{m+1} \leftarrow \mathcal{R}_m^{m+1}(b^m - H^m u^m)$

end for

$u^{\mathcal{M}-1} \leftarrow \text{ConjugateGradient}(H^{\mathcal{M}-1}, u^{\mathcal{M}-1}, b^{\mathcal{M}-1}, 0.5)$

for $m = \mathcal{M} - 2, \mathcal{M} - 3, \dots, 0$

$u^m \leftarrow u^m + \mathcal{P}_m^{m+1} u^{m+1}$

$u^m \leftarrow \text{SymmetricGaussSeidel}(H^m, u^m, b^m)$

end for

This completes the core specifications of our Hierarchical Optimization Time integrator algorithm. The HOT multigrid hierarchy is constructed at the beginning of each timestep. Then, for each L-BFGS iteration, the multiplication of our initial Hessian inverse approximation to the vector is applied by our multigrid V-cycle. To ensure the symmetric positive definiteness of the V-cycle operator, we apply colored symmetric Gauss-Seidel as the smoother for finer levels and employ Jacobi preconditioned CG solves for our coarsest-level system (see Algorithm 2). We apply PCG for our coarsest level rather than a direct solve, as the subtle convergence improvement overhead could not compensate for the overhead of factorization (see Section 7.2). Weighted Jacobi is effectively applied in Eulerian fluid simulation [Zhang et al. 2016] as a smoother for multigrid; however, here, in testing, we observe that determining proper weighting that obtains efficient or even convergent behavior for nondiagonally dominant elastodynamic Hessians is challenging. Similarly, we found Chebyshev smoothers [Adams et al. 2003] impractical, as estimating reasonable upper and lower eigenvalues of the system matrix introduces unacceptably large overhead.

HOT's curvature information is updated by low-rank secant updates with window size $w = 8$, producing a new descent direction for line search at each L-BFGS iteration. Pseudocode for the HOT method is presented in Algorithm 3. We analyze its performance, consistency, and robustness in Section 7.2 with comparisons to state-of-the-art MPM solvers. In Figure 9, we highlight design choices for HOT together with superficially reasonable alternatives that we compare and analyze in Section 7.

ALGORITHM 3: Hierarchical Optimization Time Integrator
Given: $E, \epsilon, w, \mathcal{R}, \mathcal{P}$
Output: $\Delta \mathbf{v}^n$
Initialize and Precompute:

```

 $i \leftarrow 1, \Delta \mathbf{v}^1 \leftarrow \mathbf{0}$ 
 $\mathbf{g}^1 \leftarrow \nabla E(\Delta \mathbf{v}^1)$  //  $E$  is defined in Equation (1)
 $\mathbf{P}^1 \leftarrow \text{projectHessian}(\nabla^2 E(\Delta \mathbf{v}^1))$  // [Teran et al. 2005]
 $\mathbf{H} \leftarrow \text{buildMultigrid}(\mathbf{P}^1, \mathcal{R}, \mathcal{P})$  // Equation (6)
    
```

 // Quasi-Newton loop to solve timestep $n + 1$:

```

while scaledL2norm( $\mathbf{g}^i$ ) >  $\epsilon \sqrt{n_{\text{node}}}$  do // termination criteria (Section 5.2)
    
```

```

         $\mathbf{q} \leftarrow -\mathbf{g}^i$ 
        // L-BFGS low-rank update
        for  $a = i - 1, i - 2, \dots, i - w$  // break if  $a < 1$ 
             $\mathbf{s}^a \leftarrow \Delta \mathbf{v}^{a+1} - \Delta \mathbf{v}^a, \mathbf{y}^a \leftarrow \mathbf{g}^{a+1} - \mathbf{g}^a, \rho^a \leftarrow 1 / ((\mathbf{y}^a)^T \mathbf{s}^a)$ 
             $\alpha^a \leftarrow \rho^a (\mathbf{s}^a)^T \mathbf{q}$ 
             $\mathbf{q} \leftarrow \mathbf{q} - \alpha^a \mathbf{y}^a$ 
        end for
         $\mathbf{r} \leftarrow \text{V-cycle}(\mathbf{q}, \mathbf{H})$  // Algorithm 2
        // L-BFGS low-rank update
        for  $a = i - w, i - w + 1, \dots, i - 1$  // skip (continue) until  $a \geq 1$ 
             $\beta^a \leftarrow \rho^a (\mathbf{y}^a)^T \mathbf{r}$ 
             $\mathbf{r} \leftarrow \mathbf{r} + (\alpha^a - \beta^a) \mathbf{s}^a$ 
        end for
         $\mathbf{p}^i \leftarrow \mathbf{r}$ 
         $\alpha \leftarrow \text{LineSearch}(\Delta \mathbf{v}^i, 1, \mathbf{p}^i, E)$  // backtracking line search
         $\Delta \mathbf{v}^{i+1} \leftarrow \Delta \mathbf{v}^i + \alpha \mathbf{p}^i$ 
         $\mathbf{g}^{i+1} \leftarrow \nabla E(\Delta \mathbf{v}^{i+1})$ 
         $i \leftarrow i + 1$ 
    
```

end while
 $\Delta \mathbf{v}^n \leftarrow \Delta \mathbf{v}^i$

5.2 Convergence Tolerance

To reliably obtain consistent results across heterogeneous simulations while saving computational effort, we extend the CN [Li et al. 2019; Zhu et al. 2018] from FE mesh to MPM discretization, taking multimaterial domains into account. To simulate multiple materials with significantly varying material properties, coupled in a single simulated system, the traditional *averaging* L2 measure fails to characterize the multiscale incremental potential's gradient.

For MPM, we thus first derive a node-wise CN in MPM discretization and then set tolerances with the L2 measure of the node-wise CN scaled incremental potential gradient. Concretely, we compute the norm of the stress derivative evaluated at the undeformed configuration in the diagonal space, $\xi_p = \|\frac{d\mathbf{P}}{d\mathbf{F}}\|_p$, for each particle p , and transfer this scalar field with mass weighting and normalization to corresponding grid node quantities ξ_i . Here, ξ_i is in units of J/m^3 as $\mathbf{P}$ is in the unit of energy density and $\mathbf{F}$ is unitless. We then compute a node-wise CN as

$$\ell_i \xi_i \Delta t, \quad (9)$$

per node where ℓ_i characterizes discretization, ξ_i characterizes averaged material stiffness per node, and Δt provides timestep scaling. In mesh-based FE, ℓ_i is the area of the polyhedron formed by the one-ring elements connecting to node i [Zhu et al. 2018]. For MPM, we correspondingly have $\ell_i = 24\Delta x^2$ from the uniform Cartesian grid discretization.

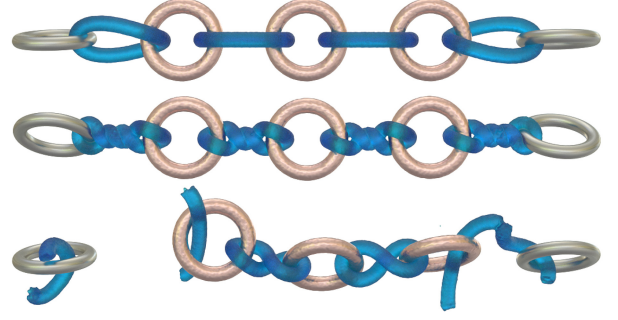


Fig. 10. Rotating chain. A chain of alternating soft and stiff rings is rotated until soft rings fracture, dynamically releasing the chain.

To check convergence, we scale each entry of the incremental potential gradient vector $\mathbf{g}$ (in units of $\text{kg} \cdot \text{m/s}$ as our optimization variable is velocity) with the corresponding node-wise CN computed in Equation (9), obtaining the unitless $\hat{\mathbf{g}}$. Termination queries then compare $\|\hat{\mathbf{g}}\|$ against $\epsilon \sqrt{n}$, where n gives the number of active grid nodes and ϵ is the selected accuracy tolerance. We confirm that when a single, uniform material is applied in the simulation, our extended CN measure correctly reduces to the L2 measure of Li et al. [2019].

5.3 Inexact Linear Solves

We apply the preceding extended CN criterion to terminate outer nonlinear solve iterates. Within each outer nonlinear iteration, inner iterations are performed to solve the corresponding linear system to a reasonable accuracy. For the first few Newton iterations of an inexact Newton-Krylov solve, the initial nonlinear residuals are generally far from convergence and so inexact linear solves are preferred. As discussed in Gast et al. [2015], inexact linear solves can significantly relieve the computational burden in large timestep simulations. Thus, they set a relative tolerance k on the energy norm $\sqrt{\mathbf{r}_0^T \mathbf{P} \mathbf{r}_0}$, where $\mathbf{P}$ is the preconditioning matrix, of the initial residual $\mathbf{r}_0 = -\nabla E(\Delta \mathbf{v}^i)$ for each linear solve in Newton iteration i . Gast et al. [2015] use $k = \min(0.5, \sqrt{\max(\|\nabla E(\Delta \mathbf{v}^i)\|, \tau)})$, where τ is a nonlinear tolerance, and perform CG iterations until the current $\sqrt{\mathbf{r}^T \mathbf{P} \mathbf{r}}$ is smaller than $k \sqrt{\mathbf{r}_0^T \mathbf{P} \mathbf{r}_0}$. This strategy can be traced back to classical optimization strategies where setting $k = \min(0.5, \sqrt{\|\nabla E(\Delta \mathbf{v}^i)\|})$ can be shown to yield superlinear convergence [Nocedal and Wright 2006].

However, this approach is challenged when it comes to heterogeneous materials in two aspects. First, the L2 norm of the incremental potential does not take into account its multiscale nature, potentially providing too small relative tolerances for stiff materials. Second, as discussed earlier, the nonlinear tolerance in Gast et al. [2015] is challenging to tune per example, especially with stiff material models. Therefore, we modify this inexact strategy for our baseline PN-PCG by applying $k = \min(0.5, \sqrt{\max(\sqrt{\mathbf{r}_0^T \mathbf{P} \mathbf{r}_0}, \tau)})$ as the relative tolerance to terminate CG iterations. Here, the preconditioning matrix $\mathbf{P}$ in the energy norm $\sqrt{\mathbf{r}_0^T \mathbf{P} \mathbf{r}_0}$ has the effect of locally scaling per-node

residuals to account for varying material stiffnesses, whereas τ is simply Li et al.’s [2019] tolerance on L2 measure characterizing the most stiff material in the running scene, ensuring that our tolerance will not be too small for stiffer materials.

HOT similarly exploits our inexact solving criterion for the coarsest-level PCG during early L-BFGS iterations. Specifically, in each V-cycle, we recursively restrict the right-hand side vector $\mathbf{b}_0$ to the coarsest level to obtain $\mathbf{b}_{m-1}$. We then set the tolerance for the CG solver to $1/2\sqrt{\mathbf{b}_{m-1}^T \mathbf{D}_{m-1}^{-1} \mathbf{b}_{m-1}}$, where $\mathbf{D}_{m-1}$ is the diagonal matrix extracted from the system matrix at level $m-1$. Note that the same V-cycle and termination criterion is also adopted in our PN-MGPCG. As L-BFGS iterations proceed, the norm of $\mathbf{b}_{m-1}$ decreases, leading to increasingly accurate solves at the coarsest level. As demonstrated in Section 7, this reduces computational effort, especially when the system matrices at the coarsest level are not well conditioned.

6 IMPLEMENTATION

Accompanying this article, we open source all of our code including scripts for running all presented examples with HOT and all other implemented methods compared with in test code. ADMM MPM [Fang et al. 2019] is separately available.² Here we provide remarks on the nontrivial implementation details that can significantly influence performance.

Lock-free multithreading. For all particle-to-grid transfer operations (including each matrix-vector multiplication in PN-PCG(MF)), we adopt the highly optimized lock-free multithreading from Fang et al. [2018]. This also enables the parallelization of our colored symmetric Gauss Seidel smoother for the multigrid V-cycle. All optimizations are thus consistently utilized (wherever applicable) across all compared methods so that our timing comparisons more reliably reflect algorithmic advantages of HOT.

Sparse matrix storage. We apply the quadratic B-spline weighting kernel for particle fine-grid transfers. The number of nonzero entries per row of the system matrix at the finest level can then be up to 5^d , where d denotes dimension. In more coarsened levels, the number of nonzero entries decreases due to the linear embedding of nodes in our MPM multigrid, as can be seen from Figure 7. Similarly, for our restriction/prolongation matrix, the number of nonzero entries per row/column is 3^d for linear kernel. Notice that in all cases, the maximum number of nonzeros per row can be predetermined, and thus we employ *diagonal storage format* in our implementation to store all three matrix types for accelerating matrix computations.

Multigrid application. In our experiments (Section 7), our MPM multigrid is tested both as a preconditioner for the CG solver in each PN-MGPCG outer iteration, and as the inner initializer for each L-BFGS iteration of HOT.

Prolongation and restriction. Our prolongation operator is defined as in traditional particle-grid transfers in hybrid methods—finer nodes are temporarily regarded as particles in the coarser

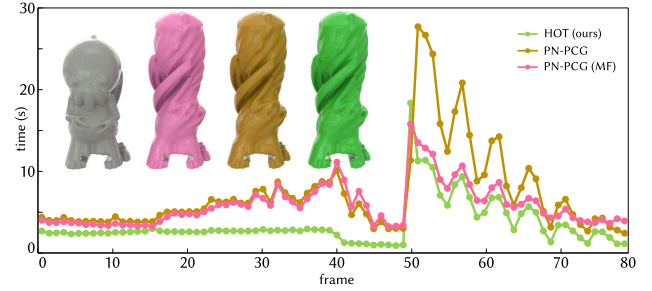


Fig. 11. Faceless. We rotate the cap and then release the head of the “faceless” mesh. Upon release, dynamic rotation and expansion follow. Here we plot the computation time of each single timestep frame for HOT and PN-PCG (both matrix based and matrix free). HOT outperforms across all timestep solves during the simulation.

level. Spatial hashing is then applied to record the embedding relation between finer- and coarser-grid nodes for efficiency.

7 RESULTS AND EVALUATION

7.1 Benchmark

Methods in comparison. We implement a common test-harness code to enable the consistent evaluation comparisons between HOT and other possible Newton-Krylov methods, such as PN-PCG, PN-PCG(MF), and state-of-the-art (MF) from Gast et al. [2015]. To ensure consistency, PN-PCG and PN-PCG(MF) adopt our node-wise CN with the same tolerances. For Gast et al. [2015] where a global tolerance for the residual norm is applied, we manually select the largest tolerance value (10^{-3}) that produces artifact-free results for all experiments. In addition to Gast et al. [2015], we also compare to the ADMM-based state-of-the-art implicit MPM method [Fang et al. 2019] on the faceless example to demonstrate differences in the order of convergence (Figure 16). We note that other than Gast et al. [2015] and Fang et al. [2019], all other methods in our study are applied here, to the best of our knowledge, for the first time for MPM.

We continue our ablation study here on how our design choices for HOT impact performance and convergence. We compare HOT with other potential new MPM solvers that one may consider designing, including (1) HOT-quadratic: HOT’s framework with the quadratic (rather than linear) embedding kernel; (2) LBFGS-GMG: L-BFGS with a more standard geometric multigrid as the initializer; (3) PN-MGPCG: a Newton-Krylov solver replacing PN-PCG’s Jacobi-preconditioned CG with HOT’s multigrid preconditioned CG; (4) and an MPM extension of the quasi-Newton LBFGS-H (FEM) from Li et al. [2019]. Note that unlike in Li et al. [2019] where the LBFGS-H is based on fully factorizing the beginning of timestep Hessian with a direct solver, here we only partially invert the Hessian by conducting Jacobi preconditioned CG iterations with adaptive terminating criteria identical to that of the coarsest-level solve in HOT (Section 5.3). In other words, it is an inexact LBFGS-H equivalent to a single-level HOT. We find that this inexact LBFGS-H often leads to better performance than those with direct solvers in large-scale problems.

²<https://github.com/squarefk/ziran2019>.

All methods in our ablation study together with Gast et al. [2015] are implemented in C++ and consistently optimized (see Section 6). Assembly and evaluations are parallelized with Intel TBB.

Simulation settings. Timestep sizes are set to $\min(\frac{1}{\text{FPS}}, 0.6 \frac{\Delta x}{v_{\max}})$ throughout all simulations. Here, 0.6 is selected to fulfill the CFL condition. We observe in our tests that a three-level multigrid preconditioner with one symmetric Gauss-Seidel smoothing and an inexact Jacobi PCG as our coarsest-level solver works best for both HOT and PN-MGPCG. We observe that a window size of 8 for LBFGS methods yields favorable overall performance. In our experiments, across a wide range of scenes, $\epsilon = 10^{-7}$ delivers consistent visual quality for all examples even when we vary materials with widely changing stiffness, shapes, and deformations.

Fixed corotated elasticity (FCR) from Stomakhin et al. [2012] is applied as our elasticity energy in all examples. In addition to our scenes with purely elastic materials, Twist (Figure 1), ArmaCat (Figure 3), Chain (Figure 10), and Faceless (Figure 11), we also test with plastic models, von Mises for Sauce (Figure 14) and for metal in Wheel (Figure 12), the center box in Boxes (Figure 4), and the bars in Donut (Figure 13), and granular flow [Stomakhin et al. 2013] in Boards (Figure 5).

We perform and analyze extensive benchmark studies on challenging simulation settings, where the material parameters are all from real-world data, and most of them are heterogenous. This not only significantly simplifies the material tuning procedure in animation production but also helps to achieve realistic physical behavior and intricate coupling between materials with varying stiffness. Figure 2 demonstrates that simulating a scene with aluminum sheets with inappropriate material parameters could end up getting unexpected behavior. See Table 1 for the physical parameters in daily life and Table 2 for material parameters used in our benchmark examples.

Detailed timing statistics for all examples are assembled in Table 1 and Table 2 from the supplemental material. As discussed in Section 5.2, all evaluated methods in our ablation study are terminated with the same tolerance computed from our extended CN for fair comparisons across all examples.

7.2 Performance

We analyze relative performance of HOT in two sets of comparisons. First, we compare HOT against the existing state-of-the-art implicit MPM methods [Fang et al. 2019; Gast et al. 2015]. Second, we perform an extensive ablation study to highlight the effectiveness of each of our design choices.

Comparison to state of the art. As discussed earlier, the performance and quality of results from Gast et al. [2015] are highly dependent, per example, on the choice of tolerance parameter. Across many simulation runs we sought optimal settings for this parameter to efficiently produce visually plausible, high-quality outputs. However, we find that suitable nonlinear tolerances vary extensively with different simulation conditions such as materials and boundary conditions. For example, we found an ideal tolerance for the Wheel example (Figure 12) at 10^2 , whereas for the Faceless example (Figure 11), 10^{-3} worked best. However, applying the 10^2 tolerance generates instabilities and even explosions for all other examples (see Figure 1 in the supplemental material), whereas

Table 2. Material Parameters

Example	Particle #	Δx (m)	Density (kg/m ³)	Young's Modulus (Pa)	ν
(Figure 1) Twist	230k	1×10^{-2}	2×10^3	$5 \times 10^5/5 \times 10^9$	0.4
(Figure 4) Boxes [†]	805k	8×10^{-3}	$2 \times 10^3/2.7 \times 10^3$	$2 \times 10^5/6.9 \times 10^{10}$	0.33
(Figure 13) Donut [†]	247k	5.7×10^{-3}	$2 \times 10^3/2.7 \times 10^3$	$1 \times 10^5/6.9 \times 10^{10}$	0.33
(Figure 3) ArmaCat	403k	4×10^{-2}	$1 \times 10^3/1 \times 10^3/2.5 \times 10^3$	$1 \times 10^5/1 \times 10^6/1 \times 10^9$	0.4/0.47/0.4
(Figure 10) Chain	308k	1×10^{-2}	$2 \times 10^3/2 \times 10^3$	$5 \times 10^5/3 \times 10^9$	0.4
(Figure 5) Boards*	188k	7×10^{-3}	1×10^3	$1 \times 10^5-1 \times 10^8$	0.33
(Figure 12) Wheel [†]	550k	5×10^{-3}	$2.7 \times 10^3/2.7 \times 10^3$	$1 \times 10^5/6.9 \times 10^{10}$	0.4/0.33
(Figure 11) Faceless	110k	1×10^{-2}	2×10^3	5×10^4	0.3
(Figure 14) Sauce [‡]	311k	1.5×10^{-2}	2.7×10^3	2.1×10^5	0.33

The von Mises yield stress: [†] 2.4×10^8 Pa; [‡]720 Pa. *The plastic flow from Stomakhin et al. [2013], where singular values of the deformation gradient are clamped into [0.99, 1.001].

using 10^{-3} tolerance produces extremely slow performance especially for examples containing stiff materials (see Table 1 in the supplemental material). As for ADMM MPM [Fang et al. 2019], as it is a first-order method, we observe slow convergence. Thus, we postpone detailed analysis to our convergence discussion below (Section 7.3). In contrast, HOT requires no parameter tuning. All results, timings, and animations presented here and in the following were generated without parameter tuning using the same input settings to the solver. As demonstrated they efficiently converge to generate consistent visual quality output.

Ablation study. We start with the homogeneous “faceless” example with a soft elastic material ($E = 5 \times 10^4$ Pa); we rotate and raise its head and then release. As shown in Figure 11, for this scene with moderate system conditioning, HOT already outperforms the two PN-PCG methods from our ablation set in almost every frame. Here, there is already a nearly 2× average speedup of HOT for the full simulation sequence compared to both the two PN-PCG variations, whereas the overall maximum speedup per frame is around 6×.

We then script a set of increasingly challenging stress-test scenarios across a wide range of material properties, object shapes, and resolutions (e.g., see Figures 3 and 14, as well as the supplemental video). For each simulation, we apply HOT with three levels so that the number of nodes is generally several thousand or smaller at the coarsest level. In Table 1 in the supplemental material, we summarize runtime statistics for these examples comparing HOT’s total wall clock speedup for the entire animation sequence, and maximum speedup factor per frame compared to PN-PCG, PN-PCG(MF), PN-MGPCG, and LBFGS-H across the full set of these examples.

Timings. Across this benchmark set, we observe that HOT has the fastest runtimes for all but two examples (see the following for discussion of these) over the best timing for each example across all methods: PN-PCG, PN-PCG(MF), PN-MGPCG, and LBFGS-H. Note that these variations for our ablation study already well exceed the state-of-the-art method from Gast et al. [2015] in most examples. In general, HOT ranges from 1.98× to 5.79× faster than PN-PCG, from 1.05× to 5.76× faster than PN-PCG(MF), from 2.26× to 10.67× faster than PN-MGPCG, and from 1.03× to 4.79× faster than LBFGS-H on total timing. The exceptions we observe are for the easy

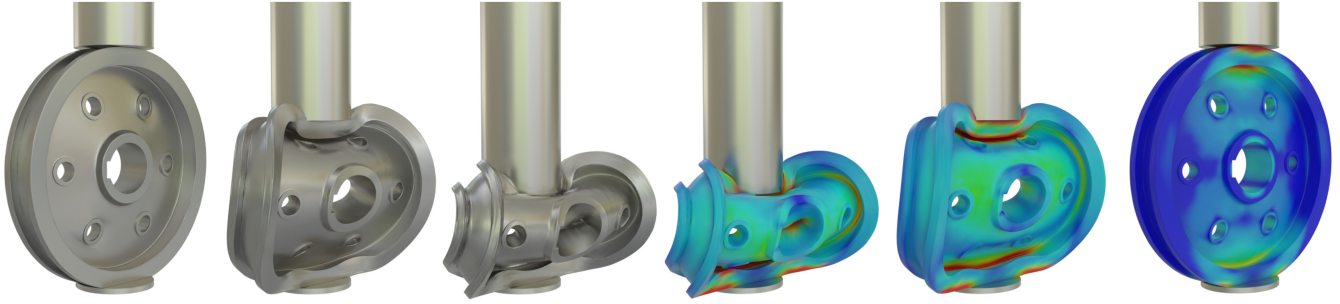


Fig. 12. Wheel. Our HOT integrator enables unified, consistent, and predictive simulations of metals with real-world mechanical parameters (with stress magnitude visualized).

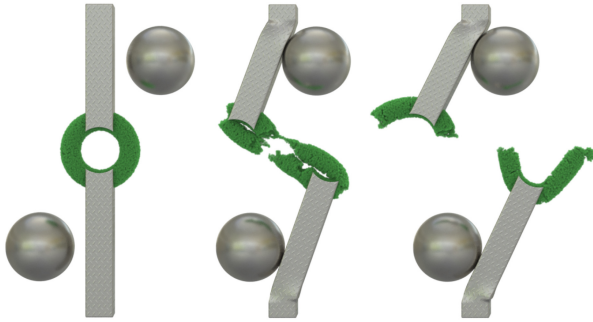


Fig. 13. Donut. An elastic torus is mounted between two metal bars. Collisions with rigid balls deform the attaching bars that then break the torus.

Sauce (Young's 2.1×10^5 Pa) and ArmaCat (Young's 10^6 Pa) examples, where materials are already quite soft and the deformation is moderate. In these two simple examples, HOT performs closely to the leading LBFGS-H method. However, when simulations become challenging, we observe that LBFGS-H can have trouble converging. This is most evident in the stiff aluminum Wheel example (Figure 12), where the metal is undergoing severe deformation. Here, HOT stays consistently efficient, outperforming all other methods. See our convergence discussion in the following for more details. Importantly, across examples, we observe that alternate methods PN-PCG, PN-PCG(MF), PN-MGPCG, and LBFGS-H swap as fastest per example so that it is never clear which would be best *a priori* as we change simulation conditions. Although in some examples each method can have comparable performance within $2\times$ slower than HOT, they also easily fall well behind both HOT and other methods in other examples (Figure 15). In other words, none of these other methods have even empirically consistent good performance across tested examples. The seemingly second-best LBFGS-H can even fail in some extreme cases. For most of the scenes with heterogeneous materials or large deformations, such as Twist, Boxes, Donut, and Wheel, which results in more PN iterations, PN-PCG is faster than its matrix-free counterpart PN-PCG(MF). Among these examples, only Boxes and Wheel can be well accelerated by using MGPCG for PN.

Gauss-Seidel preconditioned CG. Here we additionally compare the symmetric Gauss-Seidel (SGS) and Jacobi preconditioned PN-PCG to show that a simple trade-off between convergence and

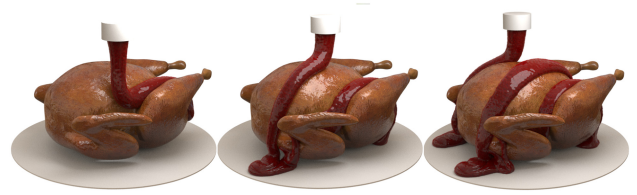


Fig. 14. HOT Sauce. HOT sauce is poured onto a turkey.

Table 3. Jacobi vs. Gauss-Seidel Preconditioned PN-PCG

Example	Symmetric Gauss-Seidel		Jacobi	
	Avg. Time	Avg. CG Iter.	Avg. Time	Avg. CG Iter.
Twist	492.20	679.91	361.53	1,054.16
Boxes	1,368.54	34.13	466.94	248.98
Donut	410.34	45.59	240.65	375.39
ArmaCat (soft)	109.08	29.04	111.04	66.43
ArmaCat (stiff)	148.38	62.63	153.84	157.77
Chain	398.79	47.27	572.01	92.67
Boards	371.66	59.17	313.62	206.35
Wheel	269.08	106.13	206.14	424.13
Faceless	12.22	16.14	7.21	63.80
Sauce	22.66	9.98	29.07	27.06

The runtime environment for all benchmark examples are identical to Table 1 from the supplemental material. Avg. Time measures average absolute cost (seconds) per playback frame, Avg. Iter. measures the average number of PCG iterations (per method) required per timestep to achieve the requested accuracy.

per-iteration cost might not easily lead to significant performance gain (Table 3). SGS preconditioned PN-PCG significantly improves convergence compared to Jacobi preconditioning as one would expect, but due to its more expensive preconditioning computation, the performance is right at the same level and sometimes even worse. This is also why we applied Jacobi preconditioned CG to solve our coarsest-level system.

Changing machines. Across different consumer-level Intel CPU machines, we tested (see Table 1 in the supplemental material), we see that HOT similarly maintains the fastest runtimes across all machines regardless of available CPU or memory resources over the best timing for each example between PN-PCG, PN-PCG(MF), PN-MGPCG, and LBFGS-H.

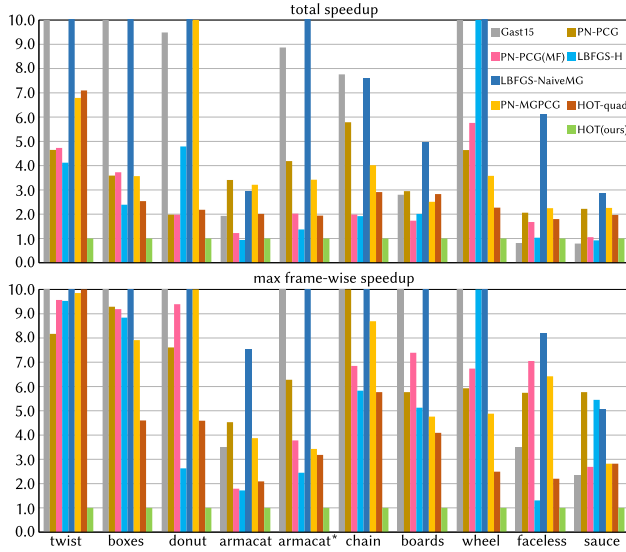


Fig. 15. Speedup overview. Top: We summarize method timings for all benchmark examples measuring the total runtime of each method normalized w.r.t. the timing of the HOT algorithm (“how HOT”) over each simulation sequence and so determine HOT’s speedup. Bottom: We comparably report the normalized maximum frame-wise timing of each method w.r.t. HOT across all benchmark examples and so again determine per-frame max speedup of HOT. Here, each simulation example is labeled at the bottom, where the cat in armacad and armacad* is with 1 MPa and 1 GPa Young’s modulus, respectively.

7.3 Convergence

HOT balances efficient, hierarchical updates with global curvature information from gradient history. In this section, we first compare HOT’s convergence to the state-of-the-art ADMM MPM [Fang et al. 2019] and then analyze the convergence behavior based on our ablation study. Here we exclude Gast et al. [2015], as the method applies a different stopping criteria and, as discussed earlier, requires intensive parameter tuning.

Comparison to ADMM. Here we compare to the ADMM-MPM [Fang et al. 2019] on a pure elasticity example faceless (Figure 11) by importing their open-sourced code into our codebase and adopted our nodewise CN based termination criteria (Section 5.2). Despite their advantages on efficiently resolving fully implicit visco-elasto-plasticity, on this pure elasticity example we observe that as a first-order method, ADMM converges much slower than all other Newton-type or Quasi-Newton methods including HOT (Figure 16). Although the overhead per iteration of ADMM is generally few times smaller, the number of iterations required to reach the requested accuracy is orders of magnitudes more. Nevertheless, ADMM-MPM is more likely to robustly generate visually plausible results within the first few iterations, whereas Newton-type or Quasi-Newton methods might not.

Ablation study. In Figure 17, we compare convergence rates and timings across methods for a single timestep of the Wheel example. In terms of convergence rate, we see in the top plot that PN-MGPCG obtains the best convergence, whereas HOT, PN-PCG, and PN-PCG(MF) converge similarly. Then, in this challenging

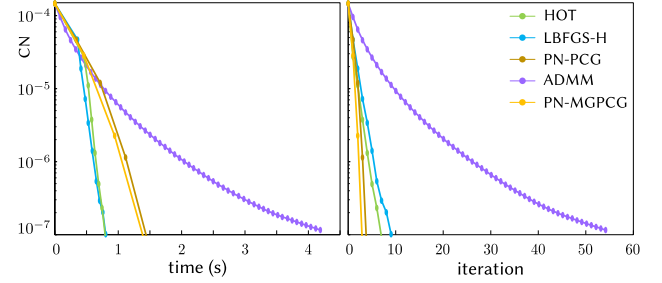


Fig. 16. Comparison to ADMM-MPM. CN-scaled gradient norm to timing and iteration plots for the first timestep of the faceless example (Figure 11) of all methods including ADMM-MPM [Fang et al. 2019]. With much lower per-iteration cost, ADMM quickly reduces the residual within the first few iterations (left). However, as a first-order method, it converges slowly to the requested accuracy compared to all others that converge superlinearly. As a result, it is 20× slower than our HOT.

example, LBFGS-H struggles to reach even a modest tolerance, as shown in the extension in the bottom plots of Figure 17.

However, for overall simulation time, HOT outperforms all three variations of PN and LBFGS-H due to its much lower per-iteration cost. PN-MGPCG, although with the best convergence rate, falls well behind HOT and only behaves slightly better than the two PN-PCG flavors, as the costly reconstruction of the multigrid hierarchy and the stiffness matrix are repeated in each Newton iteration. LBFGS-H then struggles where we observe that many linear solves well exceed the PCG iteration cap at 10,000. At the bottom of Figure 17, we see that LBFGS-H eventually converges after 400 outer iterations. Here, it appears that the diagonal preconditioner at the coarsest level in HOT significantly promotes the convergence of the whole solver; in contrast, although the same preconditioner in LBFGS-H loses its efficiency at the finest level, the system is much larger and conditioning of the system matrix exponentially exacerbates.

Visualization of convergence. In Figure 18 we visualize the progressive convergence of HOT and LBFGS-H w.r.t. the CN-scaled nodal residuals for the stretched box example. Here, HOT quickly smooths out low-frequency errors, as in iteration 6 the background color of the box becomes blue (small error) and the high-frequency errors are progressively removed until HOT converges in iteration 25. For LBFGS-H, both the low- and high-frequency errors are simultaneously removed slowly, and it takes LBFGS-H 106 iterations to converge.

Comparison to the baseline geometric multigrid. As discussed previously, building geometric multigrid directly from particle quadratures generally obtains essentially no speedup for coarser matrices and DoF mismatch. We compare to this baseline geometric multigrid on the ArmaCat example (Figure 2) by utilizing both multigrids in a PN-MGPCG time integrator. As we see in the top plot of Figure 19, geometric multigrid effectively achieves 5× faster convergence than PN-PCG with the Jacobi preconditioner but is still less effective than our 10× speedup in this specific timestep. The bottom plot shows that this convergence relation among all three candidates remains consistent throughout the animation of the ArmaCat example. However, in very few cases (e.g., Boards),

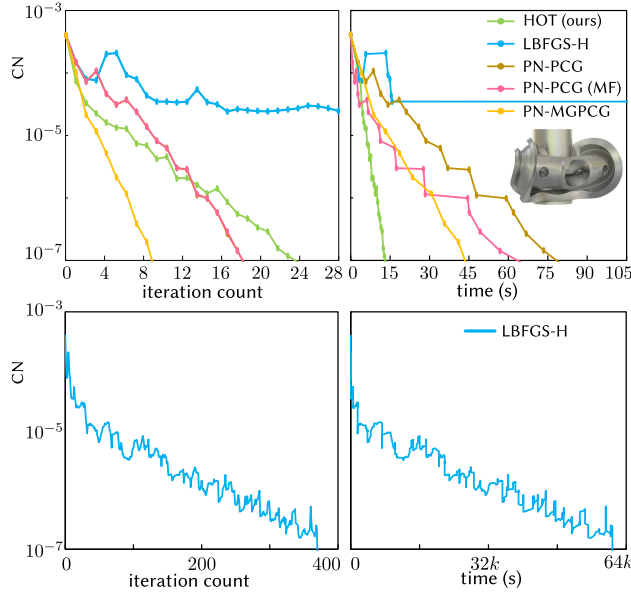


Fig. 17. Convergence comparisons. Top left: The iteration counts for the Wheel example w.r.t. CN of different methods are visualized. Here, PN-MGPCG demonstrates best convergence. Top right: Total simulation times of all methods w.r.t. CN are plotted; here, HOT, with low per-iteration cost, obtains superior performance across all methods. Bottom: In this extreme deformation, high-stiffness example LBFGS-H converges at an extremely slow rate.

we occasionally observed that baseline geometric multigrid preconditioned PN-MGPCG converges even more slowly than Jacobi preconditioned PN-PCG.

Then we compare HOT to applying GMG in LBFGS (LBFGS-GMG, see Table 2 in the supplemental material). We see that the convergence of LBFGS-GMG is orders of magnitude slower than HOT for all scenes containing stiff materials. For the two scenes with soft materials only (Sauce and Faceless), even if convergence is only slightly slower than HOT, the timing is more than 2× slower, which further demonstrates the inefficiency of the multigrid operations in GMG.

7.4 Varying Material Stiffness

Finally, to again consider consistency, we compare the convergence and overall performance of all five methods on the same simulation setup as we progressively increase the Young’s modulus of a material. Here we form a bar composed of three materials (see the inset of Figure 20). The two bar ends are kept with a fixed constant Young’s modulus of 10^5 Pa across all simulations in the experiment. We then progressively increase the Young’s modulus of the middle bar from 10^5 Pa up to 10^{10} Pa.

In the bottom plot of Figure 20, we see that HOT maintains a low and close to flat growth in iterations for increasing stiffness with PN-MGPCG demonstrating only a modestly greater iteration growth for stiffness increase. When we consider computation time, however, the modest growth in iterations for PN-MGPCG translates into much less efficient simulations as stiffness increases, due to PN-MGPCG’s much greater per-iteration cost.

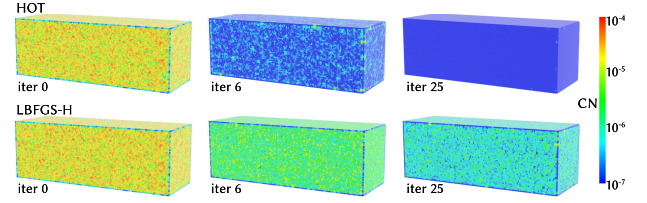


Fig. 18. Convergence visualization on a stretched box. A soft box deforms as its deformation gradient was initialized to some diagonal matrix with the diagonal entries randomly sampled in $[0.7, 1.3]$. The nodal CNs of different iterations in the first timestep are visualized on the rest shape. Here, HOT quickly removes low-frequency errors and converges in 25 iterations, whereas LBFGS-H converges in 106 iterations, removing both low- and high-frequency errors simultaneously.

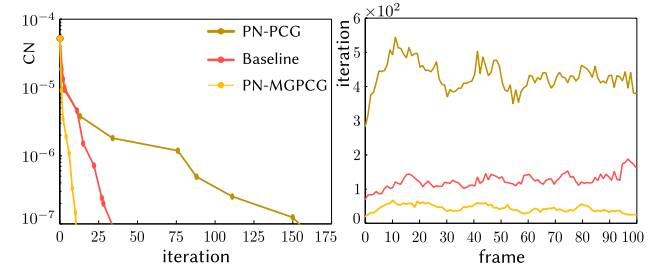


Fig. 19. Comparison to the baseline geometric multigrid. Left: CG iteration counts in one of the timesteps of the ArmaCat example w.r.t. CN of all methods. Right: Per-frame CG iteration counts. Here, the convergence of PN-MGPCG when using the baseline geometric multigrid is worse than using our node-embedding multigrid but better than Jacobi preconditioned PN-PCG. Moreover, timings of PN-PCG and PN-MGPCG highly overlap, and they are both 3× faster than the baseline overall.

Here, despite greater iteration growth, L-BFGS-H does better for scaling to greater stiffness due to its lower per-iteration cost. However, HOT with both a close-to-flat growth in iterations and low per-iteration cost maintains consistent behavior and generally best runtime performance with respect to increasing stiffness.

7.5 Ablation Study on HOT’s Kernel

Since our multigrid can be constructed using either linear or quadratic B-splines for node embedding that potentially leads to a trade-off between convergence and per-iteration cost because of the resulting coarser level shape functions, here we use an ablation study on the two design choices to back up our decision on using linear kernels. As shown in Table 2 from the supplemental material, HOT with linear embedding performs equally well on convergence compared to using quadratic embedding. In a few cases, such as Twist and Boxes, linear embedding converges much faster. This is reasonable, as we can see in Figure 7 the resulting shape functions on the coarser level when using linear or quadratic node embedding do not have significant differences. But because linear embedding leads to much sparser coarse systems, it is much faster on timing than quadratic embedding.

8 CONCLUSION AND FUTURE WORK

Starting from node embedding, we derive, to the best of our knowledge, the first MPM Galerkin multigrid. The resulting method

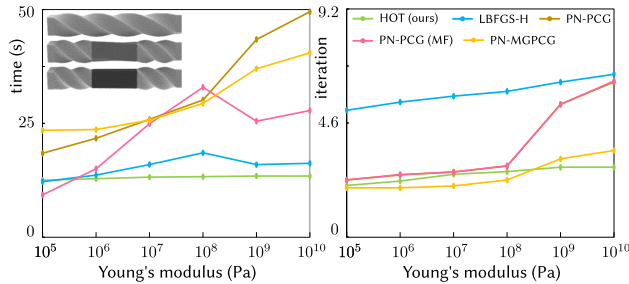


Fig. 20. Convergence and performance consistency for increasing stiffness. Twisting a multimaterial bar, we keep ends with fixed Young's moduli and progressively increase Young's for the middle segment. Across increasing stiffness HOT exhibits the best consistency w.r.t. both iteration count and the overall simulation time.

is consistent with particle quadrature geometric multigrid while still providing efficiency in construction and automatically handling boundary conditions. We then build HOT to utilize our multigrid as an inner initializer within L-BFGS to avoid the repeated expensive matrix reconstruction costs required in traditional PN-MGPCG. Together with efficient curvature updates, we ensure fast yet inexpensive convergence. HOT accelerates implicit MPM timestepping to a wide range of important material behaviors, achieving up to 10 $\times$ speedup compared to the state-of-the-art methods and shines among an exhaustive set of variations. It is our hope that this will enable further research to leverage HOT-type hierarchies to address both spatial and temporal discretization limitations in current MPM pipelines.

Semi-implicit plasticity is a limitation of HOT. For current HOT simulations with plasticity, we have not encountered instabilities. Nevertheless, adding plasticity return mapping as a constraint within HOT's stable implicit timestep optimization is an exciting and challenging future direction to explore. Another challenging but interesting extension would be to incorporate additional physical constraints in our minimization framework. One particularly useful case is volume preservation, such as for simulating stiff granular materials. Likewise, although our proposed inexact criteria resolves heterogeneous materials, we believe that it can and should be further improved. Currently, HOT offers consistent and performant MPM simulations to gain the advantages of implicit timestepping without the need for parameter tuning. We look forward to further extensions, such as those mentioned previously, and its application as a tool kit to accelerate the applications of MPM for the physical sciences, animation, engineering, and beyond.

On the implementation side, the construction of the finest-level system matrix is one of the bottlenecks for HOT. In our code, it is realized with a scattering scheme, which suffers from cache misses. Therefore, exploring the performance potential of alternative gathering schemes for building the stiffness matrices can be meaningful future work.

ACKNOWLEDGMENTS

We would like to thank Hannah Bollar for narrating the video, Joshua Wolper for proofreading, and the anonymous reviewers for their valuable comments.

REFERENCES

- M. Aanjaneya, M. Gao, H. Liu, C. Batty, and E. Sifakis. 2017. Power diagrams and sparse paged grids for high resolution adaptive liquids. *ACM Transactions on Graphics* 36, 4 (2017), Article 140.
- M. Adams, M. Brezina, J. Hu, and R. Tuminaro. 2003. Parallel multigrid smoothing: Polynomial versus Gauss-Seidel. *J. Comput. Phys.* 188, 2 (2003), 593–610.
- R. Ando, N. Thürey, and C. Wojtan. 2015. A dimension-reduced pressure solver for liquid simulations. *Computer Graphics Forum* 34, 2 (2015), 10.
- C. Batty, F. Bertails, and R. Bridson. 2007. A fast variational framework for accurate solid-fluid coupling. *ACM Transactions on Graphics* 26, 3 (2007), 100.
- S. Bouaziz, S. Martin, T. Liu, L. Kavan, and M. Pauly. 2014. Projective dynamics: Fusing constraint projections for fast simulation. *ACM Transactions on Graphics* 33, 4 (2014), 154.
- J. Brackbill, D. Kothe, and H. Ruppel. 1988. FLIP: A low-dissipation, particle-in-cell method for fluid flow. *Computer Physics Communications* 48, 1 (1988), 25–38.
- W. Briggs, Van E. Henson, and S. McCormick. 2000. *A Multigrid Tutorial* (2nd ed). SIAM.
- J. Brown and P. Brune. 2013. Low-rank quasi-Newton updates for robust Jacobian lagging in Newton-type methods. In *Proceedings of the International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C'13)*.
- T. J. Charlton, W. M. Coombs, and C. E. Augarde. 2017. iGIMP: An implicit generalised interpolation material point method for large deformations. *Computers & Structures* 190 (2017), 108–125.
- N. Chentanez and M. Muller. 2011. Real-time Eulerian water simulation using a restricted tall cell grid. *ACM Transactions on Graphics* 30, 4 (2011), Article 82, 10 pages.
- S. J. Cummins and J. U. Brackbill. 2002. An implicit particle-in-cell method for granular materials. *Journal of Computational Physics* 180, 2 (2002), 506–548.
- G. Daviet and F. Bertails-Descoubes. 2016. A semi-implicit material point method for the continuum simulation of granular materials. *ACM Transactions on Graphics* 35, 4 (2016), Article 102, 13 pages.
- P. Deufilhard. 2011. *Newton Methods for Nonlinear Problems: Affine Invariance and Adaptive Algorithms*. Springer Series in Computational Mathematics. Springer.
- D. Dinev, T. Liu, and L. Kavan. 2018a. Stabilizing integrators for real-time physics. *ACM Transactions on Graphics* 37, 1 (2018), Article 9.
- D. Dinev, T. Liu, J. Li, B. Thomaszewski, and L. Kavan. 2018b. FEPR: Fast energy projection for real-time simulation of deformable objects. *ACM Transactions on Graphics* 37, 4 (2018), Article 79, 12 pages.
- M. Ding, X. Han, S. Wang, T. F. Gast, and J. M. Teran. 2019. A thermomechanical material point method for baking and cooking. *ACM Transactions on Graphics* 38, 6 (2019), 192.
- O. Ding and C. Schroeder. 2019. Penalty force for coupling materials with Coulomb friction. *IEEE Transactions on Visualization and Computer Graphics*. Epub ahead of print. January 8, 2019.
- Y. Fang, Y. Hu, S. Hu, and C. Jiang. 2018. A temporally adaptive material point method with regional time stepping. *Computer Graphics Forum* 37 (2018), 195–204.
- Y. Fang, M. Li, M. Gao, and C. Jiang. 2019. Silly rubber: An implicit material point method for simulating non-equilibrated viscoelastic and elastoplastic solids. *ACM Transactions on Graphics* 38, 4 (2019), Article 118.
- Y. Fei, C. Batty, E. Grinspun, and C. Zheng. 2019. A multi-scale model for coupling strands with shear-dependent liquid. *ACM Transactions on Graphics* 38, 6 (Nov. 2019), Article 190, 20 pages.
- F. Ferstl, R. Westermann, and C. Dick. 2014. Large-scale liquid simulation on adaptive hexahedral grids. *IEEE Transactions on Visualization and Computer Graphics* 20, 10 (2014), 1405–1417.
- K. J. Fidkowski, T. A. Oliver, J. Lu, and D. L. Darmofal. 2005. p -Multigrid solution of high-order discontinuous Galerkin discretizations of the compressible Navier-Stokes equations. *Journal of Computational Physics* 207, 1 (2005), 92–113.
- M. Gao, A. Pradhana, X. Han, Q. Guo, G. Kot, E. Sifakis, and C. Jiang. 2018a. Animating fluid sediment mixture in particle-laden flows. *ACM Transactions on Graphics* 37, 4 (2018), 149.
- M. Gao, A. Pradhana, C. Jiang, and E. Sifakis. 2017. An adaptive generalized interpolation material point method for simulating elastoplastic materials. *ACM Transactions on Graphics* 36, 6 (2017), 223.
- M. Gao, X. Wang, K. Wu, A. Pradhana, E. Sifakis, C. Yuksel, and C. Jiang. 2018b. GPU optimization of material point methods. *ACM Transactions on Graphics* 37, 6 (2018), Article 254, 12 pages.
- T. Gast, C. Schroeder, A. Stomakhin, C. Jiang, and J. Teran. 2015. Optimization integrator for large time steps. *IEEE Transactions on Visualization and Computer Graphics* 21, 10 (2015), 1103–1115.
- J. Guilkey and J. Weiss. 2001. An implicit time integration strategy for use with the material point method. In *Proceedings of the 1st MIT Conference on Computational Fluid and Solid Mechanics*.
- J. E. Guilkey and J. A. Weiss. 2003. Implicit time integration for the material point method: Quantitative and algorithmic comparisons with the finite element

- method. *International Journal for Numerical Methods in Engineering* 57, 9 (2003), 1323–1338.
- Q. Guo, X. Han, C. Fu, T. Gast, R. Tamstorf, and J. Teran. 2018. A material point method for thin shells with frictional contact. *ACM Transactions on Graphics* 37, 4 (2018), 147.
- X. Han, T. F. Gast, Q. Guo, S. Wang, C. Jiang, and J. Teran. 2019. A hybrid material point method for frictional contact with diverse materials. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 2, 2 (2019), 17.
- Y. Hu, Y. Fang, Z. Ge, Z. Qu, Y. Zhu, A. Pradhana, and C. Jiang. 2018. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Transactions on Graphics* 37, 4 (2018), 150.
- C. Jiang, T. Gast, and J. Teran. 2017a. Anisotropic elastoplasticity for cloth, knit and hair frictional contact. *ACM Transactions on Graphics* 36, 4 (2017), Article 152.
- C. Jiang, C. Schroeder, A. Selle, J. Teran, and A. Stomakhin. 2015. The affine particle-in-cell method. *ACM Transactions on Graphics* 34, 4 (2015), Article 51, 10 pages.
- C. Jiang, C. Schroeder, and J. Teran. 2017b. An angular momentum conserving affine-particle-in-cell method. *Journal of Computational Physics* 338 (2017), 137–164.
- C. Jiang, C. Schroeder, J. Teran, A. Stomakhin, and A. Selle. 2016. The material point method for simulating continuum materials. In *Proceedings of the ACM SIGGRAPH 2016 Course*. Article 24, 52 pages.
- D. S. Kershaw. 1978. The incomplete Cholesky–conjugate gradient method for the iterative solution of systems of linear equations. *Journal of Computational Physics* 26, 1 (1978), 43–65.
- G. Klár, T. Gast, A. Pradhana, C. Fu, C. Schroeder, C. Jiang, and J. Teran. 2016. Drucker–prager elastoplasticity for sand animation. *ACM Transactions on Graphics* 35, 4 (2016), Article 103.
- M. Li, M. Gao, T. Langlois, C. Jiang, and D. Kaufman, M. 2019. Decomposed optimization time integrator for large-step elastodynamics. *ACM Transactions on Graphics* 38, 4 (2019), Article 70.
- T. Liu, S. Bouaziz, and L. Kavan. 2017. Quasi-Newton methods for real-time simulation of hyperelastic materials. *ACM Transactions on Graphics* 36, 3 (2017), Article 23.
- A. McAdams, E. Sifakis, and J. Teran. 2010. A parallel multigrid Poisson solver for fluids simulation on large grids. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA’10)*. 65–74.
- A. McAdams, Y. Zhu, A. Selle, M. Empey, R. Tamstorf, J. Teran, and E. Sifakis. 2011. Efficient elasticity for character skinning with contact and collisions. *ACM Transactions on Graphics* 30, 4 (2011), Article 37, 12 pages.
- K. Nagasawa, T. Suzuki, R. Seto, M. Okada, and Y. Yue. 2019. Mixing sauces: A viscosity blending model for shear thinning fluids. *ACM Transactions on Graphics* 38, 4 (July 2019), Article 95, 17 pages.
- A. Nair and S. Roy. 2012. Implicit time integration in the generalized interpolation material point method for finite deformation hyperelasticity. *Mechanics of Advanced Materials and Structures* 19, 6 (2012), 465–473.
- J. Nocedal and S. J. Wright. 2006. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer.
- M. Overby, G. Brown, J. Li, and R. Narain. 2017. ADMM $\supseteq$ projective dynamics: Fast simulation of hyperelastic models with dynamic constraints. *IEEE Transactions on Visualization and Computer Graphics* 23, 10 (2017), 2222–2234.
- A. Pradhana, T. Gast, G. Klár, C. Fu, J. Teran, C. Jiang, and K. Museth. 2017. Multi-species simulation of porous sand and water mixtures. *ACM Transactions on Graphics* 36, 4 (2017), Article 105.
- D. Ram, T. Gast, C. Jiang, C. Schroeder, A. Stomakhin, J. Teran, and P. Kavehpoor. 2015. A material point method for viscoelastic fluids, foams and sponges. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 157–163.
- R. Setaluri, M. Aanjaneya, S. Bauer, and E. Sifakis. 2014. SPGrid: A sparse paged grid structure applied to adaptive smoke simulation. *ACM Transactions on Graphics* 33, 6 (2014), Article 205, 12 pages.
- B. Smith, F. D. Goes, and T. Kim. 2018. Stable Neo-Hookean flesh simulation. *ACM Transactions on Graphics* 37, 2 (2018), Article 12, 15 pages.
- M. Steffen, R. Kirby, and M. Berzins. 2008. Analysis and reduction of quadrature errors in the material point method (MPM). *International Journal for Numerical Methods in Engineering* 76, 6 (2008), 922–948.
- A. Stomakhin, R. Howes, C. Schroeder, and J. Teran. 2012. Energetically consistent invertible elasticity. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 25–32.
- A. Stomakhin, C. Schroeder, L. Chai, J. Teran, and A. Selle. 2013. A material point method for snow simulation. *ACM Transactions on Graphics* 32, 4 (2013), Article 102, 10 pages.
- A. Stomakhin, C. Schroeder, C. Jiang, L. Chai, J. Teran, and A. Selle. 2014. Augmented MPM for phase-change and varied materials. *ACM Transactions on Graphics* 33, 4 (2014), 138.
- G. Strang and K. Arikka. 1986. *Introduction to Applied Mathematics*. Wellesley-Cambridge Press.
- K. Stüben. 2001. A review of algebraic multigrid. In *Numerical Analysis: Historical Developments in the 20th Century*, C. Brezinski and L. Wuytack (Eds.). Elsevier, 331–359.
- D. Sulsky, Z. Chen, and H. Schreyer. 1994. A particle method for history-dependent materials. *Computer Methods in Applied Mechanics and Engineering* 118, 1 (1994), 179–196.
- R. Tamstorf, T. Jones, and S. F. McCormick. 2015. Smoothed aggregation multigrid for cloth simulation. *ACM Transactions on Graphics* 34, 6 (2015), Article 245, 13 pages.
- J. Teran, E. Sifakis, G. Irving, and R. Fedkiw. 2005. Robust quasistatic finite elements and flesh simulation. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 181–190.
- R. Tielens, M. Möller, and K. Vuik. 2019. Efficient multigrid based solvers for B-spline MPM. In *Proceedings of the 2nd International Conference on the MPM for Modelling Soil-Water-Structure Interaction*.
- P. Wallstedt. 2009. *On the Order of Accuracy of the Generalized Interpolation Material Point Method*. University of Utah.
- H. Wang and Y. Yang. 2016. Descent methods for elastic body simulation on the GPU. *ACM Transactions on Graphics* 35, 6 (2016), 212.
- S. Wang, M. Ding, T. F. Gast, L. Zhu, S. Gagniere, C. Jiang, and J. M. Teran. 2019. Simulation and visualization of ductile fracture with the material point method. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 2, 2 (2019), Article 18.
- Z. Wang, L. Wu, M. Fratarcangeli, Min Tang, and Huamin Wang. 2018. Parallel multigrid for nonlinear cloth simulation. *Computer Graphics Forum* 37, 7 (2018), 131–141.
- M. Weiler, D. Koschier, and J. Bender. 2016. Projective fluids. In *Proceedings of the 9th International Conference on Motion in Games (MIG’16)*. 79–84.
- Z. Wen and D. Goldfarb. 2009. A line search multigrid method for large-scale nonlinear optimization. *SIAM Journal on Optimization* 20, 3 (2009), 1478–1503.
- J. Wolper, Y. Fang, M. Li, J. Lu, M. Gao, and C. Jiang. 2019. CD-MPM: Continuum damage material point methods for dynamic fracture animation. *ACM Transactions on Graphics* 38, 4 (2019), Article 119.
- J. Wrethorn, R. Armiento, and K. Museth. 2017. Animation of crack propagation by means of an extended multi-body solver for the material point method. *Computers & Graphics* 69, C (2017), 131–139.
- Z. Xian, X. Tong, and T. Liu. 2019. A scalable Galerkin multigrid method for real-time simulation of deformable objects. *ACM Transactions on Graphics* 38, 6 (2019), Article 162.
- Y. Yue, B. Smith, C. Batty, C. Zheng, and E. Grinspun. 2015. Continuum foam: A material point method for shear-dependent flows. *ACM Transactions on Graphics* 34, 5 (2015), Article 160, 20 pages.
- Y. Yue, B. Smith, P. Chen, M. Chantharayukhonthorn, K. Kamrin, and E. Grinspun. 2018. Hybrid grains: Adaptive coupling of discrete and continuum simulations of granular media. *ACM Transactions on Graphics* 37, 6 (2018), Article 283, 19 pages.
- X. Zhang and R. Bridson. 2014. A PPPM fast summation method for fluids and beyond. *ACM Transactions on Graphics* 33, 6 (2014), Article 206, 11 pages.
- X. Zhang, R. Bridson, and C. Greif. 2015. Restoring the missing vorticity in advection-projection fluid solvers. *ACM Transactions on Graphics* 34, 4 (2015), Article 52, 8 pages.
- X. Zhang, M. Li, and R. Bridson. 2016. Resolving fluid boundary layers with particle strength exchange and weak adaptivity. *ACM Transactions on Graphics* 35, 4 (2016), Article 76, 8 pages.
- Y. Zhu and R. Bridson. 2005. Animating sand as a fluid. *ACM Transactions on Graphics* 24, 3 (2005), 965–972.
- Y. Zhu, R. Bridson, and D. M. Kaufman. 2018. Blended cured quasi-Newton for distortion optimization. *ACM Transactions on Graphics* 37, 4 (2018), 40.
- Y. Zhu, E. Sifakis, J. Teran, and A. Brandt. 2010. An efficient and parallelizable multigrid framework for the simulation of elastic solids. *ACM Transactions on Graphics* 29, 16 (2010), 1–16.

Received November 2019; revised February 2020; accepted February 2020