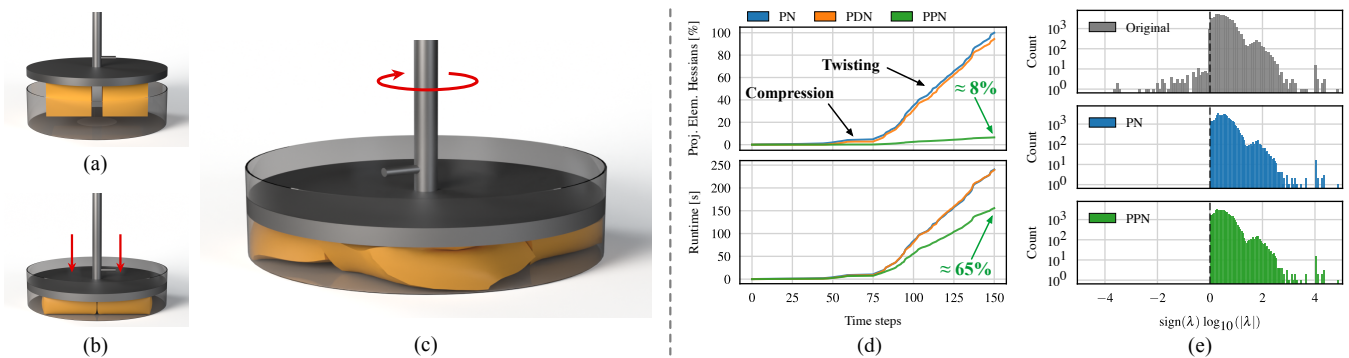


# Progressively Projected Newton's Method

J. A. Fernández-Fernández<sup>1</sup> , F. Lösschner<sup>1</sup>  and J. Bender<sup>1</sup> 

<sup>1</sup>RWTH Aachen University, Aachen, Germany



**Figure 1: Press.** (a) Four elastic orange boxes drop into a rigid cylinder; (b) a press descends; (c) the press then rotates, forcing the boxes to roll under pressure due to friction. (d) Top: cumulative count of projected element Hessians, highlighting that PPN projects only 8% of all elements; bottom: cumulative runtime highlighting that PPN completes the simulation in 65% of the time required by PN and PDN. (e) Global eigenvalue histogram at time step 125, iteration 4: both PN and PPN handle negative eigenvalues as expected, though not equally; while PN projects all the element Hessians at this iteration, PPN only projects 17.4% of them.

## Abstract

Newton's Method is widely used to find the solution of complex non-linear simulation problems. To guarantee a descent direction, it is common practice to clamp the negative eigenvalues of each element Hessian prior to assembly—a strategy known as Projected Newton (PN)—but this perturbation often hinders convergence.

In this work, we observe that projecting only a small subset of element Hessians is sufficient to secure a descent direction. Building on this insight, we introduce Progressively Projected Newton (PPN), a novel variant of Newton's Method that uses the current iterate's residual to cheaply determine the subset of element Hessians to project. The benefit is twofold: most eigendecompositions are avoided and the global Hessian remains closer to its original form, reducing the number of Newton iterations.

We compare PPN with PN and Project-on-Demand Newton (PDN) in a comprehensive set of experiments covering contact-free and contact-rich deformables, co-dimensional and rigid-body simulations, and a range of time step sizes, tolerances and resolutions. PPN reduces the amount of element projections in dynamic simulations by one order of magnitude while simultaneously improving convergence, consistently being the fastest solver in our benchmark.

## CCS Concepts

• **Computing methodologies** → **Modeling and simulation**;

## 1. Introduction

Robust and efficient simulation of dynamic deformable and rigid objects is a cornerstone of computer graphics, visual effects, and interactive design. Many modern simulators perform time-stepping by formulating implicit integration schemes as successive

nonlinear optimization problems, each solved with a variant of Newton's method. When the assembled Hessian of the potential energy is *Symmetric Positive Definite* (SPD), Newton steps are guaranteed to point in descent directions and efficient linear solvers

specialized for SPD matrices can be applied, both highly sought-after properties.

In practice, nonlinear materials, large time steps, and other factors frequently render the assembled Hessian indefinite, causing the line search to fail. As a consequence, practically every nonlinear second-order solver must implement a mechanism to remove indefiniteness from the system. This stabilization process therefore becomes a fundamental component of this type of solvers, with a strong influence on their robustness and performance. A widely adopted remedy is *Projected Newton* (PN): an eigendecomposition is performed for every element Hessian and its negative eigenvalues are clamped (or mirrored) before assembly. Although PN produces an SPD system, it overly distorts the global Hessian and thus slows down convergence by unnecessarily discarding element-local negative curvature information. Moreover, projecting all elements imposes the cost of one eigendecomposition per element per Newton step, which can be expensive. The root cause of these important drawbacks is the per-element *isolated* view on indefiniteness, which is overly limiting when analyzing the actual indefiniteness of the global system. While analytic eigenanalysis can alleviate performance concerns, it can be challenging to derive the required expressions when modeling complex effects or materials not yet established in the literature. This approach is also incompatible with many simulation frameworks that use automatic or symbolic differentiation for derivative-free modeling and which have to rely on numerical eigendecompositions for flexibility.

We propose *Progressively Projected Newton* (PPN), a direct replacement for PN for dynamic simulations that avoids most of the element Hessian projections while ensuring descent directions during the Newton search. The key improvement of PPN over existing solutions is to identify sources of indefiniteness *after* the additive effect of assembly, rather than from an isolated, per-element perspective. Each Newton iteration begins with the unaltered Hessian, exactly as in pure Newton's Method. If the linear solver exits early due to indefiniteness, PPN projects only those elements whose local residual exceed an adaptive tolerance, updates the global matrix incrementally, and retries the solve. The tolerance is tightened until a descent direction is reached, then relaxed for the next iteration. Thus, PPN trades a few inexpensive solver attempts for the elimination of most eigendecompositions and a global Hessian closer to the unmodified one, which often leads to fewer Newton iterations.

PPN behaves exactly like Newton's Method if no projections are required, and degenerates to full PN only in very rare cases. By projecting only when and where necessary, PPN provides faster convergence than existing methods in dynamic simulations, while maintaining the established and reliable robustness of PN. In typical simulations, PPN omits more than 90% of element projections, cutting its cost by one order of magnitude, and reducing Newton iterations by up to 50% compared to PN. In our implementation, PPN achieves speedups of up to  $\times 2.5$  over PN and up to  $\times 1.5$  over the best alternative.

Crucially, PPN is formulated for dynamics, explicitly exploiting the regularization inherent in time integration, most notably the mass matrix. This allows it to avoid full-spectrum strategies by recognizing that sources of indefiniteness are highly localized

in time and space, effectively rendering spectral verification unnecessary for the vast majority of elements. To the best of our knowledge, PPN is the first method to successfully circumvent this global processing requirement.

In summary, our contributions are:

- Progressively Projected Newton, a novel Newton-type solver that selectively projects element Hessians, drastically cutting eigendecomposition costs and avoiding unnecessary distortions to the global Hessian.
- A residual-driven heuristic that ranks elements by their likelihood of contributing negative curvature which reuses information already calculated in the original Newton's Method.
- A comprehensive evaluation on contact-free and contact-rich deformable bodies, shells, and rigid body systems.

## 2. Related Work

We first cover works using optimization-based time integration as our main area of application. We then discuss eigenvalue filtering and briefly introduce “automatic” frameworks that use machine-generated derivatives as a prominent use case to apply progressive projections.

### 2.1. Optimization-based Time Integration

Optimization-based time integrators reformulate implicit schemes such as backward Euler as Incremental Potential minimization problems [RO99], which enables the use of robust optimization methods [NW06] in order to advance dynamic simulations in time. Many works in the computer graphics community adopted this approach [KYT\*06, MTGG11, LBOK13]. First-order and quasi-Newton solvers reduce per-iteration cost at the expense of more iterations [BML\*14, OBLN17, MEM\*20, WY16, LBK17, CLY24]. Despite relatively higher per-iteration costs, second-order approaches are widely used due to strong convergence guarantees [GSS\*15]. Applications include frictional contact [LFS\*20], cloth and rods [LKJ21, CXY\*23], rigid-bodies [FLS\*21, LKL\*22], advanced materials [LFFJ\*23, LFFJB24] and fluids and granular media [XLYJ23, LLH\*24]. For a comprehensive overview of such energy-based models and their coupling we refer readers to the recent multiphysics state-of-the-art report by Holz et al. [HJL\*25]. Our proposed Newton-type solver is a drop-in replacement that can improve efficiency in these applications while retaining robustness.

### 2.2. Eigenvalue Filtering

To obtain descent directions during optimization, existing Newton-type solvers typically rely on the global Hessian being SPD. For most physical models, this is not always the case in practice. To address this issue, Teran et al. [TSIF05] proposed per-element Hessian projection to the cone of SPD matrices by clamping their negative eigenvalues prior to assembly. This technique, commonly referred to as Projected Newton [SPSH\*17], avoids infeasible global eigendecomposition and facilitates use of linear solvers specific to SPD matrices. The success of PN motivated a large body of work on efficient per-energy analytic eigenanalysis to

avoid expensive numerical eigendecompositions [SGK18, SGK19, Kim20, KDG19, WYW23, LCK22, SK23, WK23, HCLK24]. However, these analytic projections require careful manual modifications of the second-order derivative implementations. Further approaches for SPD projection include regularization using diagonal matrices [FL16] or multiples of the mass matrix [LLFF\*23], but these are shown to perform worse than PN in contact-rich scenarios [LLFF\*23].

In the quasistatic setting with strong volume conservation and large initial deformations, eigenvalue *mirroring* [CLL\*24] and variants of *blending* [CLF25, CLJ\*24], as opposed to clamping, have shown to improve convergence. Unfortunately, as we show in Section 5, these results do not directly transfer to dynamic problems.

A comprehensive study by Longva et al. [LLFF\*23] recently demonstrated that unconditional projection slows asymptotic convergence and breaks affine invariance. Their proposed *Project-on-Demand Newton* (PDN) method performs element projections only when the assembled matrix is detected to be indefinite, which typically occurs far from the solution, recovering Newton-like convergence as the iteration sequence progresses.

The aforementioned strategies share the limitation of acting on *all* elements and *in isolation*, ignoring the additive effect of neighboring contributions, and resulting in “over-projection”. Our progressive strategy not only projects on-demand, but also selectively, significantly reducing both the amount of element projections and the distortion imposed on the global Hessian, while still guaranteeing descent directions.

### 2.3. Automatic Frameworks

Recently, a proliferation of frameworks that automate solutions to second-order optimization has occurred in geometry processing and simulation. These systems rely on machine-generated derivatives and automated evaluation pipelines to tackle complex problems from concise symbolic expressions [SBB\*22, HSK24, FFLW\*25]. They enable rapid and safe composition of solvers and models, thus accelerating research with a measurable impact on the field; several of the referenced works above, for example, were developed on TinyAD [SBB\*22] and SymX [FFLW\*25].

Because semi-analytic projection code remains challenging to generate, these frameworks still rely on numerical eigendecompositions, which is shown to be a very significant cost in our measurements (Section 5). By avoiding most projections, PPN eliminates this bottleneck and further narrows the performance gap with hand-tuned codebases.

### 3. Newton's Method

We seek the configuration  $\mathbf{x} \in \mathbb{R}^n$  that minimizes the total potential energy  $\Psi(\mathbf{x})$ , which typically aggregates inertial, elastic, frictional and other contributions evaluated on a discretized domain (see, e.g., [GSS\*15]). Applying Newton's method to this optimization problem, at iteration  $k$  we solve

$$\mathbf{H}^k \Delta \mathbf{x}^k = -\mathbf{g}^k, \quad (1)$$

where  $\mathbf{g}^k = \nabla_{\mathbf{x}} \Psi(\mathbf{x}^k)$  is the gradient of the energy,  $\mathbf{H}^k = \nabla_{\mathbf{x}}^2 \Psi(\mathbf{x}^k)$  is the Hessian and  $\Delta \mathbf{x}^k$  is the Newton step such that  $\mathbf{x}^{k+1} = \mathbf{x}^k + \Delta \mathbf{x}^k$ . This scheme is applied iteratively until a measure of convergence is fulfilled, e.g.  $\mathbf{g}^k \approx \mathbf{0}$ . Both the global Hessian and gradient are assembled from element contributions  $\mathbf{H}^k = \sum_e \mathbf{H}^{e,k}$  and  $\mathbf{g} = \sum_e \mathbf{g}^{e,k}$ . The elements  $e$  are typically given by finite elements (e.g. tetrahedra), rigid bodies, particles, collision pairs et cetera. For clarity we omit the superscript  $k$  hereafter.

A non-zero Newton step  $\Delta \mathbf{x}$  is guaranteed to point in a descent direction if  $\mathbf{H}$  is SPD locally at the current iterate, that is,

$$\mathbf{r}^T \mathbf{H} \mathbf{r} > 0, \quad \forall \mathbf{r} \neq \mathbf{0}, \quad (2)$$

which implies that  $\Psi$  features strictly positive curvature locally in every direction, i.e. it is locally *convex*. The connection between an SPD Hessian and a descent direction can be shown by multiplying both sides of Eq. (1) by  $\Delta \mathbf{x}^T$

$$\Delta \mathbf{x}^T \mathbf{g} = -\Delta \mathbf{x}^T \mathbf{H} \Delta \mathbf{x}. \quad (3)$$

If  $\mathbf{H}$  is SPD, the right-hand side is strictly negative, hence  $\Delta \mathbf{x}^T \mathbf{g} < 0$  [NW06].

In practice however,  $\mathbf{H}$  is often indefinite (see, e.g., [KE22]). PN remedies this by filtering (e.g. clamping) the negative eigenvalues of the element Hessians prior to assembly. Consider the eigendecomposition of the Hessian of element  $e$

$$\mathbf{H}^e = \mathbf{Q}^e \mathbf{D}^e (\mathbf{Q}^e)^T, \quad (4)$$

where the columns of  $\mathbf{Q}^e$  are the eigenvectors of  $\mathbf{H}^e$  and  $\mathbf{D}^e$  is a diagonal matrix of the corresponding eigenvalues. Applying clamping, the respective SPD projected element Hessian is then  $\hat{\mathbf{H}}^e = \mathbf{Q}^e \hat{\mathbf{D}}^e (\mathbf{Q}^e)^T$ , for  $\hat{D}_{ii}^e = \max(D_{ii}^e, \epsilon)$  with  $\epsilon > 0$ . As a sum of SPD matrices is SPD, which holds for the assembled global matrix.

However, projecting all the element Hessians is unnecessary. Consider element matrices  $A$  and  $B$ , and the global matrix  $P$ :

$$A = \begin{pmatrix} -1 & 0 \\ 0 & 2 \end{pmatrix}, \quad B = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}, \quad P = A + B = \begin{pmatrix} 1 & 0 \\ 0 & 4 \end{pmatrix}.$$

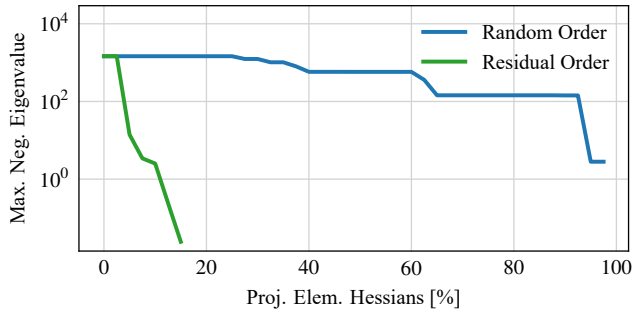
$A$  is indefinite with eigenvalues  $\{-1, 2\}$ ,  $B$  is SPD with eigenvalues  $\{2, 2\}$ , yet the assembled  $P$  is SPD with eigenvalues  $\{1, 4\}$ . Projecting  $A$  would alter the global matrix unnecessarily and likely deteriorate convergence of Newton's method as shown by Longva et al. [LLFF\*23]. Project-on-Demand Newton addresses this issue by projecting all element Hessians only if the global matrix is proven indefinite, generally improving convergence over PN.

### 4. Progressively Projected Newton's Method

In this section we introduce our method, PPN, starting with a motivating example. Consider a third element matrix added to the previous ones:

$$C = \begin{pmatrix} -10 & 0 \\ 0 & 1 \end{pmatrix}, \quad Q = A + B + C = \begin{pmatrix} -9 & 0 \\ 0 & 5 \end{pmatrix}.$$

In this case,  $Q$  is indefinite with eigenvalues  $\{-9, 5\}$ . However, only projecting  $C$  suffices to obtain an SPD approximation, demonstrating that full projection, even when conditionally



**Figure 2:** Largest global negative eigenvalue as element Hessians are progressively projected for the iteration shown in Figure 1 (d) and (e). In blue, randomized element order. In green, elements ordered by the local assembled residual from largest to smallest.

triggered as PDN does, is still unnecessary. This *element-selective projection* is the core idea of PPN: the global Hessian  $\mathbf{H}$  is built from two disjoint sets of element Hessians, the projected set  $\mathcal{H}_p$  and the unprojected set  $\mathcal{H}_u$ . The goal then is to keep  $|\mathcal{H}_p|$  minimal while ensuring that the assembled Hessian yields a descent direction at all times during Newton iterations.

In practice, however, identifying which element Hessians to project is not as straightforward as in the example above. Assembly scatters element contributions, making global eigenvalue recovery intractable, and building  $\mathcal{H}_p$  based on exact global eigenvalues would be prohibitively expensive. Instead, we propose a cheap post-assembly proxy for the *likelihood* that an element is the source of global indefiniteness. We pair this prediction with a robust global indefiniteness check to adapt  $\mathcal{H}_p$  as needed. Both the predictor and the indefiniteness check derive efficiently from information already available during the Newton minimization process.

**Projection heuristic** We motivate a residual-based heuristic to rank elements by their likelihood of contributing to global indefiniteness. Let  $\mathbf{x}^*$  be a strict local minimizer with  $\mathbf{g}(\mathbf{x}^*) = \mathbf{0}$  and  $\Psi(\mathbf{x}^* + \epsilon) > \Psi(\mathbf{x}^*)$  for small  $\epsilon$ . By Eq. (2),  $\mathbf{H}(\mathbf{x}^*)$  is SPD, and by continuity it remains (semi)definite in a neighborhood of  $\mathbf{x}^*$ . Hence, as  $\mathbf{x} \rightarrow \mathbf{x}^*$ ,  $\mathbf{g} \rightarrow \mathbf{0}$  and indefiniteness in  $\mathbf{H}$  diminishes. However, convergence does not occur uniformly across the domain. We observed that elements in regions with lower assembled residual are less likely to induce indefiniteness in the global system, even if left unprojected. Figure 2 empirically supports this hypothesis: while a random ordering requires the projection of almost all elements to recover global positive definiteness, projecting in order of larger assembled residual first isolates the indefinite contributions rapidly. Elements in low-residual regions are more likely to either already have all-positive eigenvalues or their minor indefinite spectra cannot overturn stronger SPD contributions, such as the influence of the mass matrix. As demonstrated in Section 5, this correlation is robust and consistent across a diverse set of scenarios, material models, and discretization parameters.

Motivated by this, we prioritize projecting element Hessians in regions with larger assembled residual and expand as necessary. Specifically, we introduce a projection tolerance  $\delta$  and add  $\mathbf{H}^e$

to  $\mathcal{H}_p$  when the element-local residual of  $e$  is larger than  $\delta$ , that is,  $\|\mathbf{S}^e \mathbf{g}\|_\infty > \delta$ , where  $\mathbf{S}^e$  is a selection matrix that extracts the assembled entries affected by element  $e$ . The  $\infty$ -norm captures the largest component of the local residual, acting as a selector of all elements incident to nodes with large residuals, aligning with the concept of domain partitioning. The gradient, or residual, relates directly to stationarity where the system is known to be SPD while other progress metrics, such as the Newton step or iteration decrements, depend on the optimization procedure. The tolerance  $\delta$  is adapted over the course of the Newton iterations by a tightening factor  $\alpha \in (0, 1)$  and a release factor  $\beta \geq 1$ . The former is applied when indefiniteness is detected in the global Hessian, and the latter after a successful step is taken. As a result, we obtain an effective partition using already calculated values at no extra cost. This parametrization exhibits low sensitivity, yielding consistent performance gains across broad ranges of values. Values of  $\alpha = 0.5$  and  $\beta = 2$  are chosen as safe defaults by ablation tests in Section 5. To guarantee termination and bound the worst-case runtime, we fall back to PN and project all elements if  $\delta$  drops below a small threshold  $\epsilon_\delta = 10^{-12}$ .

---

#### Algorithm 1: Progressively Projected Newton (PPN)

---

```

1  $\delta \leftarrow \infty$ 
2 while not converged do
3   Assemble unprojected  $\mathbf{H}$  and  $\mathbf{g}$ 
4    $(\Delta\mathbf{x}, ind) \leftarrow \text{SOLVE\_SPD}(\mathbf{H}, -\mathbf{g})$ 
5   while ind do
6     if  $\delta = \infty$  then
7        $\delta \leftarrow \alpha \|\mathbf{g}\|_\infty$ 
8     foreach  $e$  do
9       if  $(\|\mathbf{S}^e \mathbf{g}\|_\infty > \delta)$  or  $(\delta \leq \epsilon_\delta)$  then
10         $\text{PROJECT\_TOPD}(\mathbf{H}, e)$  ▷ In-place update
11         $(\Delta\mathbf{x}, ind) \leftarrow \text{SOLVE\_SPD}(\mathbf{H}, -\mathbf{g})$ 
12        if ind then
13           $\delta \leftarrow \alpha \delta$  ▷ Tighten projection
14         $\delta \leftarrow \beta \delta$  ▷ Relax projection
15         $\gamma \leftarrow \text{LINESEARCH}(\Delta\mathbf{x})$ 
16         $\mathbf{x} \leftarrow \mathbf{x} + \gamma \Delta\mathbf{x}$ 

```

---

**Algorithm** Our method is outlined in Algorithm 1. The standard Newton's Method logic with an SPD linear system solver is unmodified except for the inner PPN projection logic (lines 5 to 13). Note that the linear solver must report whether indefiniteness was encountered in addition to the solution  $\Delta\mathbf{x}$ . The first time projection is needed,  $\delta$  is initialized in relation to the largest absolute value of the current residual (lines 6 and 7). If the system still cannot be solved,  $\delta$  is reduced (lines 12 and 13). Once the linear solve is successful,  $\delta$  is increased for the next Newton iteration (line 14). Eigenvalue clamping [TSIF05] is used for the projection of element Hessians.

#### 4.1. Implementation

PPN integrates naturally into existing PN pipelines, requiring only two operations to be done efficiently: incremental global

Hessian updates, and an SPD linear solver that exits early upon indefiniteness.

**Incremental Hessian updates** When element  $e$  moves from  $\mathcal{H}_u$  to  $\mathcal{H}_p$  the update can be done by assembling the difference

$$\Delta \mathbf{H}^e = \widehat{\mathbf{H}}^e - \mathbf{H}^e \quad (5)$$

into the global matrix in place, which can reuse existing assembly routines. Importantly, this operation does not change the sparsity of  $\mathbf{H}$ , which should make updates faster than the original assembly.

**Linear Solvers** In this work we consider Preconditioned Conjugate Gradient (PCG) and Cholesky factorization solves (LLT), which can both exit early on indefiniteness for significantly lower cost than the total cost of the linear solve.

The numerical factorization of LLT exits on the first negative pivot encountered. Since the expensive symbolic analysis can be reused as long as the sparsity pattern does not change, failing is amortized with the eventual successful solve.

In the case of PCG, we monitor its intermediate value  $\mathbf{d}^T \mathbf{H} \mathbf{d}$  for each CG search direction  $\mathbf{d}$ . If a direction of negative curvature is encountered, indefiniteness is confirmed and the linear solver is stopped. Thus, when equipped with (P)CG, PPN draws parallels with the “Newton-CG” method: instead of using the last valid intermediate solution of CG as the Newton step (which might be very inaccurate), we restart the CG solve with more projections applied. Even if we do not eliminate all indefiniteness from the global Hessian, as long as CG does not encounter a negative search direction, the resulting intermediate solution is guaranteed to be a descent direction as in Newton-CG [NW06]. In our experiments, warm-starting subsequent PCG calls with the last descent direction yielded worse results than simply starting every solve with the zero vector, hence we use the latter approach.

**Multithreading** While our method adds a few additional steps, it does not introduce thread load imbalance and remains friendly to multithreading: the unprojected Hessian is built (all threads, unrelated to PPN), the linear solver is run (all threads, also unrelated to PPN), and, if indefiniteness is detected, a selection of elements is projected in place (all threads). Section 5 demonstrates our method’s performance on a high-core-count workstation CPU.

## 5. Results

In this section, we present a comprehensive suite of experiments to compare PPN with PN and PDN across a variety of simulations. Before that, we describe the hardware, software, and models used in our experiments, followed by an ablation study on PPN’s parameters.

### 5.1. Experimental Setup

**Hardware and software.** All experiments are conducted on a workstation equipped with a 3.60 GHz AMD Ryzen Threadripper PRO 5975WX processor (32 cores, 64 threads) and 256 GB of RAM. Code is compiled with gcc 12.2 and built on top of the open-source ÁTARK Ámulation Áramework ÁFFLL\*24]. We Áse the

© 2026 The Author(s).

Computer Graphics Forum published by Eurographics and John Wiley & Sons Ltd.



**Figure 3: Rolling sphere.** A deformable sphere with initial horizontal velocity rolls on a flat surface. Different Newton step velocity tolerance are used:  $10^{-1} \text{ms}^{-1}$  (yellow),  $10^{-2} \text{ms}^{-1}$  (blue),  $10^{-3} \text{ms}^{-1}$  (green),  $10^{-4} \text{ms}^{-1}$  (red).

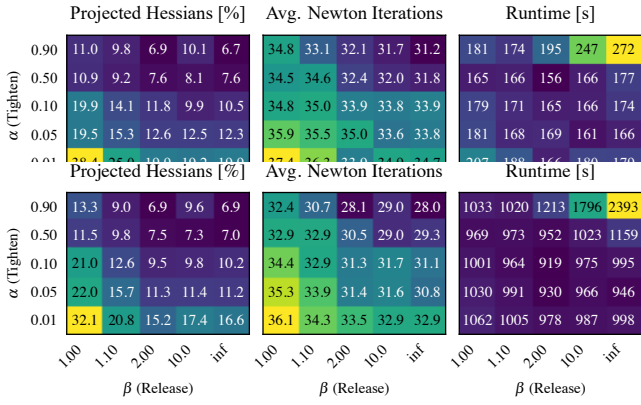
framework’s built-in  $3 \times 3$  Blocked Diagonal PCG solver and Intel MKL 2025 for Cholesky factorization. Eigen 3.4 handles all other linear algebra operations, including eigendecompositions, which we measure to be on average  $\times 1.53$  faster than MKL’s ones for matrices of size  $15 \times 15$  and smaller.

**Time stepping and tolerances.** We use the backward Euler scheme for time stepping and, unless otherwise stated, a time step size of  $\Delta t = 1/30 \text{s}$ . As stopping tolerance for Newton’s method we check if the velocity step infinity-norm  $\Delta t^{-1} \|\Delta \mathbf{x}\|_{\infty}$  falls below  $10^{-3} \text{ms}^{-1}$ . The choice of tolerance greatly influences the number of Newton iterations. The experiment shown in Fig. 3 justifies our choice: a tolerance of  $10^{-2} \text{ms}^{-1}$  or coarser causes outcome-altering energy losses, while tolerances of  $10^{-3} \text{ms}^{-1}$  and  $10^{-4} \text{ms}^{-1}$  lie much closer. For consistent comparisons and to avoid bias from inexactness, we verify convergence across solvers with a final fully projected Hessian solve (as PN would) via LLT factorization. Because such a validation solve is not typically performed in production, we exclude its cost from all timing measurements. PCG uses a relative residual tolerance of  $\|\mathbf{r}\| \|\mathbf{r}^0\|^{-1} = 10^{-4}$ . Element eigenvalues are clamped to  $\epsilon = 10^{-8}$ . For PDN, we adopt the countdown of 4 suggested in the original paper, which also yielded the best results in our “Press” benchmark.

**Boundary conditions and materials.** Dirichlet constraints are enforced using penalty potentials. All elastic solids employ the Neo-Hookean material in 2D and the Stable Neo-Hookean [SGK18] model in 3D. Frictional contact uses the IPC [LFS\*20] potentials with backtracking line search for sufficient descent and intersection-based filtering. We use the rigid body inertial potential by Macklin et al. [MEM\*20]. A list of material parameters, mesh sizes, and time step settings is provided in Table 1.

### 5.2. Experiments

**Ablation** We systematically analyze the sensitivity of PPN to the tightening ( $\alpha$ ) and release ( $\beta$ ) rates. Figures 4 and 5 map the performance landscape for the “Press” (Fig. 1) and “Armadillo Slingshot” (Fig. 11) scenes, covering both iterative (PCG) and direct (LLT) linear solvers. These scenes provide complementary stress-tests: “Press” is dominated by compression and friction-induced indefiniteness, whereas “Armadillo Slingshot” characterizes a regime of extreme tensile strains without contact. In the aggressive limit ( $\alpha = 0.9, \beta \rightarrow \infty$ ), the method successfully minimizes projection counts ( $< 7\%$  and  $< 2\%$  respectively), but total runtime degrades due to the overhead of recovering from failed indefinite linear solves. Conversely, conservative settings



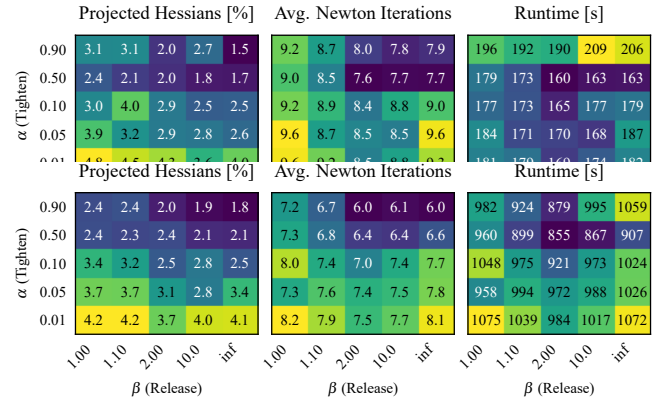
**Figure 4:** Ablation test for the parametrization of PPN in the Press scene using the iterative PCG linear solver (top) and the direct LLT solver (bottom). Color scale is independent per table.

( $\alpha = 0.01, \beta = 1.0$ ), result in excessive projection and increased Newton iterations, which is also slow. We adopt a power-of-two scaling ( $\alpha = 0.5$  and  $\beta = 2.0$ ) as our standard parametrization, owing to its optimal or near-optimal performance. The method exhibits low sensitivity around this optimum, allowing us to apply these parameters across all scenes without individual fine-tuning.

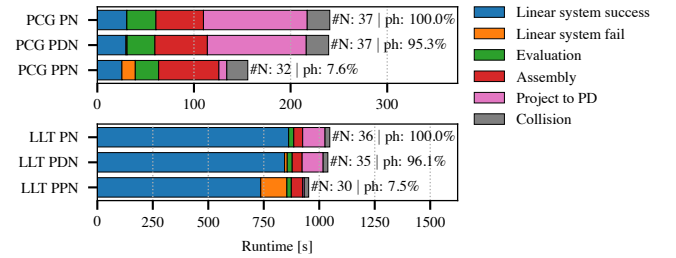
Next, we compare PPN with PN and PDN, using both PCG and LLT, in Fig. 6. We observe that PDN largely resorts to PN, revealing that only a few steps encountered zero global indefiniteness. Even in this stress conditions, the adaptive nature of PPN avoids over 92% of element projections and reduces Newton iterations by 14%. Runtime gains with LLT are modest (5.2%), as the direct solver dominates total cost, but with PCG, we observe a speedup of  $\times 1.5$  compared to PN and PDN. See Fig. 1 (d) for a visualization of these reductions over the time steps, and Fig. 1 (e) for the effect on negative eigenvalues of the three methods. All following experiments use PCG as the linear solver.

Eigenvalue mirroring [CLL\*24] was originally introduced specifically for quasistatic problems where strong indefiniteness is not counteracted by e.g. the mass matrix. For completeness, we apply mirroring to the “Press” experiment for all three solvers, however, results are poor in this elastodynamics with contact simulation: Mirroring consistently performs worse than clamping, needing an average of 51, 52, and 45 Newton iterations for PN, PDN, and PPN respectively, an increase of about 40% across all solvers. Based on this result, we apply clamping for all further dynamic experiments if not otherwise specified.

**Resolution, Time Step Size and Tolerance** Fig. 7 compares all solvers on the “Press” scene across different resolutions (2k, 15k, 108k degrees of freedom), time step sizes (100, 10, 1 ms) and tolerances ( $10^{-2}, 10^{-3}, 10^{-4} \text{ ms}^{-1}$ ). PPN solves all instances by projecting only a fraction of the elements (between 30% and 5%), correlating positively with finer resolutions, smaller time steps, and tighter tolerances: Finer resolutions localize sources of indefiniteness more effectively, smaller time steps magnify the regularizing effect of the mass matrix, and stricter tolerances extend the Newton iteration sequence with steps where most of



**Figure 5:** Ablation test for the parametrization of PPN in the Armadillo Slingshot scene using the iterative PCG linear solver (top) and the direct LLT solver (bottom). Color scale is independent per table.

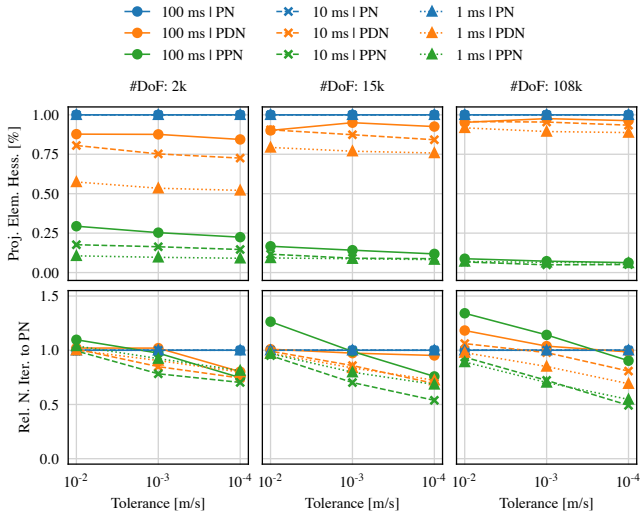


**Figure 6:** Runtime breakdown for the chosen parametrization of PPN in the “Press” scene using PCG (top) and LLT (bottom) linear solvers. The average number of Newton iterations (#N) and the percentage of projected element Hessians (ph) are shown at the right of each bar.

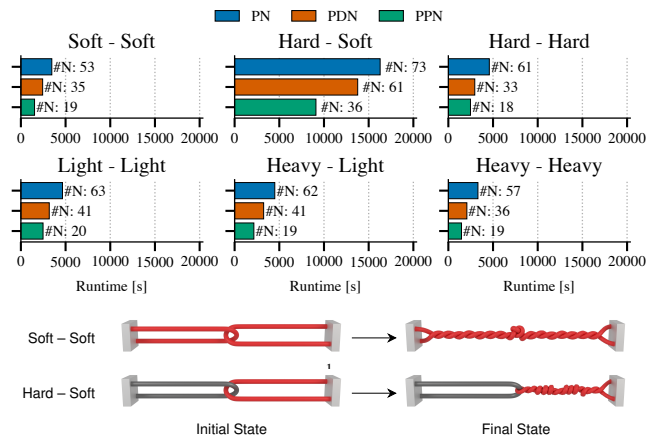
the domain has locally converged. PPN also achieves consistently fewer Newton iterations and lower runtimes, with the exception of the very large time step of  $\Delta t = 100 \text{ ms}$ , where both PDN and PPN struggle. Nevertheless, PPN performs strongly, completing the entire benchmark using 27.3% fewer Newton iterations than PN and offering a speedup of  $\times 1.93$ . In contrast, PDN takes 7.7% fewer Newton iterations and is just 7% faster when compared to PN.

**Quasistatic Simulation** We compare the three solvers using eigenvalue clamping and mirroring on a quasistatic problem involving a large initial deformation for various resolutions and Poisson ratios (Fig. 8). We use a Newton step stopping criteria of 0.1% of the domain’s size. In line with Chen et al. [CLL\*24], we reproduce the positive outcomes of eigenvalue mirroring for such scenarios while clamping produces artifacts. As suggested by the previous experiment, PDN and PPN face challenges in this inertia-free setting, indicating that unconditional projection might be preferred in this setting.

**Large Ratios** We compare all three solvers in simulations featuring large stiffness ( $E = 10^6\text{--}10^{10} \text{ Pa}$ ) and density ( $\rho = 10^1\text{--}$



**Figure 7:** Number of projected Hessians (top) and Newton iterations (bottom) in relation to PN for all solvers in different parametrizations of the “Press” scene: mesh resolution (columns), time step size (marker shape), solver type (color) and tolerance (x-axis).



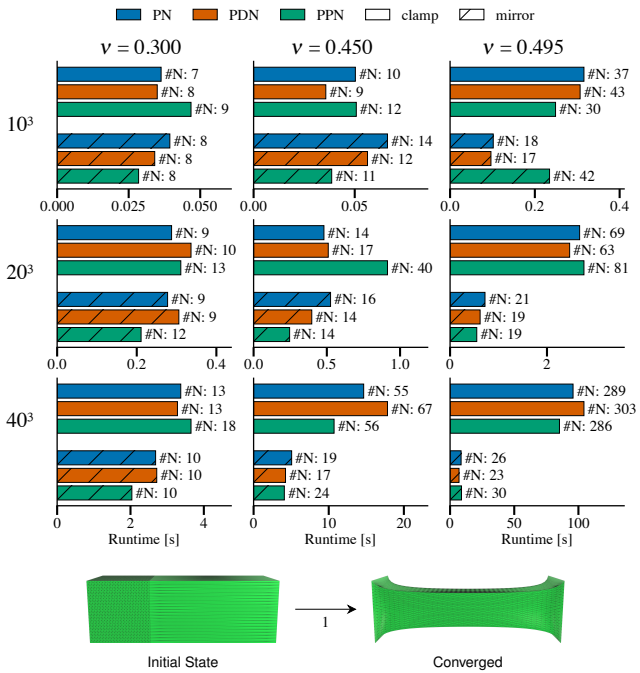
**Figure 9: U-Turn.** Two elastic cylinders are interlaced and twisted. Above, we compare solvers under varying Young’s moduli (top) and densities (bottom). Below, the initial and final states are shown. All simulations except the “Hard-Soft” produce the same deformation.

10<sup>4</sup> kg m<sup>-3</sup>) ratios in Fig. 9. Here, PPN consistently yields both the fewest Newton iterations and the fastest runtimes. While on average PPN projects 6.2% of the element Hessians, PDN projects 88.4%. On average, PPN requires only 35.6% and 54.5% of the Newton iterations of PN and PDN, respectively, corresponding to speedups of ×2.02 and ×1.42. These findings suggest that the residual-based heuristic in PPN remains robust even when adjacent elements exhibit curvature variations spanning several orders of magnitude.

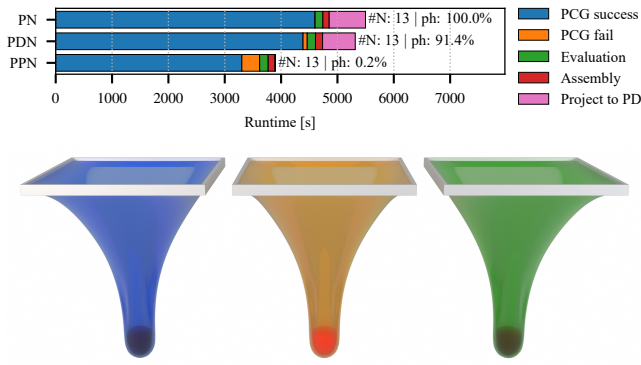
Finally, we quantify geometrical reproducibility across the three methods using a simulation characterized by large strains and extreme density ratios ( $\rho = 10^1\text{--}10^4 \text{ kg m}^{-3}$ ), as shown in Fig. 10. The mean area-weighted surface distance over the full time sequence is just 0.34 mm (0.057% of the domain size), with a maximum deviation of 1.04%. This confirms that the choice of projection strategy does not compromise the fidelity of the outcome, yet PPN delivers speedups of ×1.39 and ×1.35 over PN and PDN, respectively.

**Contact-free** The three solvers are compared in a contact-free simulation of an elastic armadillo in Fig. 11. Even in this simpler setting, PDN ends up projecting more than 50% of all the element Hessians, while PPN only needs to project less than 3%. PPN reduces the Newton iterations by 53% and 20% with respect to PN and PDN, demonstrating that PPN’s effectiveness is not exclusive to scenarios with complex frictional contact. Corresponding speedups are ×2.5 and ×1.34.

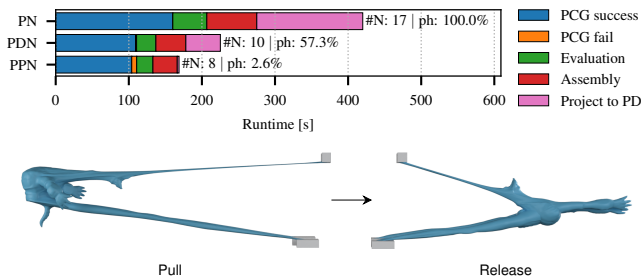
**Codimensional** We compare the three solvers in a contact-rich cloth simulation using three resolutions with vertex counts of 64<sup>2</sup>, 128<sup>2</sup> and 256<sup>2</sup> in Fig. 12. In this challenging scenario, PPN provides significant improvements for the coarsest mesh: a reduction of 45.8% and 30.7% Newton iterations, and 55.5% and 41.7% of runtime in relation to PN and PDN, respectively. For the finest discretization, PPN does not reduce iterations but still achieves more than 17% performance improvement over the alternatives.



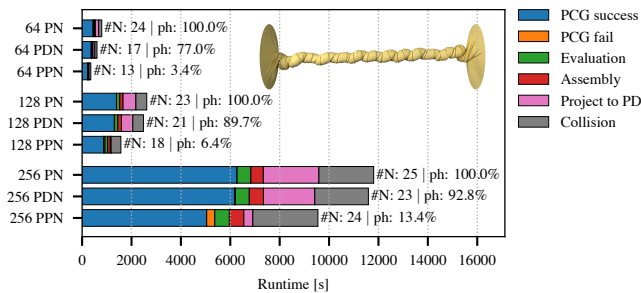
**Figure 8: Quasistatic extrusion.** An elastic box is stretched by twice its size in a quasistatic setting using different resolutions (rows) and Poisson’s ratios (columns). Solvers use different colors, and eigenvalue filtering is different shading: solid for clamping and hatched for mirroring. The number of Newton iterations is shown at the right of each bar.



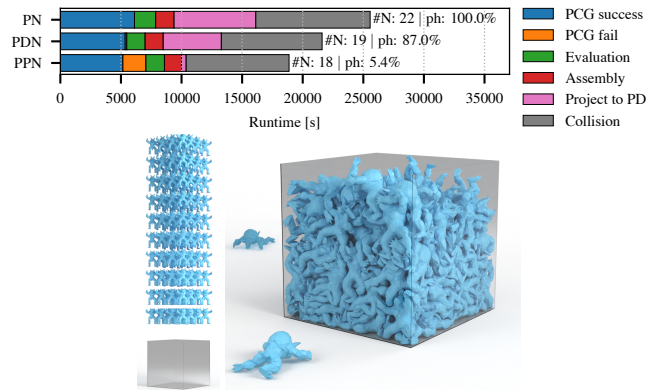
**Figure 10: Jelly Catch.** A heavy elastic ball falls on a thin sheet of very light elastic material clamped on its perimeter, resulting in very large strain in the material. Top: Comparison between the solvers. Bottom: Time step of maximum strain for PN, PDN and PPN, respectively from left to right.



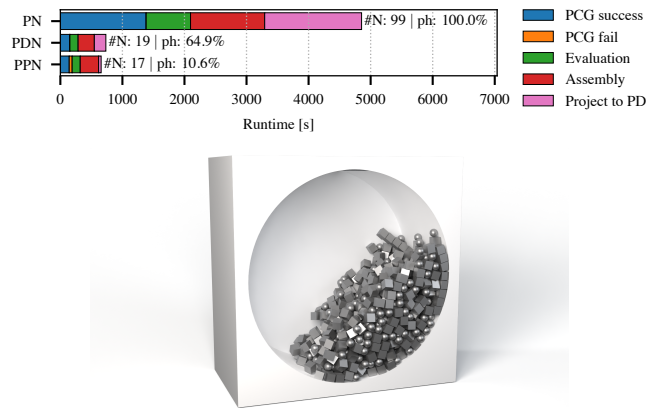
**Figure 11: Armadillo slingshot.** An elastic armadillo is pulled and then released. Comparison between solvers on top, largest deformation states below.



**Figure 12: Twisting cloth.** A cloth cylinder is twisted by rotating its ends in opposite directions using three mesh resolutions. Final configuration of the finest resolution shown inside the plot.



**Figure 13: Armadillo drop.** 160 elastic armadillos are dropped into a rigid box. Comparison between solvers on top. Initial and final state below.



**Figure 14: Tumbler.** More than 1000 rigid bodies collide inside a spinning tumbler. Comparison between solver on top. Collision runtime is omitted for clarity, as it dominates total cost. An intermediate state is shown below.

**Impact-rich** We test two scenes featuring high-energy impacts in Fig. 13 and 14. The former simulates elastic objects and the latter rigid bodies. To preserve vividness, these simulations use a time step of 1/300s, as larger time steps resulted in visibly damped dynamics. Although PPN still greatly reduces the number of projections, its advantage over PDN in terms of Newton iterations is more modest. Nevertheless, runtime was reduced by 26.2% and 12.6% in relation to PN and PDN for elastic scene, and by 86.5% and 10.4% for the rigid body one. Notably, PN struggles significantly in the rigid body scene, requiring more than five times as many iterations and representing an outlier in our tests.

**6. Limitations And Future Work**

While PPN reliably reduces element projections and Newton iterations in any simulator, the net speedup can vary depending on solver choices (e.g., direct vs. iterative) and execution platforms (e.g., CPU vs. GPU). Codebases that incorporate analytic projections may see lower gains than those relying on numerical

**Table 1:** Scene parameters. All simulation use 0.5 mm IPC contact distance.

Scene	$n_{\text{dof}}$	$\Delta t$ [s]	Dimension [m]	Duration [s]	Material ( $E, \nu$ )	Density
Rolling sphere	3K	1/30	0.15	12	$1 \times 10^3$ Pa, 0.49	$1000 \text{ kg m}^{-3}$
Press	47K	1/30	0.30	5	$1 \times 10^5$ Pa, 0.40	$1000 \text{ kg m}^{-3}$
Quasistatic extrusion	variable	$\infty$	0.50	–	$1 \times 10^8$ Pa, 0.49	–
U-Turn	87K	1/30	1.85	12	variable, 0.49	variable
Jelly Catch	570K	1/30	0.5	10	variable, 0.49	variable
Armadillo slingshot	71K	1/30	1.00	15	$1 \times 10^5$ Pa, 0.40	$1000 \text{ kg m}^{-3}$
Twisting cloth	variable	1/30	0.50, 0.001 thick	20	$1 \times 10^5$ Pa, 0.30	$0.20 \text{ kg m}^{-2}$
Armadillo drop	566K	1/300	0.35	7.5	$1 \times 10^4$ Pa, 0.45	$1000 \text{ kg m}^{-3}$
Tumbler	7K	1/300	0.50	15	rigid	$1000 \text{ kg m}^{-3}$

eigendecompositions, and applications using a fixed number of Newton iterations may only benefit from the reduced projections.

Our results also indicate that current on-demand projection strategies, including PDN and PPN, perform poorly in scenarios lacking strong mass matrix contributions (or alternative forms of regularization), as in very large time steps or in the quasistatic limit. However, the time steps where this becomes a problem are rare in practice due to the associated numerical damping and loss of detail. In any case, for realistic choices of time step sizes, including relatively large ones such as 1/30s PPN consistently excels.

## 7. Conclusion

We introduced Progressively Projected Newton, a direct replacement for Projected Newton in dynamic simulations that guarantees descent directions while reducing element projections by an order of magnitude and improving convergence. PPN begins each Newton iteration with the unmodified Hessian and only projects elements incrementally when the linear solver detects indefiniteness, guided by a residual-driven tolerance that adapts across iterations.

Extensive experiments on dynamic simulations of deformable solids, shells, frictional contact, and rigid bodies demonstrate that PPN consistently performs 90% fewer projections, decreases the number of Newton iterations by up to 50% compared to PN, and achieves total simulation speedups of up to  $\times 2.5$  relative to PN and up to  $\times 1.5$  over PDN. Hence, PPN stands out as the most efficient method among the three for robustly handling indefiniteness in second-order optimization time integration.

## Acknowledgements

This work is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) — project number 281466253. Open Access funding enabled and organized by Projekt DEAL.

## References

[BML\*14] BOUAZIZ S., MARTIN S., LIU T., KAVAN L., PAULY M.: Projective dynamics: fusing constraint projections for fast simulation.

© 2026 The Author(s).

Computer Graphics Forum published by Eurographics and John Wiley & Sons Ltd.

*ACM Transactions on Graphics* 33, 4 (July 2014), 154:1–154:11. doi:10.1145/2601097.2601116. 2

[CLF25] CHENG Y., LIU L., FU X.: Eigenvalue blending for projected newton. *Computer Graphics Forum* (Apr. 2025). doi:10.1111/cgf.70027. 3

[CLJ\*24] CHEN H., LIU H.-T. D., JACOBSON A., LEVIN D. I., ZHENG C.: Trust-region eigenvalue filtering for projected newton. In *SIGGRAPH Asia 2024 Conference Papers* (Dec. 2024), SA '24, Association for Computing Machinery. doi:10.1145/3680528.3687650. 3

[CLL\*24] CHEN H., LIU H.-T. D., LEVIN D. I., ZHENG C., JACOBSON A.: Stabler neo-hookean simulation: Absolute eigenvalue filtering for projected newton. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers '24* (July 2024), SIGGRAPH '24, ACM, pp. 1–10. doi:10.1145/3641519.3657433. 3, 6

[CLYY24] CHEN A. H., LIU Z., YANG Y., YUKSEL C.: Vertex block descent. *ACM Transactions on Graphics* 43, 4 (July 2024), 1–16. doi:10.1145/3658179. 2

[CXY\*23] CHEN Y., XIE T., YUKSEL C., KAUFMAN D., YANG Y., JIANG C., LI M.: Multi-Layer Thick Shells. In *ACM SIGGRAPH 2023 Conference Proceedings* (July 2023), SIGGRAPH '23, Association for Computing Machinery, pp. 1–9. URL: <https://dl.acm.org/doi/10.1145/3588432.3591489>, doi:10.1145/3588432.3591489. 2

[FLL\*24] FERNÁNDEZ-FERNÁNDEZ J. A., LANGE R., LAIBLE S., ARRAS K. O., BENDER J.: Stark: A unified framework for strongly coupled simulation of rigid and deformable bodies with frictional contact. In *2024 IEEE International Conference on Robotics and Automation (ICRA)* (2024). doi:10.1109/ICRA57147.2024.10610574. 5

[FFLW\*25] FERNÁNDEZ-FERNÁNDEZ J., LÖSCHNER F., WESTHOFEN L., LONGVA A., BENDER J.: Symx: Energy-based simulation from symbolic expressions. *ACM Trans. Graph.* (Sept. 2025). Just Accepted. URL: <https://doi.org/10.1145/3764928>, doi:10.1145/3764928. 3

[FL16] FU X.-M., LIU Y.: Computing inversion-free mappings by simplex assembly. *ACM Transactions on Graphics* 35, 6 (Nov. 2016), 1–12. doi:10.1145/2980179.2980231. 3

[FLS\*21] FERGUSON Z., LI M., SCHNEIDER T., GIL-URETA F., LANGLOIS T., JIANG C., ZORIN D., KAUFMAN D. M., PANOZZO D.: Intersection-free rigid body dynamics. *ACM Transactions on Graphics* 40, 4 (July 2021), 183:1–183:16. doi:10.1145/3450626.3459802. 2

[GSS\*15] GAST T. F., SCHROEDER C., STOMAKHIN A., JIANG C., TERAN J. M.: Optimization integrator for large time steps. *IEEE*

- Transactions on Visualization and Computer Graphics* 21, 10 (Oct. 2015), 1103–1115. doi:10.1109/tvcg.2015.2459687. 2, 3
- [HCLK24] HUANG K., CHITALU F. M., LIN H., KOMURA T.: Gipc: Fast and stable gauss-newton optimization of ipc barrier energy. *ACM Transactions on Graphics* 43, 2 (Mar. 2024), 1–18. doi:10.1145/3643028. 2
- [HJL\*25] HOLZ D., JESKE S. R., LÖSCHNER F., BENDER J., YANG Y., ANDREWS S.: Multiphysics simulation methods in computer graphics. *Computer Graphics Forum* (2025). doi:10.1111/cgf.70082. 2
- [HSK24] HERHOLZ P., STUYCK T., KAVAN L.: A mesh-based simulation framework using automatic code generation. *ACM Transactions on Graphics* 43, 6 (Nov. 2024), 1–17. doi:10.1145/3687986. 3
- [KDG19] KIM T., DE GOES F., IBEN H.: Anisotropic elasticity for inversion-safety and element rehabilitation. *ACM Transactions on Graphics* 38, 4 (July 2019), 1–15. doi:10.1145/3306346.3323014. 2
- [KE22] KIM T., EBERLE D.: Dynamic deformables: implementation and production practicalities (now with code!). In *ACM SIGGRAPH 2022 Courses* (Aug. 2022), SIGGRAPH '22, ACM, pp. 1–259. doi:10.1145/3532720.3535628. 3
- [Kim20] KIM T.: A finite element formulation of baraff-witkin cloth. *Computer Graphics Forum* 39, 8 (Nov. 2020), 171–179. doi:10.1111/cgf.14111. 2
- [KYT\*06] KHAREVYCH L., YANG W., TONG Y., KANSO E., MARSDEN J. E., SCHRÖDER P., DESBRUN M.: Geometric, variational integrators for computer animation. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2006), SCA '06, p. 43–51. doi:10.2312/SCA/SCA06/043-051. 2
- [LBK17] LIU T., BOUAZIZ S., KAVAN L.: Quasi-Newton Methods for Real-Time Simulation of Hyperelastic Materials. *ACM Transactions on Graphics* 36, 4 (July 2017), 116a:1. doi:10.1145/3072959.2990496. 2
- [LBOK13] LIU T., BARGTEIL A. W., O'BRIEN J. F., KAVAN L.: Fast simulation of mass-spring systems. *ACM Transactions on Graphics* 32, 6 (Nov. 2013), 1–7. doi:10.1145/2508363.2508406. 2
- [LCK22] LIN H., CHITALU F. M., KOMURA T.: Isotropic arap energy using cauchy-green invariants. *ACM Transactions on Graphics* 41, 6 (Nov. 2022), 1–14. doi:10.1145/3550454.3555507. 2
- [LFFJ\*23] LÖSCHNER F., FERNÁNDEZ-FERNÁNDEZ J. A., JESKE S. R., LONGVA A., BENDER J.: Micropolar elasticity in physically-based animation. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 6, 3 (Aug. 2023). URL: <https://doi.org/10.1145/3606922>, doi:10.1145/3606922. 2
- [LFFJB24] LÖSCHNER F., FERNÁNDEZ-FERNÁNDEZ J. A., JESKE S. R., BENDER J.: Curved three-director cosserat shells with strong coupling. *Computer Graphics Forum* 43, 8 (2024). doi:10.1111/cgf.15183. 2
- [LFS\*20] LI M., FERGUSON Z., SCHNEIDER T., LANGLOIS T., ZORIN D., PANOZZO D., JIANG C., KAUFMAN D. M.: Incremental potential contact: intersection-and inversion-free, large-deformation dynamics. *ACM Transactions on Graphics* 39, 4 (Aug. 2020). doi:10.1145/3386569.3392425. 2, 5
- [LKJ21] LI M., KAUFMAN D. M., JIANG C.: Codimensional incremental potential contact. *ACM Transactions on Graphics* 40, 4 (July 2021), 1–24. doi:10.1145/3450626.3459767. 2
- [LKL\*22] LAN L., KAUFMAN D. M., LI M., JIANG C., YANG Y.: Affine body dynamics: fast, stable and intersection-free simulation of stiff materials. *ACM Transactions on Graphics* 41, 4 (July 2022), 67:1–67:14. URL: <https://dl.acm.org/doi/10.1145/3528223>. 3530064, doi:10.1145/3528223.3530064. 2
- [LLFF\*23] LONGVA A., LÖSCHNER F., FERNÁNDEZ-FERNÁNDEZ J. A., LARIONOV E., ASCHER U. M., BENDER J.: Pitfalls of projection: A study of newton-type solvers for incremental potentials, 2023. arXiv Preprint. arXiv:2311.14526. 3
- [LLH\*24] LI X., LI M., HAN X., WANG H., YANG Y., JIANG C.: A dynamic duo of finite elements and material points. In *ACM SIGGRAPH 2024 Conference Papers* (2024), SIGGRAPH '24, Association for Computing Machinery. URL: <https://doi.org/10.1145/3641519.3657449>, doi:10.1145/3641519.3657449. 2
- [MEM\*20] MACKLIN M., ERLEBEN K., MÜLLER M., CHENTANEZ N., JESCHKE S., KIM T.-Y.: Primal/dual descent methods for dynamics. In *Computer Graphics Forum* (2020), vol. 39, pp. 89–100. doi:10.1111/cgf.14104. 2, 5
- [MTGG11] MARTIN S., THOMASZEWSKI B., GRINSPUN E., GROSS M.: Example-based elastic materials. *ACM Transactions on Graphics* 30, 4 (July 2011), 1–8. doi:10.1145/2010324.1964967. 2
- [NW06] NOCEDAL J., WRIGHT S.: *Numerical optimization*. Springer Science & Business Media, 2006. doi:10.1007/978-0-387-40065-5. 2, 3, 5
- [OBLN17] OVERBY M., BROWN G. E., LI J., NARAIN R.: Admm  $\supseteq$  projective dynamics: Fast simulation of hyperelastic models with dynamic constraints. *IEEE Transactions on Visualization and Computer Graphics* 23, 10 (Oct 2017), 2222–2234. doi:10.1109/TVCG.2017.2730875. 2
- [RO99] RADOVITZKY R., ORTIZ M.: Error estimation and adaptive meshing in strongly nonlinear dynamic problems. *Computer Methods in Applied Mechanics and Engineering* 172, 1–4 (Apr. 1999), 203–240. doi:10.1016/s0045-7825(98)00230-8. 2
- [SBB\*22] SCHMIDT P., BORN J., BOMMES D., CAMPEN M., KOBBELT L.: Tinyad: Automatic differentiation in geometry processing made simple. *Computer Graphics Forum* 41, 5 (2022), 113–124. doi: <https://doi.org/10.1111/cgf.14607>. 3
- [SGK18] SMITH B., GOES F. D., KIM T.: Stable neo-hookean flesh simulation. *ACM Transactions on Graphics* 37, 2 (Mar. 2018), 1–15. doi:10.1145/3180491. 2, 5
- [SGK19] SMITH B., GOES F. D., KIM T.: Analytic eigensystems for isotropic distortion energies. *ACM Transactions on Graphics* 38, 1 (Feb. 2019), 1–15. doi:10.1145/3241041. 2
- [SK23] SHI A., KIM T.: A unified analysis of penalty-based collision energies. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 6, 3 (Aug. 2023), 1–19. doi:10.1145/3606934. 2
- [SPSH\*17] SHTEINGEL A., PORANNE R., SORKINE-HORNUNG O., KOVALSKY S. Z., LIPMAN Y.: Geometric optimization via composite majorization. *ACM Transactions on Graphics* 36, 4 (July 2017), 1–11. doi:10.1145/3072959.3073618. 2
- [TSIF05] TERAN J., SIFAKIS E., IRVING G., FEDKIW R.: Robust quasistatic finite elements and flesh simulation. SCA '05, Association for Computing Machinery, p. 181–190. doi:10.2312/SCA/SCA05/181-190. 2, 4
- [WK23] WU H., KIM T.: An eigenanalysis of angle-based deformation energies. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 6, 3 (Aug. 2023), 1–19. doi:10.1145/3606929. 2
- [WY16] WANG H., YANG Y.: Descent methods for elastic body simulation on the gpu. *ACM Transactions on Graphics* 35, 6 (Dec. 2016). doi:10.1145/2980179.2980236. 2
- [WYW23] WANG Z., YANG Y., WANG H.: Stable discrete bending by analytic eigensystem and adaptive orthotropic geometric stiffness. *ACM Transactions on Graphics* 42, 6 (Dec. 2023), 1–16. doi:10.1145/3618372. 2
- [XLYJ23] XIE T., LI M., YANG Y., JIANG C.: A Contact Proxy Splitting Method for Lagrangian Solid-Fluid Coupling. *ACM Transactions on Graphics* 42, 4 (July 2023), 122:1–122:14. URL: <https://dl.acm.org/doi/10.1145/3592115>, doi:10.1145/3592115. 2