Mixed Variational Finite Elements for Implicit Simulation of Deformables

TY TRUSTY, University of Toronto, Canada
DANNY M. KAUFMAN, Adobe Research, USA
DAVID I.W LEVIN, University of Toronto, Canada

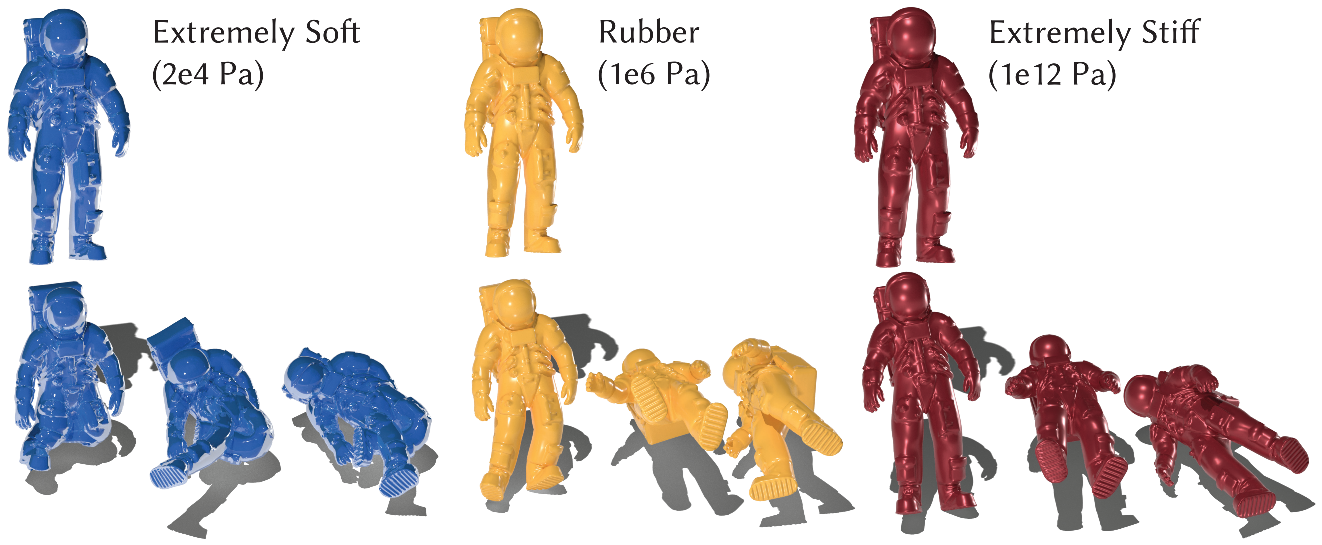

Fig. 1. Our new mixed finite element method (MFEM) can produce simulations of elastica with wildly different materials (including rigid) both accurately and quickly. Our key contribution is that our method is both capable of converging to an accurate solution, matching that of a Newton's method, as well as generate visually plausible results when stopped early. This makes it ideal for a plethora of engineering and graphics applications.

We propose and explore a new method for the implicit time integration of elastica. Key to our approach is the use of a mixed variational principle. In turn, its finite element discretization leads to an efficient and accurate sequential quadratic programming solver with a superset of the desirable properties of many previous integration strategies. This framework fits a range of elastic constitutive models and remains stable across a wide span of time step sizes and material parameters (including problems that are approximately rigid). Our method exhibits convergence on par with full Newton type solvers and also generates visually plausible results in just a few iterations comparable to recent fast simulation methods that do not converge. These properties make it suitable for both offline accurate simulation and performant applications with expressive physics. We demonstrate the efficacy of our approach on a number of simulated examples.

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

Additional Key Words and Phrases: physics-based animation, physics simulation

ACM Reference Format:
Ty Trusty, Danny M. Kaufman, and David I.W Levin. 2022. Mixed Variational Finite Elements for Implicit Simulation of Deformables. In *SIGGRAPH Asia 2022 Conference Papers (SA '22 Conference Papers), December 6–9, 2022, Daegu, Republic of Korea.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3550469.3555418

SA '22 Conference Papers, December 6–9, 2022, Daegu, Republic of Korea
2022. ACM ISBN 978-1-4503-9470-3/22/12…$15.00
https://doi.org/10.1145/3550469.3555418

1 INTRODUCTION

In this paper we explore the use of a mixed variational principle to build an efficient and general-purpose simulation algorithm for the physics-based animation of elastica.

Standard approaches for the implicit time integration of continua discretize with finite differences in time and finite elements in space. Recent methods often leverage the observation that, for these implicit time integration choices, each individual time step solve can then be cast as a minimization problem. In turn, the applied strategy for solving these optimization problems then leads to a wide range of well-known simulation algorithms [Li et al. 2019]. For example, a "standard" finite element approach involves minimizing an implicit integration energy via Newton's method while solving the bottleneck of inner linear-systems solves either via direct or iterative methods. Extended Position-Based Dynamics replaces standard direct or iterative solvers with iterations (e.g., GS, Jacobi, and/or SOR) acting on the dual variables (constraint forces) while Projective Dynamics and its more recent generalizations apply various forms of ADMM-type solvers to split, augmented Lagrangian forms.

Despite their common variational origin, implicit solvers for elastica exhibit a wide range of features and limitations, and so tradeoffs.

Standard Newton-type approaches and ADMM-based methods (including Projective Dynamics) exhibit various difficulties simulating stiff materials. On the other hand, Position-Based Dynamics, while fast and stable (even at high stiffness), lacks a direct correspondence to a discretization model and underlying PDE – it is then often unclear how to convert general material moduli and models to fit its constraint-based formulation. Furthermore some algorithms deliver visually compelling results early in their iteration sequences, making them ideal for interactive applications. However these same methods are generally unable to converge [Li et al. 2019] making them unsuitable for applications requiring accurate simulation solutions. In stark counterpoint gold-standard Newton-type methods generally converge well but are slow to do so. Likewise their early iterations are generally polluted by artifacts so that they are often unsuitable for fast applications. This bifurcated feature landscape means that practitioners of physical simulation and physics-based animation require a quiver of solvers to handle the multitude of materials, time steps and varying accuracy and speed requirements one might encounter.

Arguably an implicit time integrator for physics-based animation should

- support a full range of elastic constitutive models;
- be robust to large deformations;
- offer stability for a wide range of materials including those stiff enough to be effectively rigid;
- remain stable for large, frame-rate sized time steps;
- provide efficient solutions of every time step;
- produce visually plausible results at interactive or near-interactive rates; and
- given a practical time frame, converge to a consistent and accurate solution.

Existing popular methods, as covered briefly here and below, are often custom-specialized to a small subset of these properties and cases. In this paper we show that a mixed variational finite-element model, coupled with an appropriate optimization algorithm yields a method that aims to cover the full spread of the above target properties. In turn this provides a solver that is at home across the range of both interactive and slower applications for deformable-object simulation. At the same time, this solver is also able to converge to the same solutions that would be obtained by a converging Newton's method solver.

2 RELATED WORK

Implicit time integrators, especially backwards Euler, are ubiquitous tools in simulating elastica. In computer graphics, implicit integration steps are often solved via Newton's method [Terzopoulos and Qin 1994] or else via a single-iteration, linearly implicit approximation [Baraff and Witkin 1998]. A range of applied forces can, in turn, be derived variationally from distortion metrics [Terzopoulos and Qin 1994] and mechanical conditions on the simulated objects [Baraff and Witkin 1998]. More recent approaches often derive and implement implicit time integration from a variational perspective [Gast et al. 2015; Hahn et al. 2012; Hairer and Lubich 2014; Martin et al. 2011] which, once fully discretized in time (via finite differences), and space (via finite elements), yields a nonlinear minimization which is solved to update each forward time step in a simulation. Efficient, robust, and accurate solution of this optimization problem now lies at the heart of recent, physics-based animation research.

A default solution approach is to apply Newton's method with a direct solver to handle the resulting per-iteration sequence of linear systems solves. In turn iterative linear solvers [Smith et al. 2018; Wang and Yang 2016; Xian et al. 2019] can offer better linear solve performance and memory usage with the trade-off of overall slowing convergence for stiffer material models. Avoiding the direct solve via Quasi-Newton approaches can also lead to performance improvements [Liu et al. 2017]. An exceedingly efficient alternative approach is to apply primal-dual solver strategies which model elastic forces as constraints. Resulting dual optimization problems are then (approximately) solved to compute each forward time step. An extremely successful and effective application of this strategy is the Position Based Dynamics (PBD) algorithm [Muller et al. 2007] which acts on compliant constraints (similar to the condition energies from Baraff and Witkin [1998]), with fast, local iterative updates to generate approximating solutions for the dynamics. Extended Position Based Dynamics (XPBD) extends PBD with a quadratic compliant formulation [Servin et al. 2006] to establish a relationship between PBD constraints and some material models [Macklin et al. 2016] and enables the simulation of both rigid and quasi-rigid bodies [Müller et al. 2020]. However the local nature of the PBD updates, and its constitutive behavior dependence on iteration tuning often necessitates additional complexity in achieving desired results in high resolution meshes and large domains [Müller et al. 2017]. XPBD type solves also arise as intermediate steps in non-smooth Newton's Methods for contact and friction [Macklin et al. 2019].

Projective Dynamics (PD) [Bouaziz et al. 2014] applies an ADMM-type algorithm [Overby et al. 2017] to minimize the integration energy of a subset of deformation energies. This strategy has since been fully generalized by Overby et al. [2017] to a full ADMM model covering a complete range of hyperelastic energies. These methods incorporate a highly effective global projection step which helps balance locally solved forces across an entire meshed domain and lead to exceedingly stable behavior when simulating nonlinear elastica. However, this comes with a trade-off. When simulating nonlinear materials these ADMM strategies suffer from slow and even nonconvergence [Li et al. 2019]. This brings algorithmic parameter tuning questions and challenges when it comes to simulating stiffer materials. Brown and Narain [2021] use the splitting inherent in the ADMM formulation to introduce separate rotation and deformation variables (along with traditional positional degrees-of-freedom). While increasing the number of variables and the complexity of the global and local update steps, this approach shows improved convergence rates over previous methods – especially for examples exhibiting large rotational motions. Unfortunately, as we demonstrate, this does not alleviate ADMM's inability to converge to the FE solution, leaving these solvers inapplicable to accuracy-focused tasks.

In this work we present an algorithm for implicit integration that is convergent and consistent, while also able to produce compelling visual results in compute-time-limited scenarios such as interactive simulation. This solver can thus, unlike recent popular methods discussed, be used both in applications that require accuracy and applications that require fast visual updates. Our method has a host of additional features including being performant for a wide range of extreme material parameters (including effectively rigid bodies), robustness to large deformations, and is agnostic to material energy.

Our key insight is that, while recent work has rapidly explored improvements and tradeoffs at the solver level (and occasionally in choice of time-discretization), the underlying variational energies being optimization have largely remained the same. Rather, we set off from a mixed variational perspective to arrive at a new, efficient, accurate, sequential quadratic programming method [Wright and Nocedal 1999]. Mixed principles are not new to graphics, for instance the Hamilton-Pontryagin principle [Kharevych et al. 2006] has been applied to resolve position and displacement as independent discrete time-stepped variables coupled via constraints, in order to derive discrete variational time integrators. Here we are rather motivated by Reissner's [1985] approach, which decouples deformation and displacements. We, however, depart from this approeach to a new mixed variational principle design which naturally leads us to an efficient elastodynamics solver that exhibits a superset of features currently available in physics-based methods. These include convergence and consistency with a gold-standard Newton's methods, compatibility with general elastic constitutive models and stability for a wide range of time step sizes and material parameters. We demonstrate the accuracy of our method with comparisons to Newton's method and the state-of-the-art WRAPD [Brown and Narain 2021] solver, as well as demonstrate our methods efficacy on a number of examples simulated at interactive rates.

3 METHODS

3.1 Notation

Our mixed finite element time stepping scheme (Alg. 1) results from applying sequential quadratic programming to a mixed finite element variational form. Like much previous work, our algorithm involves both global and local updates, but it will demonstrate several advantages of other methods (e.g. ADMM or local-global splitting) that result in similar computational loops.

Going forward, we denote vectors and matrices respectively in bold lower and upper case, and scalars in lower case, for FE systems with n nodes and m elements.

3.2 Continuous Problem

Our central focus is the time integration of nonlinear elastodynamics. For a domain $\Omega \in \mathbb{R}^d$ with reference coordinates $\mathbf{X} \in \mathbb{R}^d$, and corresponding world-space positions $\mathbf{x}(\mathbf{X}) \in \mathbb{R}^d$, an elastodynamic system extremizes the action

$$S(\mathbf{X}) = \int_0^T \left(\int_\Omega \frac{\rho}{2} ||\dot{\mathbf{x}}(\mathbf{X})||^2 - \Psi(\mathbf{X}) + \mathbf{x}(\mathbf{X})^T \mathbf{f}_{\mathbf{ext}}(\mathbf{X}) \right) d\Omega dt, \quad (1)$$

where $\Psi(\mathbf{X})$ is the hyper-elastic deformation energy, $\mathbf{f}_{\mathbf{ext}}$ the external forces, and ρ density. We write each value as a function of $\mathbf{X}$ to emphasize these values spatially vary.

Discretizing in time, there are many potential integration methods, but we primarily focus for clarity on implicit Euler integration via the minimization of the incremental potential (IP) [Kane et al. 2000]. We define a finite element discretization with a set of elements $\mathcal{T} \in \Omega^d$ where from now on we assume elements to be tetrahedra and $d = 3$. Standard displacement-based FEM takes the nodal positions $\mathbf{x} \in \mathbb{R}^{3n}$ to be the degrees of freedom, and using the IP time integrator with positions, $\mathbf{x}^t$, and velocities, $\dot{\mathbf{x}}^t$, at time t, we find the nodal positions for the next time step by solving

$$\mathbf{x}^{t+1} = \arg\min_{\mathbf{x}} \frac{1}{2}(\mathbf{x} - \tilde{\mathbf{x}})^T M (\mathbf{x} - \tilde{\mathbf{x}}) + h^2 \sum_{K \in \mathcal{T}} \Psi_K(\mathbf{x}) w_K, \quad (2)$$

where h is the timsetep, M is the finite element mass matrix, $\tilde{\mathbf{x}} = \mathbf{x}^t + h\dot{\mathbf{x}}^t + h^2 M^{-1} \mathbf{f}_{\mathbf{ext}}$, and w_K is the per-element volume. This approach yields the standard implicit Euler, displacement-based Finite Element Method, but we depart from this in our mixed formulation.

Returning to the spatially continuous setting, we define a continuous mixed energy

$$U = \int_\Omega \Psi(S(\mathbf{X})) - \Lambda(\mathbf{X}) : (S(F(\mathbf{X})) - S(\mathbf{X})) d\Omega \quad (3)$$

where $F(\mathbf{X}) = \frac{\partial \mathbf{x}(\mathbf{X})}{\partial \mathbf{X}} \in \mathbb{R}^{3\times3}$ is the deformation gradient. $S(\mathbf{X}) \in \mathbb{R}^{3\times3}$ is a symmetric deformation matrix, and lastly $\Lambda(\mathbf{X}) \in \mathbb{R}^{3\times3}$ is a symmetric matrix of Lagrange multipliers. $S(F(\mathbf{X})) = R(\mathbf{X})^T F(\mathbf{X})$ is used to couple $S(\mathbf{X})$ and $F(\mathbf{X})$ via polar decomposition, and the Lagrange multipliers represent stresses enforcing the constraint. Measuring elastic deformation strictly with the symmetric $S(\mathbf{X})$ simplifies energy evaluation (hyperelastic constitutive models are easily defined in terms of S [Sifakis and Barbic 2012]), but the key benefit of this formulation is that rotations are implicitly included in the $S(F(\mathbf{X}))$ term, allowing our solver to track local rigid motions more efficiently, improving convergence [Brown and Narain 2021].

Algorithm 1: Mixed SQP Solver

```
Algorithm SimulationStep($\mathbf{x}^t, \dot{\mathbf{x}}^t, \mathbf{s}^t$, max_iters)
    $\mathbf{x} \leftarrow \mathbf{x}^t$, $\mathbf{s} \leftarrow \mathbf{s}^t$, $\lambda \leftarrow \mathbf{0}$
    iter $\leftarrow 0$
    do
        // Compute gradients and hessians
        $\mathbf{g}_\mathbf{x} \leftarrow$ nodal_derivatives($\mathbf{x}, \mathbf{x}^t, \dot{\mathbf{x}}^t$)
        $[G_\mathbf{x}, G_\mathbf{s}, \mathbf{c}] \leftarrow$ constraints($\mathbf{x}, \mathbf{s}$)          // Eq. (12)
        $[H_\mathbf{s}, \mathbf{g}_\mathbf{s}] \leftarrow$ deformation_derivatives($\mathbf{s}$)
        $[H_\lambda, \mathbf{g}_\lambda] \leftarrow$ lambda_derivatives          // Eq. (18)

        // Setup system and solve for $\delta\mathbf{x}$
        $[H, \mathbf{g}] \leftarrow$ assemble_system          // Eq. (20)
        $\delta\mathbf{x} = H^{-1}\mathbf{g}$

        // Local solves for $\lambda, \delta\mathbf{s}$
        $\lambda \leftarrow$ update_lambdas          // Eq. (19)
        $\delta\mathbf{s} \leftarrow$ update_s          // Eq. (15)
        $\mathbf{x}, \mathbf{s} \leftarrow$ backtracking linesearch
        iter $\leftarrow$ iter +1
    while not converged or iter < max_iters
    $\dot{\mathbf{x}} \leftarrow (\mathbf{x} - \mathbf{x}^t)/h$
    return $\mathbf{x}, \dot{\mathbf{x}}, \mathbf{s}, \lambda$
```

3.3 Discretized Mixed Energy

Now we return to the spatially discrete setting and construct a mixed potential on a finite element mesh. To simplify this discretization we make the assumption that the shape function for positions, $\mathbf{x}(\mathbf{X})$, are linear, making the deformation gradients, F, constant over each element. Consequently, we make $S(\mathbf{X})$ and $\Lambda(\mathbf{X})$ element-wise constant. Furthermore, we write $F_K(\mathbf{x})$ to denote the deformation gradient for the K-th element and that it is a function of the nodal degrees of freedom $\mathbf{x} \in \mathbb{R}^{3n}$. In this discrete setting, our mixed energy is

$$U \approx \tilde{U} = \sum_{K \in \mathcal{T}} \left(\Psi(S_K) - \Lambda_K : (S(F_K(\mathbf{x})) - S_K)\right) w_K, \tag{4}$$

where w_K is the K-th element's volume. Next considering that Λ_K and S_K are 3×3 symmetric matrices, we represent them each by 6×1 vectors λ_K and $\mathbf{s}_K$, respectively.

Using these simplifications, we can rewrite our discrete energy equivalently as

$$\tilde{U}(\mathbf{x}, \mathbf{s}, \lambda) = \sum_{K \in \mathcal{T}} \left(\Psi(\mathbf{s}_K) - \lambda_K^T \mathbf{c}_K(\mathbf{x}, \mathbf{s})\right) w_K, \tag{5}$$

where

$$\mathbf{c}_K(\mathbf{x}, \mathbf{s}) = \mathrm{Sym}(\mathbf{s}(F_K(\mathbf{x})) - \mathbf{s}_K) \quad \in \mathbb{R}^6, \tag{6}$$

and $\mathrm{Sym} = diag(\begin{bmatrix}1 & 1 & 1 & 2 & 2 & 2\end{bmatrix})$ is a 6×6 diagonal matrix that ensures $\Lambda_K : (S(F_K(\mathbf{x})) - S_K) = \lambda_K^T \mathrm{Sym}(\mathbf{s}(F_K(\mathbf{x})) - \mathbf{s}_K)$.

With our new discrete potential energy, we seek to build an implicit time integrator in the same fashion as the previously mentioned displacement-based FEM. We write our IP energy as the Lagrangian

$$\mathcal{L}(\mathbf{x}, \mathbf{s}, \lambda) = \frac{1}{2}||\mathbf{x} - \tilde{\mathbf{x}}||_M^2 + h^2 \tilde{U}(\mathbf{x}, \mathbf{s}, \lambda) \tag{7}$$

where $||\cdot||_M$ is the energy norm with the nodal mass matrix. With this Lagrangian, our IP-based implicit Euler update becomes the saddlepoint problem

$$\mathbf{x}^{t+1}, \mathbf{s}^{t+1}, \lambda^{t+1} = \arg\min_{\mathbf{x},\mathbf{s}} \left(\arg\max_{\lambda} \mathcal{L}(\mathbf{x}, \mathbf{s}, \lambda)\right) \tag{8}$$

where $\mathbf{x} \in \mathbb{R}^{3n}$, $\mathbf{s} \in \mathbb{R}^{6m}$, and $\lambda \in \mathbb{R}^{6m}$.

3.4 Sequential Quadratic Programming

The saddlepoint problem in (8) is equivalent to the equality-constrained minimization:

$$\begin{aligned} \min_{\mathbf{x},\mathbf{s}} \quad & f(\mathbf{x}, \mathbf{s}) \\ \text{s.t.} \quad & \mathbf{c}(\mathbf{x}, \mathbf{s}) = 0, \end{aligned} \tag{9}$$

where $f(\mathbf{x}, \mathbf{s}) = \frac{1}{2}||\mathbf{x} - \tilde{\mathbf{x}}||_M^2 + h^2 \Psi(\mathbf{s})$, and $\mathbf{c}(\mathbf{x}, \mathbf{s}) \in \mathbb{R}^{6m}$ consists of the $\mathbf{c}_K(\mathbf{x}, \mathbf{s})$ functions for all $K \in \mathcal{T}$.

Nonlinearity in this constraint arises from the $\mathbf{s}(F_K(\mathbf{x}))$ term, which extracts the symmetric components from the polar decomposition of the deformation gradient, F. This motivates our use of sequential quadratic programming (SQP), which generates steps by solving a sequence of quadratic subproblems, and addresses the nonlinearity of the constraints by linearly approximating them in each iteration. The generality of this approach gives us a broad range of options for numerical methods to solve this SQP, but we find that characteristics of our mixed formulation can be exploited to arrive at a particularly stable and easy-to-solve Newton-like method.

SQP can be understood simply as an application of Newton's method to the first-order KKT conditions of equality constrained problem in (9):

$$\begin{pmatrix} \nabla_{\mathbf{x}} f(\mathbf{x}, \mathbf{s}) + G_{\mathbf{x}}(\mathbf{x})^T \lambda \\ \nabla_{\mathbf{s}} f(\mathbf{x}, \mathbf{s}) + G_{\mathbf{s}}(\mathbf{s})^T \lambda \\ \mathbf{c}(\mathbf{x}, \mathbf{s}) \end{pmatrix} = \mathbf{0}, \tag{10}$$

where $G_{\mathbf{x}}(\mathbf{x}) \in \mathbb{R}^{6m \times 3n}$ and $G_{\mathbf{s}}(\mathbf{s}) \in \mathbb{R}^{6m \times 6m}$ are the constraint Jacobians:

$$G_{\mathbf{x}}(\mathbf{x}) = -\left[\nabla_{\mathbf{x}} \mathbf{c}_1(\mathbf{x}, \mathbf{s}), \nabla_{\mathbf{x}} \mathbf{c}_2(\mathbf{x}, \mathbf{s}), \cdots, \nabla_{\mathbf{x}} \mathbf{c}_m(\mathbf{x}, \mathbf{s})\right]^T \tag{11}$$

$$G_{\mathbf{s}}(\mathbf{s}) = -\left[\nabla_{\mathbf{s}} \mathbf{c}_1(\mathbf{x}, \mathbf{s}), \nabla_{\mathbf{s}} \mathbf{c}_2(\mathbf{x}, \mathbf{s}), \cdots, \nabla_{\mathbf{s}} \mathbf{c}_m(\mathbf{x}, \mathbf{s})\right]^T. \tag{12}$$

It is important to note that $G_{\mathbf{x}}$ has the same structure as the standard gradient operator in FEM, but with the addition of derivatives of $\mathbf{s}(F(\mathbf{x}))$ with respect to $\mathbf{x}$. These derivatives give our method a first order awareness of rotational motion which we demonstrate helps improve its convergence.

In SQP, Newton's method is typically used to solve (10), which in the k-th iteration gives us a linear system:

$$\begin{pmatrix} \nabla^2_{\mathbf{xx}} \mathcal{L} & 0 & G_{\mathbf{x}}^T \\ 0 & \nabla^2_{\mathbf{ss}} \mathcal{L} & G_{\mathbf{s}}^T \\ G_{\mathbf{x}} & G_{\mathbf{s}} & 0 \end{pmatrix} \begin{pmatrix} \delta \mathbf{x} \\ \delta \mathbf{s} \\ \lambda^{k+1} \end{pmatrix} = \begin{pmatrix} -\nabla_{\mathbf{x}} f \\ -\nabla_{\mathbf{s}} f \\ \mathbf{c}(\mathbf{x}^k, \mathbf{s}^k) \end{pmatrix}, \tag{13}$$

where all the above gradients and Jacobians are evaluated at $(\mathbf{x}^k, \mathbf{s}^k)$, and we have that $\mathbf{x}^{k+1} = \mathbf{x}^k + \delta \mathbf{x}$ and $\mathbf{s}^{k+1} = \mathbf{s}^k + \delta \mathbf{s}$.

3.5 A Positive Definite SQP Solver

Here the LHS in (13) is indefinite and difficult to solve, requiring specialized factorizations or iterative methods to solve [Cheshmi et al. 2020]. Instead, we reformulate our system into a symmetric positive definite (SPD) form, enabling us to apply stable and efficient SPD-based solvers.

First, to simplify the notation, let $\mathbf{g}_{\mathbf{s}} = \nabla_{\mathbf{s}} f$, $\mathbf{g}_{\mathbf{x}} = \nabla_{\mathbf{x}} f$, and $H_{\mathbf{s}} = \nabla^2_{\mathbf{ss}} \mathcal{L}$. The latter has block diagonal structure where the K-th block equals $\frac{\partial^2 \Psi(\mathbf{s}_K)}{\partial \mathbf{s}_K^2}$. Then, to improve computation we swap the mass matrix M for the $\nabla^2_{\mathbf{xx}} \mathcal{L}$ block, which amounts to neglecting the second derivative of the constraint function with respect to $\mathbf{x}$. This gives us the new linear system for the $(k+1)$-th iteration:

$$\begin{pmatrix} M & 0 & G_{\mathbf{x}}^T \\ 0 & H_{\mathbf{s}} & G_{\mathbf{s}}^T \\ G_{\mathbf{x}} & G_{\mathbf{s}} & 0 \end{pmatrix} \begin{pmatrix} \delta \mathbf{x} \\ \delta \mathbf{s} \\ \lambda^{k+1} \end{pmatrix} = \begin{pmatrix} -\mathbf{g}_{\mathbf{x}} \\ -\mathbf{g}_{\mathbf{s}} \\ \mathbf{c} \end{pmatrix}, \tag{14}$$

where $\mathbf{c} = \mathbf{c}(\mathbf{x}^k, \mathbf{s}^k)$, and we reiterate that all the derivatives are computed using variables from the k-th iteration.

To arrive at a reduced SPD matrix we perform two Schur complements to eliminate variables from the solve. This is a frequent strategy in mixed FEM formulations, and is commonly referred to as *static condensation* in the FEM community. Eliminating the $\delta \mathbf{s}$, and λ from (14) enables us to perform an SPD global solve for $\delta \mathbf{x}$, followed by parallel local solves for $\delta \mathbf{s}$ and λ. First, for the $\delta \mathbf{s}$ variables we have

$$\delta\mathbf{s} = -H_{\mathbf{s}}^{-1}(G_{\mathbf{s}}\lambda^{k+1} + \mathbf{g}_{\mathbf{s}}), \tag{15}$$

which we substitute into (14) to get

$$\begin{pmatrix} M & G_{\mathbf{x}}^T \\ G_{\mathbf{x}} & -G_{\mathbf{s}}H_{\mathbf{s}}^{-1}G_{\mathbf{s}} \end{pmatrix} \begin{pmatrix} \delta\mathbf{x} \\ \lambda^{k+1} \end{pmatrix} = \begin{pmatrix} -\mathbf{g}_{\mathbf{x}} \\ \mathbf{c}(\mathbf{x}^k, \mathbf{s}^k) + G_{\mathbf{s}}H_{\mathbf{s}}^{-1}\mathbf{g}_{\mathbf{s}} \end{pmatrix}, \tag{16}$$

and we reiterate that $G_{\mathbf{s}}$ is diagonal so we safely omit transposes. This system still has no guarantees of being SPD, so we perform another Schur complement to eliminate λ^{k+1}. To further simplify things we define

$$\mathbf{g}_\lambda = \mathbf{c}(\mathbf{x}^k, \mathbf{s}^k) + G_{\mathbf{s}}H_{\mathbf{s}}^{-1}\mathbf{g}_{\mathbf{s}}, \tag{17}$$

$$H_\lambda = (G_{\mathbf{s}}H_{\mathbf{s}}^{-1}G_{\mathbf{s}})^{-1} = G_{\mathbf{s}}^{-1}H_{\mathbf{s}}G_{\mathbf{s}}^{-1}, \tag{18}$$

so that the equation for λ^{k+1} becomes

$$\lambda^{k+1} = -H_\lambda(\mathbf{g}_\lambda - G_{\mathbf{x}}\delta\mathbf{x}). \tag{19}$$

Substituting into (16), we get an equation only in terms of $\delta\mathbf{x}$:

$$\underbrace{(M + G_{\mathbf{x}}^T H_\lambda G_{\mathbf{x}})}_{H}\,\delta\mathbf{x} = \underbrace{G_{\mathbf{x}}^T H_\lambda \mathbf{g}_\lambda - \mathbf{g}_{\mathbf{x}}}_{\mathbf{g}}. \tag{20}$$

This resulting system now directly solves for nodal DOF as in standard FEM, however, our stiffness matrix incorporates deformation Hessians as a function of the independent $\mathbf{s}$ variables. For hyperelastic material models, it is common for $H_{\mathbf{s}}$ to become indefinite, which we address in the standard way by applying a positive semi-definite (PSD) fix to the per-element Hessians [Wright and Nocedal 1999] to project them to PSD matrices. With these modifications we obtain nodal solutions to the FE elastodynamic problem with, as we will demonstrate, in many cases superior convergence and performance to directly solving via standard FE solved via a direct Newton solver.

3.6 The final algorithm

After we the solve for $\delta\mathbf{x}$ we perform the local updates for $\delta\mathbf{s}$ and λ^{k+1} (Eq. (19) and (15)). The asymptotic performance in our algorithm is dominated by the global $\delta\mathbf{x}$ variable solve, so the per-step cost is effectively equivalent to standard FEM solved via Newton's method. The variables $\delta\mathbf{x}$ and $\delta\mathbf{s}$ represent search directions for a single substep in our SQP problem, and the final $\mathbf{x}^{k+1}$ and $\mathbf{s}^{k+1}$ updates are computed with backtracking linesearch. See Alg. 1 for an outline of our algorithm. As in previous work, we make the performance of our algorithm scalable by solving Eq. 20 iteratively. Specifically, we exploit the positive definiteness of our system to employ the preconditioned conjugate gradient method, preconditioned with the ARAP-like Hessian from Liu et al. [2017]. We warm start the solver with an initial reduced direct solve performed in an affine deformation space [Lan et al. 2022]. By simply controlling the number of allowed iterations this provides controllable performance ranging from slow but convergent and consistent behavior to visually plausible, interactive time stepping.

4 RESULTS AND DISCUSSION

All our expreiments were performed on a desktop computer with an AMD Ryzen 7 5800X 8-Core Processor and 16GB of RAM. We implemented our method using Eigen [Guennebaud et al. 2010] for linear algebra routines, SuiteSparse for direct linear solves [Davis 2006], libigl [Jacobson et al. 2018] for geometry processing, Bartels [Levin 2020] for physics utility code and Polyscope [Sharp et al. 2019] for display. All our simulations are run with a timestep of 0.0333 seconds (30 fps). All timings are CPU only with the minimal optimization afforded by parallelization via OpenMP.

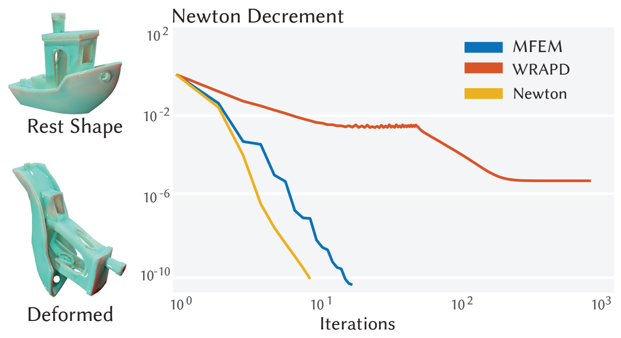

Fig. 2. Convergence of Newton's method, WRAPD and our solver (MFEM) on a single step. Error is measured using the Newton Decrement and both Newton's method and our method converge to within numerical error. WRAPD fails to converge even after a thousand iterations.

We begin by demonstrating the efficacy of our mixed FEM solver (MFEM) via comparison to both a standard, Newton-based implicit integrator (using backtracking line search) and the WRAPD algorithm [Brown and Narain 2021]. WRAPD uses an identical rotation/deformation splitting to our method but both derives it with and applies it to an ADMM-based optimization solver. In turn, this means that differences in method convergence and performance can be largely attributed to algorithmic differences and will elucidate the relative advantages of the SQP and ADMM approaches.

Next we perform a convergence test using a single time step optimization problem from the simulation of a rubber toy boat (Fig. 2). Error is measured via the Newton Decrement [Wright and Nocedal 1999], a standard stopping criteria for gradient-based optimization schemes. MFEM converges at a rate comparable to the standard Newton solver whereas WRAPD fails to converge after a thousand iterations. This shows one immediate advantage of our approach over ADMM type solvers - that it can converge to the correct (Newton) solution of our optimization problem.

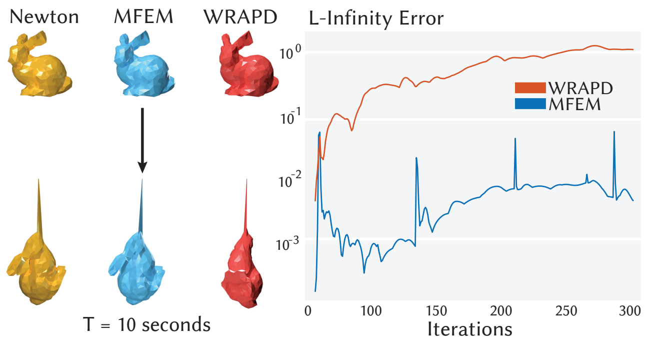

Fig. 3. Small per-time step errors accumulate to cause large trajectory differences in these bunny simulations.

Table 1. Simulation statistics for all examples. Material parameters of density (ρ), Young's Modulus, (**E**), Poisson's ratio (ν), and material model(**Model**) options, stable neo-Hookean (SNH), neo-Hookean (NH), corotational (Corot), Fung or ARAP, specified per example. The density parameter is set to $\rho = 1e3$ (kg/m^3) and Poisson's ratio is set to $\nu = 0.45$ across all examples. We report wall-clock timings in milliseconds (ms) with **Iterations** as the maximum number of solver iterations, **LS** is line search time. Here, the provided timings give the time taken in a single substep. **Assembly** is the time to assemble the linear system; **Global Solve** is the time to solve the linear system; and **Local Solve** is the time taken to solve for λ and s following the global solve.

Example	$\lvert V\rvert$	$\lvert\mathcal{T}\rvert$	Model	E(Pa)	Iterations	Assembly (ms)	Global Solve (ms)	Local Solve (ms)	LS (ms)
Astronaut(soft)	12132	39270	SNH	2e4	3	111.46	222.93	10.57	19.13
Boat	3674	11295	SNH	1e5	50	31.66	36.17	3.17	4.91
Gummy Bear	5701	24393	SNH	6e4, 1e14	5	32.11	218.57	3.68	39.34
Bunny	699	2274	SNH	3e4	50	5.23	12.23	0.31	1.17
Gecko	5929	11552	SNH	1e12	10	42.68	46.23	4.09	6.46
Virus	4593	13342	Corot	1e6	5	32.77	37.46	2.90	5.32
Beam Twist	20000	100793	Corot	1e6	5	248.834	976.70	21.13	33.93
Beam Stretch (SNH)	5000	23701	SNH	1e6	10	71.60	127.59	5.97	8.97
Dragon (BDF1,BDF2)	29229	120045	SNH	1e5	5	286.55	828.36	29.50	41.52

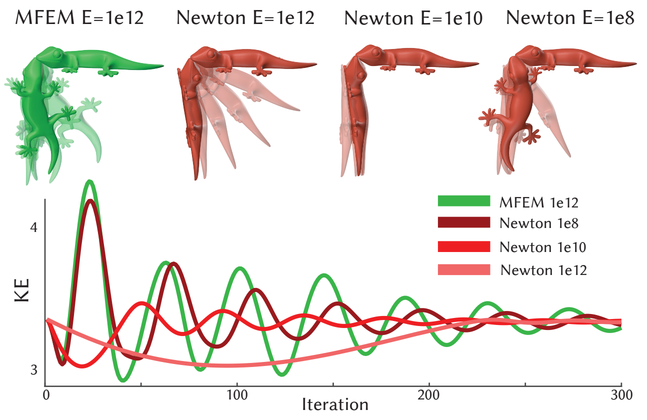

Fig. 4. Swinging Geckos with a range of stiffnesses are simulated using our method and Newton's method with both algorithms capped at a maximum of 10 iterations per time step. Note that in this limited compute time scenario our method produces more natural motions with less damping.

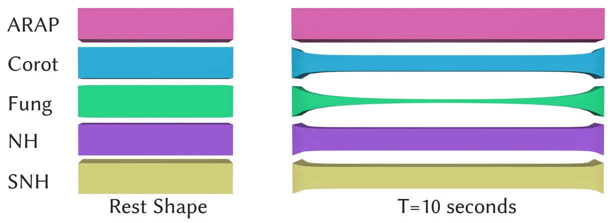

Fig. 5. Simulated beams with a variety of hyperelastic material models

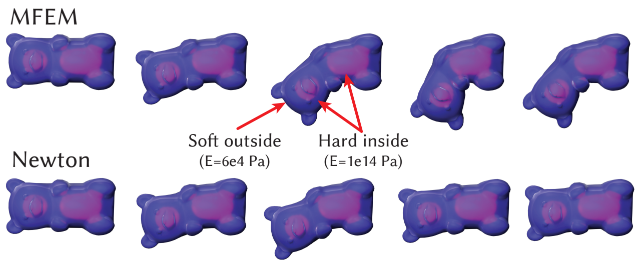

Fig. 6. A heterogeneous gummy bear composed of a hard interior and a layer of soft material on the exterior is simulated with MFEM and Newton solvers. We cap both solvers at 5 iterations per-time step and observe that rotations are well preserved in our method and severely damped in Newton's.

The per-time step differences between algorithms may initially be small, but they accumulate to large trajectory errors over whole simulation runs. Fig. 3 shows a swinging bunny simulation executed by all three algorithms (same initial conditions). Here, the WRAPD output deviates significantly from that of our method and the Newton-based integrator.

Next, when comparing to the Newton-based solver the advantage of our rotation-deformation splitting is that it produces significantly better results when stopped early for performant applications. Fig 4 shows simulations of a gecko mesh at three different stiffnesses, ranging from effectively rigid to stiff rubber. We compare MFEM to Newton's under the conditions of a per-timestep iteration cap of ten per method. Our method produces results with significantly less damping. This damping is a consequence of "under integrating" via early termination of the Newton solver's iterations. While our results are visually plausible, the results from Newton's method are too damped to provide useful animation. This is a key advantage of our approach over standard Newton's method: that it can be stopped very early and still produce expressive deformations as output for a wide range of material properties.

Many algorithms struggle to simulate heterogeneous materials [Modi et al. 2020]. Here in Fig. 6 we simulate a "gummy" bear composed of a soft exterior and partially stiff interior. Here we see the Newton solver produces clear rotational damping. In Figures 7 and 8 we provide convergence plots comparing each algorithm's behavior as we vary the stiffness of the interior. As the stiffness increases, the Newton solver's convergence suffers, whereas our algorithm converges reliably.

These experiments demonstrate that MFEM generates both an accurate simulation when allowed to converge, and plausible animations when terminated early. In both these regimes, our method

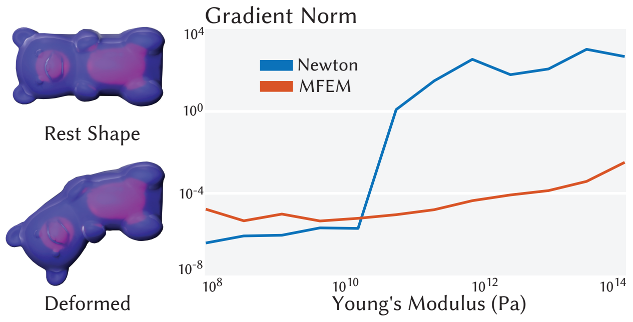

Fig. 7. Starting at an initially deformed state, we simulate a single timestep with each method for 30 iterations and report the final norm of the Newton gradient. We report this value at a number of stiffness values for the interior component of the gummy bear (holding the soft outside fixed) and observe that the Newton-based solver's performance significantly degrades.

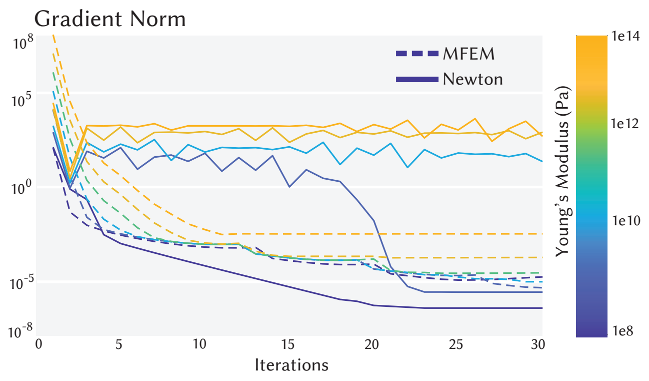

Fig. 8. For the same experiment from Fig. 7, we plot the gradient norm for each solver iteration. The line color indicates the stiffness of the interior component. Here we see Newton's inability to converge within 30 iterations as stiffness increases, whereas MFEM converges much more reliably.

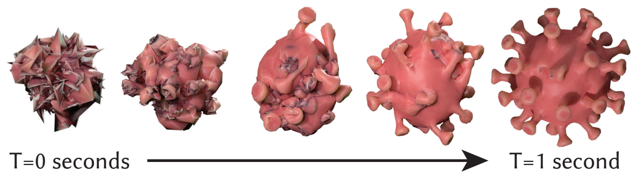

Fig. 9. We randomly place all vertices in this Coronavirus model. Our method generates a stable simulation from this degenerate initial point.

maintains robust simulation output. Fig. 9 shows a stable simulation of a Coronavirus model started from a scrambled initial configuration. Similarly, Fig. 10 shows a large deformation generated by twisting each end of a beam 180 degrees in opposite directions. Our method generates the correct, evenly spaced set of twists.

Our method supports all hyperelastic material models. Fig. 5 shows a set of stretched beams with ARAP, Neohookean, Fung, Stable Neohookean and Co-Rotatated Linearly Elastic material models. In each case we observe the expected behavior (i.e ARAP does not conserve volume whereas the Neohookean models attempt to).

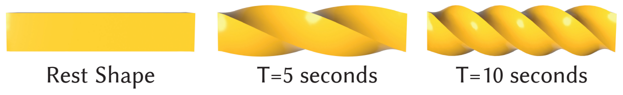

Fig. 10. This twisted beam example illustrates how our method handles large deformations in the presence of moving boundary conditions.

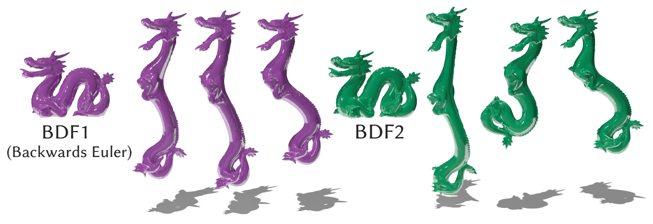

Fig. 11. Two dragons are simulated, one with an Implicit Euler time integrator (BDF1) and the other using BDF2. BDF2 produces a significantly less damped result.

Because our method is optimization based, we are not limited to implicit Euler integration. Fig. 11 shows a comparison of our method using implicit Euler (BDF1), to its second-order relative BDF2.

Last but not least, our method supports contact via penalty springs. Fig 1 shows three space people with stiffnesses spanning a range of 8 orders-of-magnitude, dropped onto a ground plane and robustly deformed. Despite these wide ranges of stiffness, MFEM simulates all of them with same low budget of five iterations (and similarly an inexpensive five inner-iterations of preconditioned conjugate gradient to solve each iterate's linear system). In turn this obtains output at 3-4fps while our supplemental video similarly demonstrates an example of a bunny simulated with output generated at 50fps.

5 CONCLUSION AND FUTURE WORK

We have presented a new, rotationally-aware mixed finite element scheme and have demonstrated that it is convergent, accurate, robust and versatile. Initial experiments indicate that our Mixed FEM approach generates solutions consistent with standard Newton algorithms while producing far less "under integration" when stopped early. These experiments also demonstrate that our method has significantly better convergence properties than state-of-the-art local-global approaches. Typically simulation algorithms are divided into those which converge to accurate solutions and those which can produce plausible output with few iterations. This work suggests that our solver is capable of both. Our algorithm's ability to elegantly handle stiff potentials suggests it would be a good match to state-of-the art methods for barrier-based collision handling and other physical systems with stiff constraints (e.g. incompressible fluids). Further analysis of why our method, structurally quite similar to previous local global methods, should exhibit such improved convergence behavior could help improve other simulation methods as well.

REFERENCES

David Baraff and Andrew Witkin. 1998. Large Steps in Cloth Simulation. In *SIGGRAPH*. ACM, New York, NY, USA, 43–54.

Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. 2014. Projective Dynamics: Fusing Constraint Projections for Fast Simulation. *ACM Trans. Graph.* 33, 4, Article 154 (July 2014), 11 pages.

George E. Brown and Rahul Narain. 2021. WRAPD: Weighted Rotation-aware ADMM for Parameterization and Deformation. *ACM Trans. Graph* 40, 4 (8 2021).

Kazem Cheshmi, Danny M Kaufman, Shoaib Kamil, and Maryam Mehri Dehnavi. 2020. NASOQ: numerically accurate sparsity-oriented QP solver. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 96–1.

Timothy A Davis. 2006. *Direct methods for sparse linear systems.* SIAM.

Theodore F Gast, Craig Schroeder, Alexey Stomakhin, Chenfanfu Jiang, and Joseph M Teran. 2015. Optimization integrator for large time steps. *IEEE transactions on visualization and computer graphics* 21, 10 (2015), 1103–1115.

Gaël Guennebaud, Benoît Jacob, et al. 2010. Eigen v3. http://eigen.tuxfamily.org.

Fabian Hahn, Sebastian Martin, Bernhard Thomaszewski, Robert Sumner, Stelian Coros, and Markus Gross. 2012. Rig-space physics. *ACM transactions on graphics (TOG)* 31, 4 (2012), 1–8.

Ernst Hairer and Christian Lubich. 2014. Energy-diminishing integration of gradient systems. *IMA J. Numer. Anal.* 34, 2 (2014), 452–461.

Alec Jacobson, Daniele Panozzo, et al. 2018. libigl: A simple C++ geometry processing library. https://libigl.github.io/.

Couro Kane, Jerrold E Marsden, Michael Ortiz, and Matthew West. 2000. Variational integrators and the Newmark algorithm for conservative and dissipative mechanical systems. *International Journal for numerical methods in engineering* 49, 10 (2000), 1295–1325.

L. Kharevych, Weiwei Yang, Y. Tong, E. Kanso, J. E. Marsden, P. Schröder, and M. Desbrun. 2006. Geometric, Variational Integrators for Computer Animation *(SCA '06)*. 43–51.

Lei Lan, Danny M Kaufman, Minchen Li, Chenfanfu Jiang, and Yin Yang. 2022. Affine Body Dynamics: Fast, Stable & Intersection-free Simulation of Stiff Materials. *arXiv preprint arXiv:2201.10022* (2022).

David I.W. Levin. 2020. Bartels: A lightweight collection of routines for physics simulation. https://github.com/dilevin/Bartels.

Minchen Li, Ming Gao, Timothy Langlois, Chenfanfu Jiang, and Danny M. Kaufman. 2019. Decomposed Optimization Time Integrator for Large-Step Elastodynamics. *ACM Transactions on Graphics* 38, 4 (2019).

Tiantian Liu, Sofien Bouaziz, and Ladislav Kavan. 2017. Quasi-Newton Methods for Real-Time Simulation of Hyperelastic Materials. *ACM Transactions on Graphics (TOG)* 36, 3 (2017), 23.

Miles Macklin, Kenny Erleben, Matthias Müller, Nuttapong Chentanez, Stefan Jeschke, and Viktor Makoviychuk. 2019. Non-Smooth Newton Methods for Deformable Multi-Body Dynamics. *CoRR* abs/1907.04587 (2019). arXiv:1907.04587

Miles Macklin, Matthias Müller, and Nuttapong Chentanez. 2016. XPBD: Position-Based Simulation of Compliant Constrained Dynamics *(MIG '16)*.

Sebastian Martin, Bernhard Thomaszewski, Eitan Grinspun, and Markus Gross. 2011. Example-Based Elastic Materials. In *SIGGRAPH (SIGGRAPH '11)*. ACM, New York, NY, USA, Article 72, 8 pages.

V. Modi, L. Fulton, A. Jacobson, S. Sueda, and D.I.W. Levin. 2020. EMU: Efficient Muscle Simulation in Deformation Space. *Computer Graphics Forum* (Dec 2020). https://doi.org/10.1111/cgf.14185

Matthias Müller, Nuttapong Chentanez, Miles Macklin, and Stefan Jeschke. 2017. Long Range Constraints for Rigid Body Simulations. In *SCA (SCA '17)*. ACM, New York, NY, USA, Article 14, 10 pages.

Matthias Muller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. 2007. Position based dynamics. *J Vis Commun Image R* 18, 2 (2007), 109 – 118.

Matthias Müller, Miles Macklin, Nuttapong Chentanez, Stefan Jeschke, and Tae-Yong Kim. 2020. Detailed Rigid Body Simulation with Extended Position Based Dynamics. *Computer Graphics Forum* (2020).

Matthew Overby, George E. Brown, Jie Li, and Rahul Narain. 2017. ADMM $\supseteq$ Projective Dynamics: Fast Simulation of Hyperelastic Models with Dynamic Constraints. *IEEE TVCG* 23, 10 (2017), 2222–2234.

E. Reissner. 1985. On mixed variational formulations in finite elasticity. *Acta Mechanica* 56, 3-4 (1985), 117–125.

Martin Servin, Claude Lacoursiere, and Niklas Melin. 2006. Interactive simulation of elastic deformable materials. In *SIGRAD 2006*. Citeseer.

Nicholas Sharp et al. 2019. Polyscope. www.polyscope.run.

Eftychios Sifakis and Jernej Barbic. 2012. SIGGRAPH 2012 Course Notes FEM Simulation of 3D Deformable Solids: A practitioner's guide to theory, discretization and model reduction. (version: August 4, 2012). http://femdefo.org

Breannan Smith, Fernando De Goes, and Theodore Kim. 2018. Stable Neo-Hookean Flesh Simulation. *ACM Trans. Graph.* 37, 2, Article 12 (2018), 15 pages.

Demetri Terzopoulos and Hong Qin. 1994. Dynamic NURBS with Geometric Constraints for Interactive Sculpting. *ACM Trans. Graph.* 13, 2 (April 1994), 103–136.

Huamin Wang and Yin Yang. 2016. Descent methods for elastic body simulation on the GPU. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 1–10.

S J Wright and J Nocedal. 1999. Numerical optimization. (1999).

Zangyueyang Xian, Xin Tong, and Tiantian Liu. 2019. A Scalable Galerkin Multigrid Method for Real-Time Simulation of Deformable Objects. *ACM Trans. Graph.* 38, 6, Article 162 (2019), 13 pages.