Volume xx (200y), Number z, pp. 1–15

EMU: Efficient Muscle Simulation in Deformation Space

V. Modi[1], L. Fulton[1], S. Sueda[2], A. Jacobson[1], D.I.W. Levin[1]

[1]University of Toronto, Toronto, Canada
[2] Texas A&M University, College Station, TX

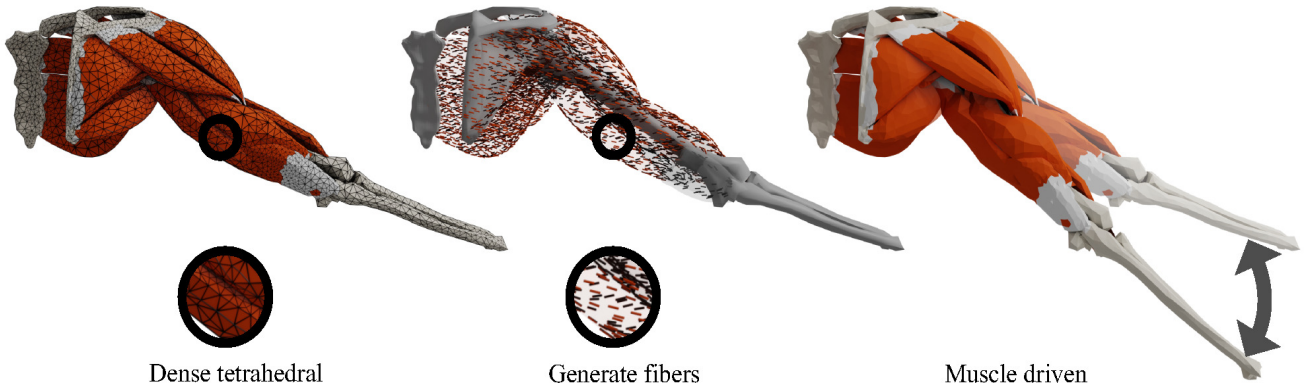

Figure 1: *EMU allows volumetric muscle driven skeletal motion for efficient quasi-static simulation. EMU simulates volumetric musculoskeletal systems (left), complete with embedded, anisotropic fiber fields (middle), and correctly handles joints, stiff tendons, and bones to provide a holistic approach to musculoskeletal animation (right). EMU is asymptotically much faster than FEM—10x faster on a mesh size similar to the one above. (Video: 0m54s)*

Abstract

EMU is an efficient and scalable model to simulate bulk musculoskeletal motion with heterogenous materials. First, EMU requires no model reductions, or geometric coarsening, thereby producing results visually accurate when compared to an FEM simulation. Second, EMU is efficient and scales much better than state-of-the-art FEM with the number of elements in the mesh, and is more easily parallelizable. Third, EMU can handle heterogeneously stiff meshes with an arbitrary constitutive model, thus allowing it to simulate soft muscles, stiff tendons and even stiffer bones all within one unified system. These three key characteristics of EMU enable us to efficiently orchestrate muscle activated skeletal movements. We demonstrate the efficacy of our approach via a number of examples with tendons, muscles, bones and joints.

CCS Concepts

• ***Computing Methodologies*** → ***Applied computing;***

1. Introduction

An accurate portrayal of character motions in animation requires biologically representative musculoskeletal simulations. Computer graphics has a long and successful history of developing efficient simulations of elastica [TPBF87]. However typical approaches suffer from both modeling and performance issues when applied to musculoskeletal applications. For instance, methods that rely on using a coarse simulation mesh also coarsen the muscle fiber field leading to difficulties generating realistic deformations, limitations in the ability to resolve small anatomical features such as tendons and numerical stiffening. Relying on fast, projective dynamics methods limits the types of material models that can be applied, which can lead to simulated behavior that is both visually

off-putting and unstable. Finally, subspace methods coupled with optimized cubature can significantly alter the effect of material parameters making assets difficult to create as textbook material parameters cannot be used directly. In contrast, we provide an algorithm that can produces *visually indistinguishable, unreduced* results that still scale well with the number of elements.

In this work we focus on quasi-static simulation of large-scale muscle activated motion. Quasi-static simulations, in which the inertial effects of the musculoskeletal system are ignored, are often used to animate movements. A well known example is Weta's Tissue solver which animates a large musculoskeletal motion as a series of small quasi-static steps. Quasi-statics is ideal to animate movements with slow-to-medium acceleration where secondary dynamics effects such as elastic wave propogation are visually unimportant. Time steps are essentially infinite, which leads to a variety of special problems such as tunneling effects during collision resolution.

Towards this goal, we propose an efficient finite element scheme which allows for the unified simulation of muscles, tendons, bones and joints, at high-resolution for both bone- and muscle-first applications. We combine our simulator with a manual authoring system that allows a user to setup joints, build muscle fiber fields and identify tendon regions, given input surface geometry of a musculoskeletal system. Finally, we demonstrate the efficacy of our approach on a number of examples of musculoskeletal simulation.

To summarize, our method, EMU, offers the following three desirable attributes:

- *Visually accurate*: EMU produces results visually comparable to unreduced FEM.
- *Efficient*: EMU scales and parallelizes well.
- *Heterogeneous*: EMU simultaneously handles muscles, tendons, bones and joints in a unified fashion.

2. Related Works

A classical approach for bulk musculoskeletal (many muscles and bones together) simulation is to rely on standard finite element methods (FEM) applied to high resolution meshes in order to capture the required musculoskeletal detail as implemented in Weta's *Tissue* software. However, such an approach is computationally intensive, suffers from numerical stiffening on heteregenous material, and is difficult to parallelize. Much of the subsequent research has focused on accelerating this procedure. Most approaches are stymied by three complicating factors:

1. The muscle constitutive model is complex, and so simple alternatives do not always provide robust, visually compelling results [SGK18].
2. The motion of a muscle is heavily influenced by the embedded muscle fiber field. Using coarse meshes often leads to coarsening the fiber field as well, and this can therefore limit the space of actuated poses the muscle can reach [IKKP17].
3. The musculoskeletal system is heterogenous and composed of materials which exhibit wildly varying mechanical properties (tendon is 1000× stiffer than muscle). These high stiffness ratios can cause numerical stiffening which can "lock" the system catastrophically [CLMK17].

Below, we review previous attempts at tackling the important, but difficult problem of bulk musculoskeletal simulation.

One approach is to represent the musculoskeletal system using line-of-force models. Here each muscle is represented not as a volume, but as a line (3D curve, potentially with wrapping surfaces or via points) along which a contractile force can act, and the skeletal system is represented as a system of articulated rigid bodies [LT06, DAA*07, SKP08, WHDK12, GvdPvdS13, LPKL14]. Lines of force do not produce volumetric deformations or capture the richness of muscle fiber configurations and require coupling with rigid skeletons [SSB*15, TBHF03, TSB*05, TSIF05, LST09, SLST14].

Coarsening the simulation mesh can also yield speed-ups but at the cost of accurate deformations, as noted in Phace [IKKP17]. However muscle fiber fields must be ignored, or reformed via experimentation [LGMP11]. Recently, fast projective approaches have been applied to muscle simulation [LYP*18]. These approaches necessitate coarsening the simulation mesh and also restrict the class of constitutive models that can be applied. This can result in very stiff animations with artifacts [SGK18].

Eulerian methods have been used for musculoskeletal simulation [FLP14] but these also eschew fiber field modeling and rely on kinematic coupling to rigidly simulated bones. Finally, data-driven approaches have also become popular [PMRMB15, KPMP*17], but these methods are designed for modeling the body as a whole and do not model muscles.

Reduction approaches such as substructuring and pose space methods are difficult to apply to bulk musculoskeletal simulation due to various locking problems that result from complex biomechanical setups [BZ11, XB16]. Furthermore, reduction based methods limit material model choise and can significantly alter the behavior of nonlinear simulations [BEH18, AKJ08, FMD*19]. Sparse meshless methods [FGBP11] use frames and material-aware interpolation functions to significantly reduce the number of degrees-of-freedom in the numerical system. However, these methods have yet to demonstrate efficacy for large-scale, muscle-first simulation and often require approximating force evaluation to achieve good performance [GBFP11].

Multigrid methods promise exact solutions with linear scaling [Bra77], and have been applied to a number of computer graphics problems [ZSTB10, TJM15]. However, heterogeneous nonlinear material, complex geometry, and time-varying activation present the worst-case scenario for constructing effective multigrid hierarchies. We avoid this and operate on a single volumetric mesh directly.

In this paper we focus on developing an unreduced and efficient algorithm for muscle-first simulation of musculoskeletal systems. The key to EMU's success is its use of deformation gradients, rather than nodal positions as the degrees-of-freedom of the simulation. This bares a resemblance to discontinuous methods for shape modeling [BPGK06, KMBG08]. However, stiching the continuous mesh back together from discontinuous elements is still an open problem. An alternative discontinuous approach is rotation-strain coordinates which can unfortunately cause results to significantly differ from the gold-standard finite element approach [PBH15].

EMU differs from previous discontinuous approaches by mea-

suring discontinuity using the explicit minimizer of a coupling energy, rather than minimizing the coupling energy and physics energy of the system simultaneously. In this way, it has something in common with projective dynamics [BML*14, IKKP17] or Fast Automatic Skinning Transforms [JBK*12]. However details matter, and there are key differences between EMU and projective dynamics as highlighted in Table 1. Although one component of our energy term resembles the term in Projective Dynamics, rather than minimize this energy using alternating projections or a variant of the alternating direction method of multipliers [NOB16], we leverage the algebraic properties of this energy to construct an efficient quasi-newton algorithm [WN99] with capabilities *beyond* the quasi-newton algorithms proposed in [LBK17, ZBK18, LGL*19].

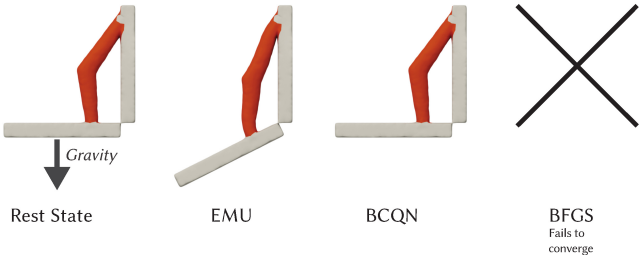

Figure 2: *(Left) A simple, geometric hinge joint fixed at the top, under gravity. (Middle) With our framework, the hinge operates as expected. (Right) With BCQN the hingle locks.*

Readers should note that even though we describe EMU as a "quasi-newton method", we approximate our Hessian differently than other quasi-newton methods as described by 3.2.1. Unlike EMU which uses an approximate Hessian to propagate second order information across joints, recent quasi-newton methods such as [ZBK18] and [LBK17] use a BFGS like approach with a pre-conditioned Hessian which results in locking on the simplest meshes as shown in Figure 3 and Figure 2. Our comparison in Figure 2 shows the superiority of EMU for jointed systems. Other quasi-newton methods are applicable solely to homogenous, elastic, jello-like objects. Unlike EMU, they do not inherently support heterogenous materials and joints. Thus for Figure 2, the hinge joint is modeled geometrically as a shared edge between the two bone regions.

Table 1 summarizes the advantages of EMU over recent projection based methods. Of these methods, [ARM*19] and [LYP*18] are the two most recent ones focused on musculoskeletal deformation. EMU's mesh density and muscle fiber density is much higher with noticeably better volume preservation. EMU handles tendons, which are three orders of magnitude stiffer than muscles. We handle joints and bones without any coupling terms. And we allow arbitrary constitutive models. EMU is a high performance algorithm with capabilities beyond those demonstrated by previous works.

3. Method

We model the bulk motion of a musculoskeletal system as a quasistatic elasticity simulation driven by varying activation of muscle groups. In this paper, we do not consider dynamics-dominated motions (e.g., leaping, running, punching); instead we ignore inertial effects and focus on the muscle-driven deformation of the musculoskeletal system by assuming slowly accelerating (but non-trivial) activations. In the language of continuum mechanics, this quasistatic deformation at some time t can be written as the solution to a scalar energy minimization problem

$$\underset{\mathsf{q}}{\operatorname{argmin}} \int_{\Omega} \Psi_{\text{iso}}(\mathsf{F}(\mathsf{q})) + \Psi_{\text{fiber}}(\mathsf{F}(\mathsf{q}), \mathsf{u}, a(t)) - W(\mathsf{q})\, d\mathsf{Q}, \quad \text{subject to pin and joint constraints} \tag{1}$$

over the domain ω, where $\mathsf{F} : \omega \to \mathbb{R}^{3x3}$ is the deformation gradient, $\mathsf{q} : \omega \to \mathbb{R}^3$ is deformed positions of corresponding rest positions $\mathsf{Q} : \Omega \to \mathbb{R}^3$ over the domain Ω. The internal potentials Ψ_{iso} and Ψ_{fiber} are in general spatially varying, parameterized by materials, and defined as functions of the deformation gradient F which is based on q.

In particular, Ψ_{iso} is a Neo-Hookean isotropic elastic potential constructed to be significantly stiffer in bone and tendon regions of the domain than in the muscles. Meanwhile, Ψ_{fiber} represents the active fiber-reinforcement parameterized by the muscle fiber direction $\mathsf{u} : \Omega \to \mathbb{S}^2$ and time-varying activation function $a(t) : \Omega \times \mathbb{R}_{\geq 0} \to \mathbb{R}$. Work induced by external loads is captured by W, and we consider constraints to the system such as pinning points, fixing regions (e.g., entire bones) or constraining neighboring bones to rotate according to a specified joint (see subsection 3.3). Without loss of generality, we omit W for the next section of this discussion and reintroduce it later on.

While we use stable Neo-Hookean materials [SGK18] and a standard model of muscle behavior as a fiber reinforced composite [TBHF03], the discretization and numerical methods to follow accept any valid potential energy for Ψ_{iso} and Ψ_{fiber}.

Discretizing and optimizing the problem in Eq. 1 is numerically daunting due to fiber anisotropy and the large disparity in material stiffness (bones, tendons, and muscles have Young's moduli of $\sim 10^{10}$ Pa, $\sim 10^8$ to $\sim 10^9$ Pa and $\sim 10^5$ Pa respectively as noted in [MP99, RKSZ98]). Discretizing bones and muscles differently requires awkward coupling constraints [SSF08] (e.g., treating bones as perfectly rigid objects and muscles as soft bodies). Even state-of-the-art coupling algorithms [WWB*19] require chain-rule-like computations to implement linearly-implicit time integration, which is significantly more complex than the non-linear statics solve that EMU performs. Meanwhile, direct simulation with the finite-element method suffers from numerical instability and poor convergence due to the high condition number of the resulting system. EMU's framework preserves the near perfect rigidity of real bones, stiffness of tendons and the compliance of soft muscles, without the overhead of any coupling constraints.

3.1. Discretization

We propose discretizing the problem in Eq. 1 using a variable separation approach. Let n and m be the the number of vertices and tetrahedra, respectively. We use vertex positions $\mathsf{q} \in \mathbb{R}^{3n}$ to track the volumetric deformation and introduce *independent variables* representing the deformation gradient $\mathsf{F}_i \in \mathbb{R}^{3\times 3}$ for each tetrahedron, $i \in \{1, \ldots, m\}$.

	Muscle Driven Motion	Handles Joints	Anisotropic Fibers	Nonlinear Stiffness	Handles Tendons	No Coupling Terms
EMU	Y	Y	Y	Y	Y	Y
[ZBK18] BCQN	N	N	N	Y	N	N
[LGL*19] DOT	N	N	N	Y	N	N
[LBK17] Liu et al.	N	N	N	Y	N	N
[KDGI19] Kim et al.	N	N	Y	Y	N	N
[ARM*19] VIPER	N	Y	Y	N	N	N
[LYP*18] Lee et al.	Y	N	Y	N	N	N

Table 1: *Comparing the capabilities of EMU against some of its contemporaries. 'Y' indicates the capability is supported. 'N' indicates it is either not supported or not demonstrated in the paper. EMU alone supports all the features highlighted above. The column "No Coupling Terms" refers to the necessity of having extra coupling terms for rigid bones and soft muscles in other methods. EMU is the only method that handles muscles and bones in one unified framework.*

For piecewise-linear finite-elements, the deformation gradient of a tetrahedron is linearly *dependent* on the deformed vertex positions:

$$\begin{pmatrix} \mathsf{q}_{i1}-\mathsf{q}_{i4} \\ \mathsf{q}_{i2}-\mathsf{q}_{i4} \\ \mathsf{q}_{i3}-\mathsf{q}_{i4} \end{pmatrix}^{\top} = \mathsf{F}_i \begin{pmatrix} Q_{i1}-Q_{i4} \\ Q_{i2}-Q_{i4} \\ Q_{i3}-Q_{i4} \end{pmatrix}^{\top}. \tag{2}$$

where q_{ij} and Q_{ij} are the deformed and rest vertex positions of the jth corner of the ith tetrahedron, respectively.

However, we do *not* require strict satisfaction of this equation. Instead each independent deformation gradient F_i is free to represent deformations of a much wider class of meshes than the continuous tetrahedral mesh parametrized by vertex positions q. However, since we are ultimately interested in visualizing the continuous deformations, we find the **nearest continuous mesh**, q^*, by satisfying Eq. 2 in a least-squares sense:

$$\begin{aligned} E_{\mathrm{C}}(\mathsf{q},\mathsf{F}) &= \frac{1}{2}\sum_{i=1}^{m}\left\| \begin{pmatrix} \mathsf{q}_{i1}-\mathsf{q}_{i4} \\ \mathsf{q}_{i2}-\mathsf{q}_{i4} \\ \mathsf{q}_{i3}-\mathsf{q}_{i4} \end{pmatrix}^{\top} \begin{pmatrix} Q_{i1}-Q_{i4} \\ Q_{i2}-Q_{i4} \\ Q_{i3}-Q_{i4} \end{pmatrix}^{-\top} - \mathsf{F}_i \right\|^2 \\ &= \frac{1}{2}\mathsf{q}^{\top}\mathsf{G}^{\top}\mathsf{G}\mathsf{q} - \mathsf{q}^{\top}\mathsf{G}^{\top}\mathsf{F} + \frac{1}{2}\mathsf{F}^{\top}\mathsf{F} \end{aligned} \tag{3}$$

where $\mathsf{F} \in \mathbb{R}^{9m}$ is a single vector stacking coefficients of all m pertet deformation gradient *variables* and $\mathsf{G} \in \mathbb{R}^{9m\times 3n}$ is the sparse matrix that computes the *actual* deformation gradients from the mesh deformed according to q.

This energy is zero when the deformed mesh implied by F is continuous. In other words, this energy describes the distance between our independent DOFs F and some continuous mesh represented by vertices q. This amounts to a poisson-like solve with a constant Laplacian similar to [YZX*04] where the poisson equation is viewed as an alternate to least-squares minimization. In our least-squares reconstruction of the nearest continuous mesh, the nullspace corresponding to rigid transformations is removed by fixing vertices in at least one bone. We refer to E_{C} as the *as-continuous-as-possible* (ACAP) energy.

Given a particular set of deformation gradients F, the optimal deformed vertex positions (or **nearest continuous mesh**) q^* that minimize E_{C} are the solution to a sparse linear system:

$$\mathsf{q}^* = \underset{\mathsf{q}}{\operatorname{argmin}}\ E_{\mathrm{C}}(\mathsf{q},\mathsf{F}) = \left(\mathsf{G}^{\top}\mathsf{G}\right)^{-1}\mathsf{G}^{\top}\mathsf{F}. \tag{4}$$

Out of all other continuous mesh representations q, continuous mesh represented by q^* most closely resembles our deformation gradients F. Any change in F would require us to re-solve the equation above for a new q^*, thus making q^* a function of F.

Since, q is now a function of F, we can discretize the energy minimization Eq. 1 as a minimization over only F:

$$\min_{\mathsf{F}} \underbrace{\Psi_{\mathrm{iso}}(\mathsf{F}) + \Psi_{\mathrm{fiber}}(\mathsf{F},\mathsf{u},a(t)) + \alpha E_{\mathrm{C}}(\mathsf{q}(\mathsf{F}),\mathsf{F})) - W(\mathsf{q}(\mathsf{F}))}_{E(\mathsf{F})} \tag{5}$$

where $\alpha \geq 0$ is a scalar parameter controlling the continuity implied by F. Intuitively, we have replaced the hard constraint on continuity implied by standard finite element approaches, with a soft penalty approach, which will yield dividends later on. For all finite choices of α, the deformation gradient variables can *break* continuity to provide a type of compliance in the system, which aids optimization. A higher α generally means a higher level of continuity. As with all penalty methods, α can be chosen by the user to achieve a desired effect; however, in section *3.7* we will detail an effective heuristic for choosing an α that provides both good simulation performance and visual accuracy.

With this variable separation we can see that the internal potential terms Ψ_{iso} and Ψ_{fiber} in Eq. 5 no longer depend on the deformed vertex positions q. Furthermore, if these are discretized using the standard piecewise-constant strain assumption, these terms become *easily parallelizable* summations over the tetrahedra. The computation at each tetrahedron only depends on data associated with exactly that tetrahedron (even the E_{C} term is easily parallelizable as matrix-vector multiplications):

$$\Psi_{\mathrm{iso}}(\mathsf{F}) = \sum_{i=1}^{m}\Psi_{\mathrm{iso}}(\mathsf{F}_i), \tag{6}$$

$$\Psi_{\mathrm{fiber}}(\mathsf{F},\mathsf{u},a(t)) = \sum_{i=1}^{m}\Psi_{\mathrm{fiber}}(\mathsf{F}_i,\mathsf{u}_i,a_i(t)). \tag{7}$$

In our examples, we use the (non-linear) inversion safe Stable Neo-Hookean energy [SGK18] for Ψ_{iso}. Since biomechanical simulations do not require extreme elemental deformations, a non inversion-safe energy would work just as well as Stable Neo-Hookean as long as inversions are penalized by assigning a large (1e40) energy value to inverted elements. For Ψ_{fiber}, we use linear activation [TBHF03] along a piecewise-constant direction field:

$$\Psi_{\text{fiber}}(\mathsf{F}_i, \mathsf{u}_i, a_i(t)) = a_i(t)\mathsf{u}_i^\top \mathsf{F}_i^\top \mathsf{F}_i \mathsf{u}_i, \tag{8}$$

where $a_i(t) \geq 0$ is the non-negative activation at the ith tetrahedron at time t, and $\mathsf{u}_i \in \mathbb{S}^2$ is the unit-length fiber direction vector at the ith tetrahedron.

3.2. Descent-direction solver choice

Next we turn to the question of how to best minimize Eq. 5. One option is BFGS. BFGS [WN99] or its limited-memory variant (LBFGS) are quasi-newton methods that are both popular and effective for physics simulation [BC80]. These approaches have the benefit of only requiring the gradient of the objective function, avoiding expensive Hessian computations. The gradient of Eq. 5 can be computed as:

$$\begin{aligned} \frac{dE}{d\mathsf{F}} &= \frac{\partial \Psi_{\text{iso}}}{\partial \mathsf{F}} + \frac{\partial \Psi_{\text{fiber}}}{\partial \mathsf{F}} + \alpha \frac{dE_{\text{C}}}{d\mathsf{F}} \\ &= \frac{\partial \Psi_{\text{iso}}}{\partial \mathsf{F}} + \frac{\partial \Psi_{\text{fiber}}}{\partial \mathsf{F}} + \alpha \frac{\partial E_{\text{C}}}{\partial \mathsf{F}} + \alpha \cancelto{0}{\frac{\partial E_{\text{C}}}{\partial \mathsf{q}}} \frac{\partial \mathsf{q}}{\partial \mathsf{F}}, \end{aligned} \tag{9}$$

where

$$\frac{\partial E_{\text{C}}}{\partial \mathsf{F}} = \mathsf{G}\mathsf{q}(\mathsf{F}) + \mathsf{F} = -\mathsf{G}\left(\mathsf{G}^\top \mathsf{G}\right)^{-1} \mathsf{G}^\top \mathsf{F} + \mathsf{F}. \tag{10}$$

Here we utilize the optimality of $\mathsf{q}(\mathsf{F})$ to eliminate the term depending on $\partial E_{\text{C}}/\partial \mathsf{q}$. This $dE/d\mathsf{F}$ can be computed efficiently by *precomputing* a sparse factorization of the constant sparse symmetric matrix $\mathsf{G}^\top \mathsf{G} \in \mathbb{R}^{3n \times 3n}$.

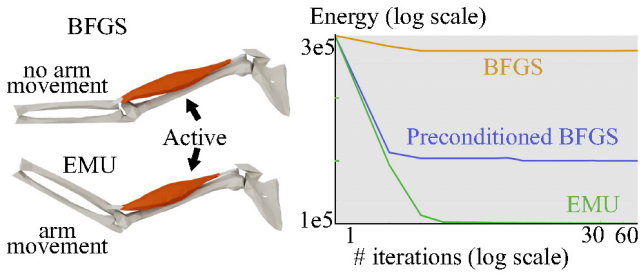

Figure 3: *BFGS approximates the Hessian using only gradient information. This leads to catastrophic locking at joints. Our method works as expected.*

Unfortunately, our experiments showed that the BFGS method can get immediately stuck in a locked configuration. This is especially likely once we later introduce joint constraints. Because the physics energies are completely decoupled (Ψ_{iso} and Ψ_{fiber} depend only on F_i of each tetrahedron), it becomes the job of E_{C} to distribute motion throughout the system. At the initial position, both E_{C} and $\partial E_{\text{C}}/\partial \mathsf{F}$ are 0. Upon activation of a muscle, the individual muscle tetrahedra contract, but there is no movement induced in the bones. Because bones and tendons are very stiff relative to the muscles, the system will remain in place, unable to transfer the force created by muscle contraction across the joints (Figure 3). Essentially, this force transfer is a second-order effect and is not captured by the gradient, which only provides information about each tetrahedron in isolation. Previous work shows BFGS is not well suited for this type of application [ZBK18]. Even pre-conditioning the LBFGS search direction does not sufficiently capture the second-order effects needed for joint motion.

To incorporate these second-order effects, the natural solution would be to simply use newton's method. However standard newton's method requires the computation of the Hessian matrix:

$$\frac{d^2E}{d\mathsf{F}^2} = \frac{\partial^2 \Psi_{\text{iso}}}{\partial \mathsf{F}^2} + \frac{\partial^2 \Psi_{\text{fiber}}}{\partial \mathsf{F}^2} + \alpha \frac{\partial^2 E_{\text{C}}}{\partial \mathsf{F}^2}, \tag{11}$$

where

$$\frac{\partial^2 E_{\text{C}}}{\partial \mathsf{F}^2} = -\mathsf{G}\left(\mathsf{G}^\top \mathsf{G}\right)^{-1} \mathsf{G}^\top + \mathsf{I}. \tag{12}$$

Terms $\frac{\partial^2 \Psi_{\text{iso}}}{\partial \mathsf{F}^2}$ and $\frac{\partial^2 \Psi_{\text{fiber}}}{\partial \mathsf{F}^2}$ are sparse, block diagonal and simple to compute. However, the last term ($\frac{\partial^2 E_{\text{C}}}{\partial \mathsf{F}^2}$) requires taking the inverse of a sparse matrix (which is dense), making a direct computation of the Hessian intractable for all the but smallest of examples. For example, on a mesh with 50k tets, the dense inverse would be 1GB large. Many of our examples are similar in size or larger. Thus, even construction of this dense Hessian is prohibitory, let alone computation with it. Therefore, standard newton's method is not viable for EMU.

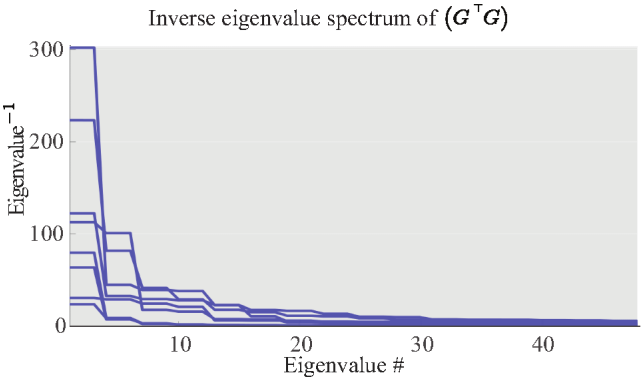

Figure 4: *Eigenvalue spectrum of $G^\top G$ for all the examples. Regardless of the example mesh, the first 48 eigen-modes capture most of the variance in $G^\top G$. Thefore we choose a number of eigen vectors that encompasses 90 percent of the spectrum.*

3.2.1. Alternative Quasi-Newton Method

Instead we derive a new quasi-newton approach, similar in asymptotic performance to the standard BFGS update. A quasi-newton

method is one which doesn't use the exact Hessian [WN99]. Motivated by this we derive a fast new way to evaluate approximate Hessian which uses the exact Hessian for the first two sparse, block-diagonal terms ($\partial^2\Psi_{\text{iso}}/\partial\mathsf{F}^2$ and $\partial^2\Psi_{\text{iso}}/\partial\mathsf{F}^2$) but approximates the last, dense term $\partial^2 E_C/\partial\mathsf{F}^2$ using a low-rank approximation.

The dominant step in Eq. 11 is the inversion of the large matrix $\mathsf{G}^\top\mathsf{G}$. Therefore, we take the eigen decomposition:

$$\left(\mathsf{G}^\top\mathsf{G}\right)^{-1} \approx D = \Phi^\top\Lambda^{-1}\Phi, \tag{13}$$

where $\Phi \in \mathbb{R}^{k\times 3n}$ collects the eigen vectors corresponding to the lowest k eigen values, which are placed along the diagonal of $\Lambda \in \mathbb{R}^{k\times k}$. Observing the eigenvalue spectrum of $G^\top G$ (Figure 4), we justify a reduction via the first $k << m$ eigen-modes. In this case, we notice that $k = 48$ sufficiently describes over 90% of the variance in $G^\top G$ for all our examples.

Substituting Eq. 13 in Eq. 11 results in an expression that still does not yet admit efficient *solving* with the Hessian, which is now:

$$\frac{d^2E}{d\mathsf{F}^2} \approx \mathsf{H} - \alpha\mathsf{B}^\top\Lambda^{-1}\mathsf{B}, \tag{14}$$

where

$$\mathsf{H} = \frac{\partial^2\Psi_{\text{iso}}}{\partial\mathsf{F}^2} + \frac{\partial^2\Psi_{\text{fiber}}}{\partial\mathsf{F}^2} + \alpha\mathsf{I} \quad \text{and} \quad \mathsf{B} = \Phi\mathsf{G}^\top. \tag{15}$$

The matrix $\mathsf{H} \in \mathbb{R}^{9m\times 9m}$ is composed of 9×9 blocks along the diagonal. A key insight now becomes apparent. We can make use of the Woodbury matrix identity (see, e.g., [JP99]) which holds that:

$$(A+UCV)^{-1} = A^{-1} - A^{-1}U\left(C^{-1}+VA^{-1}U\right)^{-1}VA^{-1}, \tag{16}$$

for correctly sized matrices A,U,C,V (and A,C invertible).

Applying the Woodbury matrix identity to the inverse of the Hessian expression in Eq. 14 produces:

$$(\frac{d^2E}{d\mathsf{F}^2})^{-1} \approx \mathsf{H}^{-1} + \alpha\mathsf{H}^{-1}\mathsf{B}^\top\left(\Lambda - \alpha\mathsf{B}\mathsf{H}^{-1}\mathsf{B}^\top\right)^{-1}\mathsf{B}\mathsf{H}^{-1}. \tag{17}$$

Each iteration of our quasi-newton solver will need to multiply this expression on the right with the gradient $dE/d\mathsf{F}$ from Eq. 9 to determine the step direction. We can compute this *action* very efficiently: to solve against H we precompute a 9×9 dense factorization corresponding to each tetrahedron and conduct back substitutions in parallel. We also use this action to compute the dense $k\times k$ matrix $\Lambda + \alpha\mathsf{B}\mathsf{H}^{-1}\mathsf{B}^\top$ in Eq. 17 and then solve against it (e.g., using a factorization method at run-time). Multiplications against the precomputed $k\times 9m$ dense rectangular matrices B are conducted in parallel using Eigen [GJ*19].

Our Hessian approximation is guaranteed to be symmetric-positive-definite (SPD) since each term is SPD (we use the standard definiteness fix for the elastic energy). Therefore, our quasi-newton search direction (derived via a low rank approximation of **only** the dense term) is guaranteed to be a *descent* direction.

Once the step direction is computed, we use a back-tracking line search, satisfying the Armijo condition, to ensure sufficient decrease in the objective so our method converges to a local minimum. Unlike modal reduction methods (e.g., [XB16]) which permanently alter the solution space, we only use the eigendecomposition to build an approximate Hessian and retain the exact gradient. Importantly, our quasi-newton optimization approximates the search direction, but solves the *full-space* problem in Eq. 5.

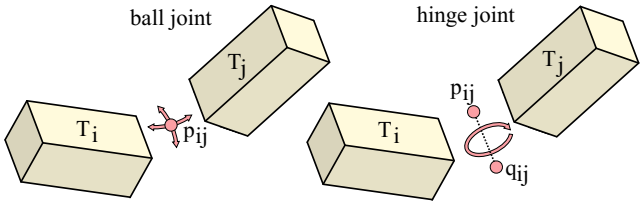

Figure 5: *Joints are implemented as algebraic constraints on the vertices of corresponding bones. We employ two types of joints. Ball joints (left) and hinge joints (right).*

3.3. Affine Bones

Armed with an efficient solver for muscles, we turn our attention to adding bones to the system. Remarkably, our method allows us to model bones, muscles and tendons in one system without any coupling terms. Due to their stiffness, bones deform negligibly and so, we chose to represent all the F_i for a bone mesh by a single deformation gradient. We model bone rigidity by applying a very high young's modulus to the bone elements and ensure the rigidity of the deformation gradient translates to rigid motion of the bone vertices through a constraint as shown in Eq. 20 by the constraint $B_q q = B_F F$. In this equation $B_f F$ are the deformation gradients of the bones elements and $B_q q$ represents the deformation gradients of the bones on the continuous mesh.

$$\mathsf{q}^* = \underset{\mathsf{q}}{\text{argmin}}\ E_C(\mathsf{q},\mathsf{F}) = \left(\mathsf{G}^\top\mathsf{G}\right)^{-1}\mathsf{G}^\top\mathsf{F}, \tag{18}$$

$$\text{s.t.} \qquad B_q q = B_F F \tag{19}$$

$$Jq = 0 \tag{20}$$

In Eq. 4, rather than represent the position of bones using their vertex positions, we instead use a single affine transform for each bone. Joint constraints can then be represented as affine constraints $Jq = 0$ added to Eq. 20 and shown geometrically in Figure 5. The joint constraint expresses that for each joint connecting two bones, the deformation of bone one and bone two, as applied to the points of the joint, are the same, thus $Jq = 0$. This yields a KKT system, shown in Eq. 21, where the left-hand-side is a constant. Though, theoretically, $\varepsilon_1 = \varepsilon_2 = 0$, in practice however, both the joint constraint and bone constraints apply to the same DOFs. Therefore, we introduce a little slack into one of them. By setting $\varepsilon_1 = -1e-4$ we prevent numerical problems while ensuring the deformation remains virtually unaffected.

$$\begin{pmatrix} \mathsf{G}^T\mathsf{G} & J^T & B^T \\ J & \varepsilon_1 & 0 \\ B & 0 & \varepsilon_2 \end{pmatrix} \begin{pmatrix} q \\ \lambda_1 \\ \lambda_2 \end{pmatrix} = \begin{pmatrix} \mathsf{G}^T F \\ 0 \\ B_F F \end{pmatrix} \tag{21}$$

For best performance we use the parallelizable Pardiso solver.

Iterative methods are not competetive since the left-hand-side can be pre-factored (we tested Conjugate Gradients and found it to be slower). Thus EMU is able to handle muscle, tendons, bones and joints all within one contiguous system without the need for coupling terms. In practice we find that it is useful to incorporate weights proportional to material stiffness into Eq. 4 for muscles and tendons, but not for bones. For bones, this is akin to adding a very stiff spring to the physical system which can lead to locking.

ALGORITHM 1: Our contact-aware iterative newton solver.

Data: F- Initial guess for deformation gradients
Result: F- Updated deformation gradients
//Line search step size
$\sigma \leftarrow 10$
//Line search f tolerance
$c \leftarrow 10^{-4}$
//Line search decrement
$\rho \leftarrow 0.5$
do
  //Calculate initial energy
  $e_i = \Psi(\mathsf{F})$
  //Calculate the gradients
  $\mathbf{g} \leftarrow \frac{d\Psi(\mathsf{F})}{dF}$
  //Woodbury method to find the search direction
  $\mathbf{d} \leftarrow \mathsf{H}(\mathsf{F})^{-1}\mathbf{g}$
  //Backtracking line search to find step size
  do
    //Update temp F with descent direction
    $\mathbf{F_{temp}} = \mathsf{F} + \sigma\mathbf{d}$
    //ACAP solve from Eq. 20
    $\mathsf{q} = argmin_{\mathsf{q}}\, E_c(\mathsf{q}, \mathbf{F_{temp}})$
    $\sigma \leftarrow \rho \cdot \sigma$
  while $(\Psi(\mathsf{F}_{temp}) < e_i + \sigma c \mathsf{g}^\top \mathsf{d})$;
  //Update F with new descent direction
  $\mathsf{F} \leftarrow \mathsf{F} + \sigma * \mathbf{d}$
  //Vertex-wise gravity force on the continuous vertices. Mg is the full mass matrix times the per-vertex gravity accelerations.
  $\mathbf{v} \leftarrow Mg$
  //At first set external work gradient (negative external force) to gravity by Eq. 22
  $\mathbf{g_{ext}}$ = ExternalForces($\mathbf{v}$)
  //Compute contact forces that counteract mesh overlap.
  do
    //Run [HTK*04] on the continuous vertices
    $\mathbf{f_c}$ = CollisionForces(q)
    //Update external work gradient
    $\mathbf{g_{ext}} \leftarrow \mathbf{g_{ext}} - \mathbf{f_c}$
    //Woodbury method to update the search direction with contact forces
    $\mathbf{d_{contact}} \leftarrow \mathsf{H}(F)^{-1}\mathbf{g_{ext}}$
    //Update temp F with descent direction
    $\mathbf{F_{temp}} = \mathsf{F} + \mathbf{d_{contact}}$
    //ACAP solve from Eq. 20
    $\mathsf{q} = argmin\, E_c(\mathsf{q}, \mathbf{F_{temp}})$
  while $(\|\delta\mathbf{f_c}\| < \varepsilon_{contact})$;
  //Update F with descent direction accounting for external force
  $\mathsf{F} \leftarrow \mathsf{F} + \mathbf{d_{contact}}$
while $(\|\mathbf{g}\| < \varepsilon_1$ *or* $(\Psi(F) - e_i) < \varepsilon_2)$;

Additionally, we find that in order to prevent locking while increasing the quality of the deformation, certain liberties must be taken with joint and bone deformation constraints in Eq. 20. Bones on the continuous mesh must be allowed to slightly deviate from the discontinuous elements during newton's method in order for the algorithm to find a good search direction as shown in Figure 6. This requires a loosening of the bone deformation constraint in Eq. 21 during newton's method by setting $\varepsilon_2 = -1e-3$ while tightening the joint constraint by setting $\varepsilon_1 = 0$. After the method has converged, the constraint can simply be updated $\varepsilon_2 = 0$ to ensure strict adherence of the bone vertices to the bone's deformation gradient and introducing a negligible slack on the joint as $\varepsilon_1 = -1e-4$.

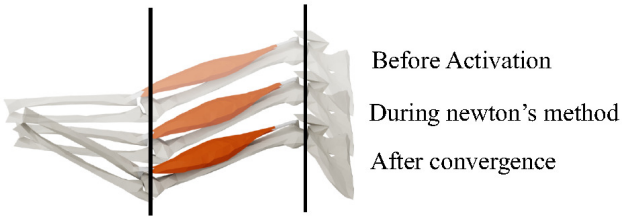

Figure 6: *During newton's method, we update the constraints on Eq. 4 in order to introduce compliance between the bone F and the bone's vertices. Notice the length of the bone temporarily changes within the quasi-newton method allowing the muscle elements to attain a plausible deformation without locking the system. After convergence, Eq. 4 is run once more with updated constraints that ensure the bone F exactly match the continuous mesh, thus verifying that the bone does not deform.*

3.4. External Forces

External forces such as gravity can be applied to our system using the standard Jacobian transpose method for converting per-vertex forces acting on the continuous tetrahedral mesh to generalized forces acting on the per-tetrahedron deformation gradients. The rate of work done by an external force is given by

$$\dot{\mathsf{q}}^T \boldsymbol{f}_{ext} = \dot{\mathsf{F}}^\top \underbrace{\left(\left(\mathsf{G}^\top \mathsf{G}\right)^{-1} \mathsf{G}^\top\right)^\top \boldsymbol{f}_{ext}}_{\text{Generalized Force: } \nu}. \tag{22}$$

For constant forces, such as gravity, the work in Equation 1 becomes $\mathsf{F}^\top \nu$ where ν can be efficiently computed at startup using the prefactored $\mathsf{G}^\top\mathsf{G}$. During the optimization, f_{ext} is added as a constant external force.

3.5. Collision Resolution

Although not the focus of this work, the advantage of using a quasi-newton search strategy for optimization is that we can easily incorporate standard collision resolution into the algorithm — all that is required is a method of: (1) detecting collisions in between muscles and bones and (2) computing forces that will resolve these collisions. For (1) and (2) we rely on [HTK*04]. We run collision resolution after the line search in our quasi-newton method to ensure

that our meshes are collision free at the end of each newton iteration, similar to other projection based algorithms for contact and constraint handling. In line 43, we show the details of our particular implementation. We exploit the efficiency of inverting the EMU Hessian and our prefactored ACAP Hessian to propagate per-vertex contact forces through the mesh efficiently. These two properties can be exploited in other contact-aware gradient based algorithms as well. We typically allow a small amount of overlap in our simulations as it helps to reach converged solutions in cases with many closely conforming muscles. In general contact handling in such scenarios remains an open problem in graphics that we leave for future explorations.

It should be noted that although we show, for the benefit of the reader, that standard collision resolution methods work within EMU, this interpenetration is biologically infeasible. Muscle and bones are surrounded by sheaths of connective tissue called fascia, which limit deformation and limit contact. The simplest and most biologically accurate solution for interpenetrating muscles would be to fuse the muscle meshes together and activate each section separately.

3.6. Modeling

Our musculoskeletal models start life as separate triangle meshes for each bone and muscle. In all our examples, we must manually set joint locations for each socket and hinge joint. Next, we fuse muscle and bone meshes by manually overlapping them and then tetrahedralizing using TetWild [HZG*18, SCM*18]. After this stage, each tetrahedron is labelled as either muscle or bone. We manually select muscle tetrahedra near the origin and insertion of each muscle to serve as tendons. Finally we assign material properties to our tetrahedral mesh. For muscles we use Young's Modulus of $6e6$ Pa, bones $1e10$ Pa and tendons $4.5e8$ Pa to $1.2e9$ Pa derived from biological measurements. We use a Poisson's ratio of 0.49 for all materials.

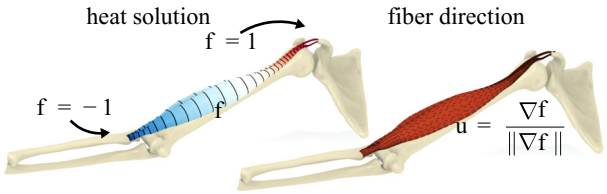

Figure 7: *The muscle fiber directions* u *are computed as the gradient direction of a solution to the heat equation* $\Delta f = 0$ *with sources and sinks at opposite attachment points.*

For each muscle, we automatically compute a fiber direction u using the heat equation. We set Dirichilet boundary conditions of 1 and -1 at insertion and origin points of the muscle and compute the equilibrium heat distribution. We take the normalized gradient of this field to be the fiber direction shown in Figure 7.

3.7. The Weighting Paramter

With all the pieces of EMU in place, we can now detail how we choose the ACAP energy weighting term, α. On homogeneously stiff muscle meshes, higher α values produce deformations closer to FEM (Figure 8). However, since higher α increases the stiffness of the system, it non-linearly increases the number of newton iterations to convergence. For example, the 22k homogenous muscle mesh at $\alpha = 1$ requires 21 newton iterations to converge while at $\alpha = 1e8$ it requires 42,151 iterations. On the other hand, an exceedingly small α will allow the continuous mesh to drift away from the deformation gradients.

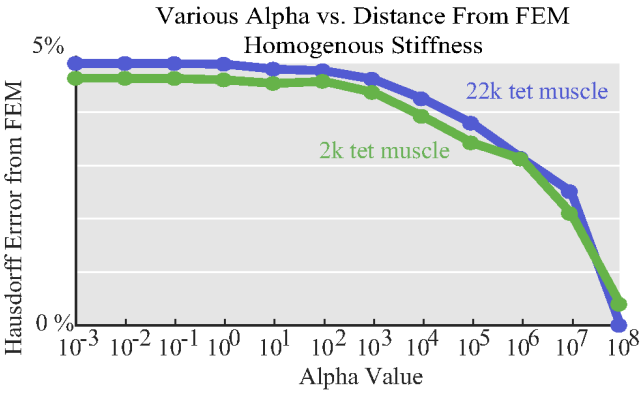

Figure 8: *Measures the Hausdorff distance divided by the rest length of the mesh between EMU results varying* α *values and the FEM result on a 2k tet muscle and 22k tet muscle with homogenous stiffness. As* α *increases the distance goes to 0.*

With the addition of stiff tendons and bones into the system, the relationship between α and distance from FEM deformations is not as clear, as shown in Table 2. Under an exceedingly high α the stiff region's continuous mesh will not be allowed to deviate from the deformation gradient. As explained in subsection 3.3 and shown in Figure 6 this will lock the deformation since the bone vertices will not be allowed to deviate from the bone F during the optimization. However, experiments show that there exists a value of α which results in deformations that resemble FEM even more closely than ADMM. Therefore, we provide a heuristic below to find a good α.

To find a good α we linearly search over the 1D space of weighting parameters (starting with $\alpha = 1$) and tally the number of newton iterations taken by EMU for some fixed muscle activation. We increase α until we observe a sharp increase in the number of Newton terations per step. We take the alpha immediately before this increase motivated by notion that penalty term optimizations admit an optimal alpha that exhibits fast convergene to the local optimum [WN99]. We have found this heuristic generates good visual agreement with finite element results and also optimizes for speed as shown in Figure 9. Let us note that our experiments illustrate that EMU produces visually pleasing results (though with differing deformations) for all values of α. Ultimately, animation involves a fair bit of artwork, and visual appeal is subjective. Alpha can

Tets	12,184	51,271
	Error	*Error*
FEM	0.000	0.000
ADMM	7.048	7.381
EMU $\alpha = 1e0$	12.095	11.119
EMU $\alpha = 1e1$	8.214	6.428
EMU $\alpha = 1e2$	2.143	1.500
EMU $\alpha = 1e3$	2.381	3.214
EMU $\alpha = 1e4$	1.667	3.119
EMU $\alpha = 1e5$	8.095	5.786

Table 2: *Accuracy comparisons between ADMM, EMU and FEM with muscle activation under gravity. The error measured here is the Haussdorf distance from FEM for each result, divided by the rest-state length of the mesh.*

be used to tailor the visual output of the EMU simulation in more artistic applications.

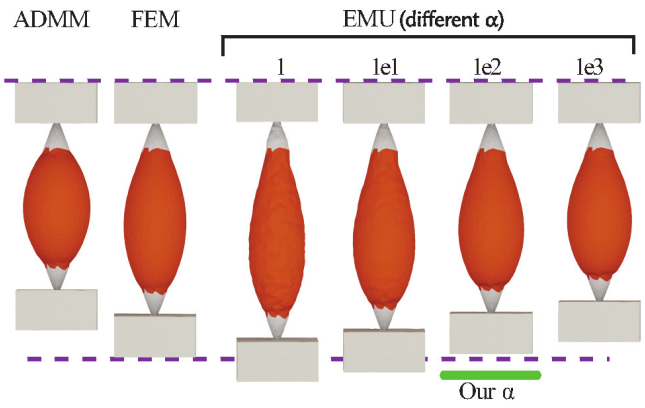

Figure 9: *Comparison of ADMM, FEM and EMU on fusiform muscles hanging under gravity. EMU results vary with the chosen alpha parameter but the optimal alpha chosen using our heuristic produces excellent agreement with the FEM solution and is more accurate than ADMM.*

4. Results and Discussion

We simulate a variety of musculoskeletal geometries using EMU. The tetrahedral count of our models range from a small 3k tetrahedron soft robot to a large 600k tetrahedron muscle as shown in Table 3. Not included in the table below were the various sized simple fusiform meshes generated for the performance and scalability tests. For each mesh, the initial step involved finding the first k modes of the Hessian (Eq. 17) once, upfront. The runtime of this pre-processing step differs significantly based on the mesh connectivity and number of bones in the mesh, but ranges from several seconds to several minutes for our larger examples. The second pre-processing step involves choosing an α. This involves, at most, 10 Newton solves on the mesh. However, since these are one time operations, we exclude them from our performance numbers.

Model	Num Tets	Bones	Muscles
simple fusiform muscle	600k	2	1
simple bipennate muscle	12k	2	1
simple contacting muscles	20k	2	2
curved contacting muscle	11k	2	1
elephant head	29k	2	2
bicep	33k	3	1
leg	43k	4	5
upper arm	33k	4	6
soft robot wheel	3k	2	2
soft robot hand	18k	7	9
cartoon skull	48k	2	4
chest-arm-back	47k	5	11
Fruit picker	289k	6	6
Quadropus	220k	5	8

Table 3: *Musculoskeletal models simulated with EMU.*

EMU exhibits excellent performance when compared to the state-of-the-art open source FEM solver [Lev19]. Our FEM algorithm uses Stable Neo-Hookean elasticity from [SGK18] solved using the open source Pardiso solver, [DCDBK*16, VCKS17, KFS18]. We compare the performance of both algorithms in terms of scaling with respect to mesh size and scaling with respect to number of available CPU cores. For testing we measured convergence of both methods by checking if the change in energy of an iteration was $< 1e-2$. We found this sufficient to produce results with excellent visual fidelity. Single-core scaling tests were performed on Intel Core i7-6700HQ CPU (2.60GHz). Multi-core tests were performed on a Dual Intel Xeon Gold 5120 (2.20GHz). Scaling tests were done on our simple fusiform muscle (Figure 20). Every mesh, including the 600k tetrahedral mesh ran without memory issues on a 16GB RAM laptop. Since we only use fixed size dense matrices, memory usage is not a problem in our simulations.

On a single core machine, EMU scales better than state-of-the-art FEM, as a function of number of mesh tetrahedra (Figure 20). Our most intensive operations are a sparse back substitution required to solve Eq. 4 and a dense matrix inversion required to compute the Woodbury identity. For a given example, increasing mesh resolution does not have a large effect on the spectral characteristics of the Hessian in Eq. 4. Thus k (Eq. 17) typically stays constant so the cost of the required dense solve remains fixed. The effect of this is that EMU is faster than FEM for all but the smallest examples and, for medium to large meshes, impressively so – exhibiting speedups of over 20×. This is particularly impressive when one considers that EMU is not reducing the solution space in any way; it is solving the same problem as the FEM discretization. Even though not specifically designed for simulating isotropic, homogenous materials, EMU retains its performance advantage over FEM. We tested the performance on the Stanford Bunny mesh, up to 40k tetrahedra and observed a *5-6x* speed-up over state-of-the-art FEM solver.

EMU also parallelizes well. We observe a further *3x* performance improvement by running EMU on a 12 core machine (hyperthreading disabled). As discussed, the EMU Hessian update is extremely

parallizable. The bottleneck in our implementation comes from the limited ability of the linear solver (Pardiso) to efficiently parallelize the backsolves needed to minimize the ACAP energy. Exploiting other solvers with better parallel scaling would improve EMU performance even further.

Our contact-aware solver is able to simulate muscles in close contact and allows EMU to exert force along relatively complex muscle paths. The top row of Figure 10 shows two fusiform muscles in a side-by-side configuration. The muscles are isometrically contracted (the bottom bone is fixed) and then the bone is released allowing the muscles to fully contract. Our contact solver prevents interpenetrations in both cases. The bottom row of Figure 10 shows a muscle with a sharp bend. Contracting the fibers in this muscle allows it to exert force around a corner and apply a vertical force to the square bone at its end-point. Again, the EMU contact solver prevents the muscle from contracting into the underlying bones.

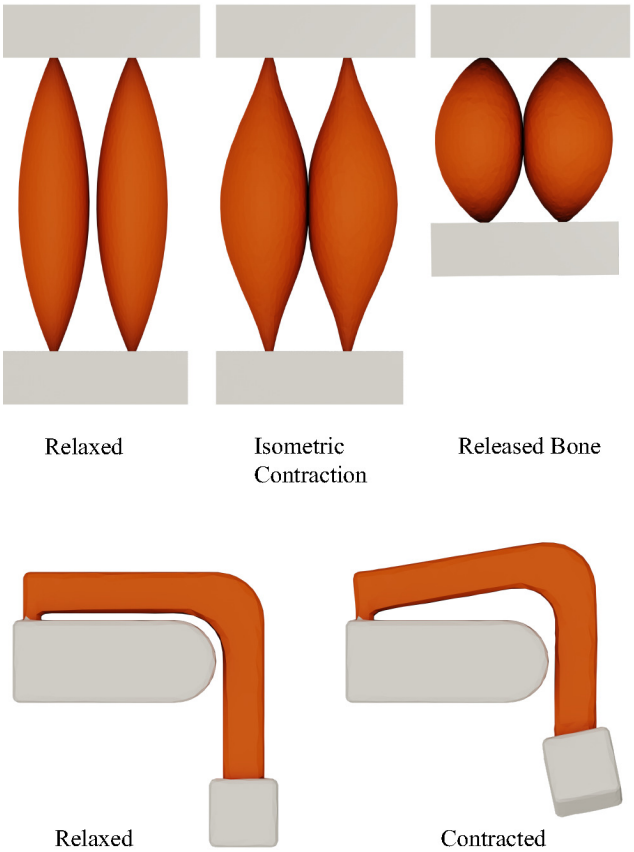

Figure 10: *Examples of muscles in contact, simulated using EMU. Top: EMU prevents side-by-side muscles from interpenetrating under both isometric, and full motion contraction. Bottom: A muscle with a sharp bend pulls a square bone around a corner. (Video: 1m05s, 1m50s)*

Next we show that EMU can generate realistic large scale muscle-first motions by simulating an elephant trunk (Figure 11). The cross-section of the trunk is divided into quarters with each quarter being an independent muscle. Fibers run along the length of the trunk and contracting various muscles causes the trunk to bend. We simulate the canonical elephant feeding motion – the elephant reaches up to grab food, then bends the trunk in the opposite direction to bring the food to its mouth.

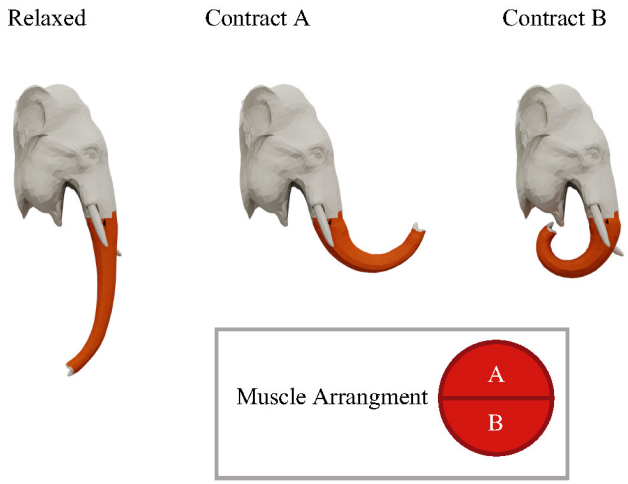

Figure 11: *A simulated elephant trunk. EMU can efficiently generate realistic, large deformation motions of this elephant trunk by contracting the appropriate muscles. Bottom: Muscle arrangement of the trunk. Muscle fibers run along the trunks length. (Video: 2m05s)*

Figure 12: *Simulating the motion of the humerus, radius and ulna induced by contracting the bicep. The bicep is a biarticular muscle which connects the shoulder directly to the forearm, skipping the humerus entirely. EMU can handle such complicated muscles*

One of the advantages of the EMU deformation gradient formalism is the ease with which joints can be incorporated. Figure 12 shows how the contraction of the bicep drives the motion of the humerus, radius and ulna. This is because the bicep is biarticular – it connects the shoulder directly to the forearm. Motion of the humerus results from the forearm being driven by the muscle. EMU effortlessly handles complicated muscles such as this and is able to properly transmit forces from the contracting muscle and

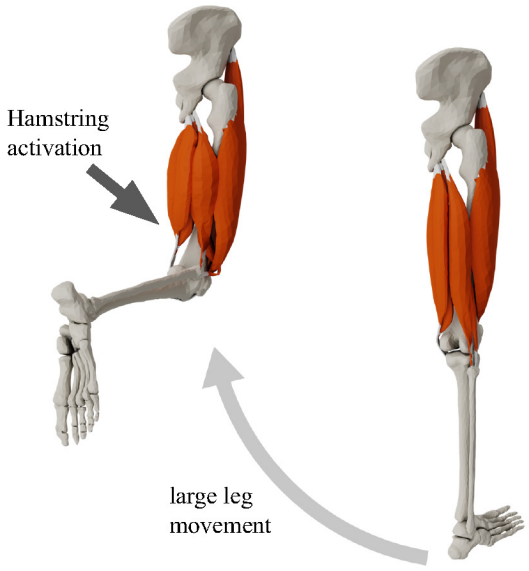

Figure 13: *Simulating the motion of a leg, induced by contracting the hamstring. (Video: 0m41s)*

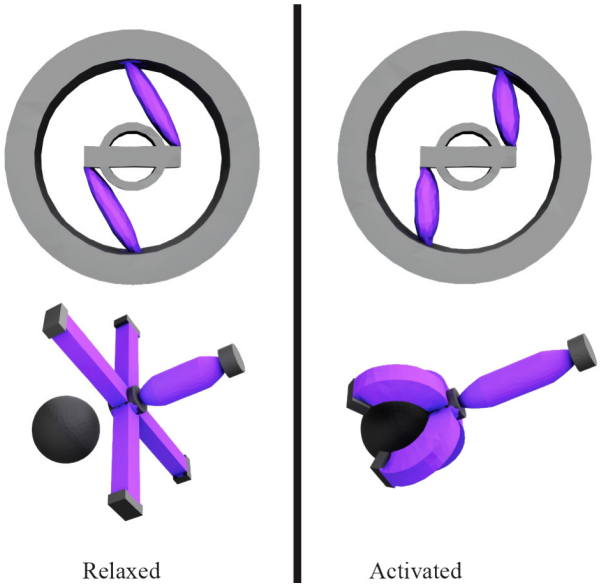

Figure 14: *EMU can also be used to simulate pneumatically actuated mechanical systems. Here we use EMU's ability to simulate deformable objects, joints and rigid bodies, to simulate (Top) this soft mechanism that can rotate its outer ring when its actuators are contracted and (Bottom) a soft robotic grasper. (Video: 2m25s)*

through both the elbow (modelled as a hinge joint) and the shoulder (modelled as a spherical joint). Figure 13 shows a simulation of a contracted hamstring which drives the large scale motion of a multi-muscle leg. This shows the ability of EMU to generate realsitic, muscle-first motion in the presence of multiple muscles, joints, tendons and bones.

EMU has uses beyond biomenchanical simulation. In Figure 14 (Top) we simulate a pneumatically actuated mechanism. This mechanism can rotate its outer ring when its pneumatic actuators are activated. EMU's ability to handle deformable and rigid bodies, connected with joints, is perfect for such applications. Figure 14 (Bottom) shows the simulation of a soft robotic gripper which reaches down and grasps a ball. Next, we present two more examples. In Figure 15, a robot which can pick ripe fruits for juicing simulated by 289k tets at 13.3 seconds per frame. And in Figure 16 we present a squid-like creature discretized by 200k tets simulated at 127.9 seconds per frame.

Finally we turn our attention to more complex biomechanical models. Figure 18 demonstrates EMU's ability to generate simulations using realisitc biomechanical activation sequences. In this figure we simulate a bicep contraction, followed by a tricep contraction. This first flexes the elbow and then hyperextends. This example shows off all of EMUs features, its efficiency, and its ability to seamlessly animate bones, tendons and muscles to generate muscle-first bulk musculoskeletal motion. Figure 19 illustrates the use of EMU to generate muscle-first head motion. This cartoon head roll is completely driven by muscle actuations of the four neck muscles. EMU allows us to flex the muscles of this complex upper body model (Figure 1) to get a biomechanically feasible pump. Finally we can also use EMU to add muscle motion on top of scripted

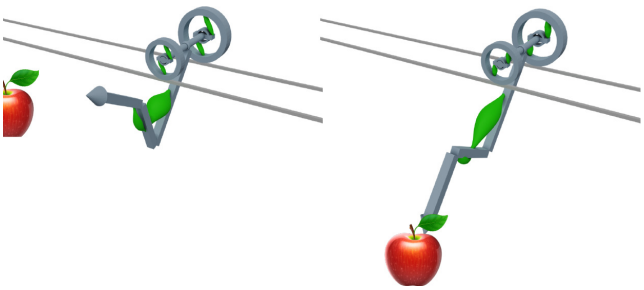

Figure 15: *A 289k tetrahedron fruit picking robot which rolls back and forth along the rails, grabs fruits and drops them.(Video: 2m19s)*

bones, like this arm mesh (Figure 17) which is animated to rotate through a large motion, and creates natural looking complimentry muscle motion.

5. Conclusion and Future Work

We have presented a new, efficient algorithm for bulk musculoskeletal simulation. Our algorithm is a multi-objective, discontinuous iterative approach to finite element simulation which uses a novel, minimal energy penalty to enforce continuity. We demonstrate how this approach leads to the construction of an efficient algorithm for musculoskeletal simulation which at run time requires

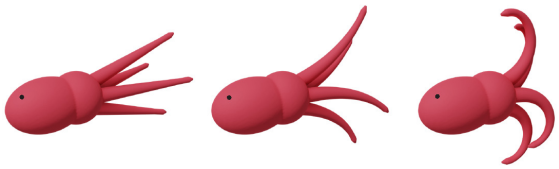

Figure 16: *EMU used to simulate the motion of a squid-like animal with 220k tetrahedrons. (Video: 0m0s)*

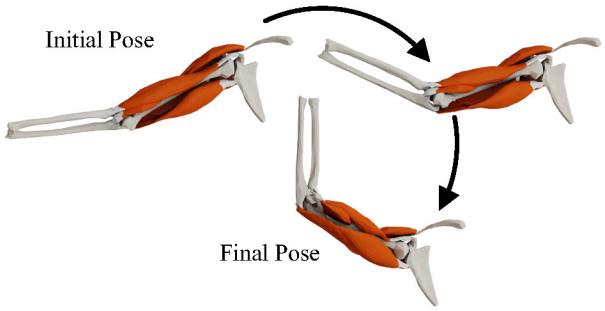

Figure 17: *The skeletal motion of this arm is scripted by an artist and we use EMU to add compelling muscle deformations. (Video: 2m12s)*

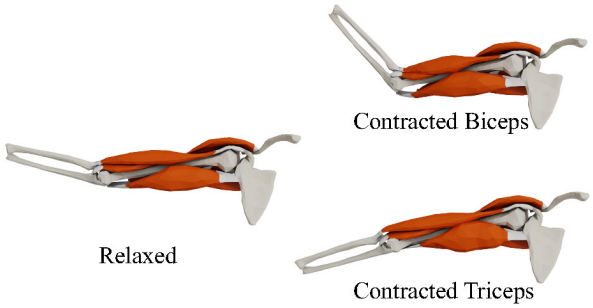

Figure 18: *A simulation using a full musculoskeletal model of a human arm. We use EMU to simulate a bicep contraction, followed by a tricep contraction to hyper-extend the elbow. (Video: 1m56s)*

only the inversion of a small dense matrix, sparse back substitution and the factorization of a block diagonal matrix (all extremely fast operations).

Furthermore we show how to incorporate both bones and simple tendons into the method without needing to resort to specialized approaches such as coupled rigid body simulations or line-of-force methods. To our knowledge, we are the first to demonstrate such a holistic approach to bulk musculoskeletal simulation (as all previous approaches avoid tendonous attachment points as in Figure 12 and Figure 13).

Although EMU has performance benefits over FEM and our results show visually appealing deformations, for simulations where accuracy is essential, EMU falls short of FEM due to the inexactness introduced by the α parameter. However, we show that by using our heuristic to find an optimal α, we can better approximate the deformations produced by FEM than other contemporary methods such as ADMM. Additionally, unlike ADMM and other projection based methods, EMU allows the use of any material model and muscle activation model.

We believe our method will have immediate application for character animations and control, but we are most excited about the areas of future work EMU opens up, both on the numerical optimization front and in biomechanical simulations. Incorporating more complicated tendon routing and sliding is a crucial area of future work. Previous approaches for efficient sliding work on 1D [SJLP11] and 2D [WPLS18] geometries, but volumetric tendon sliding is unaddressed. Co-dimensional simulation could also be explored, allowing the coupling of 1D, 2D and 3D elements inside of our framework. Since we only need to represent the deformation of such elements our method should be well suited to this. We could also improve the performance of EMU by exploring the use of fast solvers in our continuity energy (akin to using specialized solvers in the pressure projection step of a fluid simulation).

Modeling is a crucial area of future work that would benefit from the availability of an algorithm like EMU. Idealized joint constraints restrict the motions that can be achieved by any simulator, no matter how fast or how robust. Using high-performance simulators like EMU will enable us to model connections between bones using ligaments and other soft tissue structures, thus capturing more natural motions. Building tools that can produce these detailed volumetric representations of the human body will be increasingly important moving forward.

Acknowledgements

This work is funded in party by National Science Foundation (CAREER-1846368), NSERC Discovery (RGPIN-2017-05524, RGPIN-2017-05235, RGPAS-2017-507938), Connaught Fund (503114), CFI-JELF Fund, Accelerator (RGPAS-2017-507909), New Frontiers of Research Fund (NFRFE–201), the Ontario Early Research Award program, the Canada Research Chairs Program, the Fields Centre for Quantitative Analysis and Modelling and gifts by Adobe Systems, Autodesk and MESH Inc. We thank John Kanji, and Josh Holinaty for help with designing figures; Rinat Abdrashitov for his video editing skills; Josh Holinaty for lending us his beautiful voice in the video; Sarah Kushner, Honglin Chen, Abhishek Madan, Hsueh-Ti Derek Liu and Darren Moore for proofreading; John Hancock for IT support; anonymous reviewers for their helpful comments and suggestions.

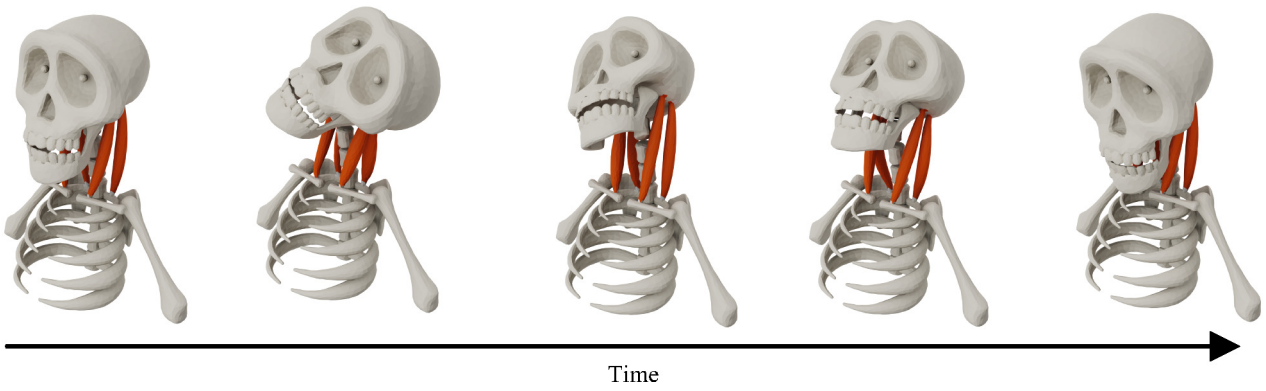

Figure 19: *This cartoon head roll is completely driven by actuating the four neck muscles of this model. The neck joint is hollistically simulated using EMU. (Video: 0m08s)*

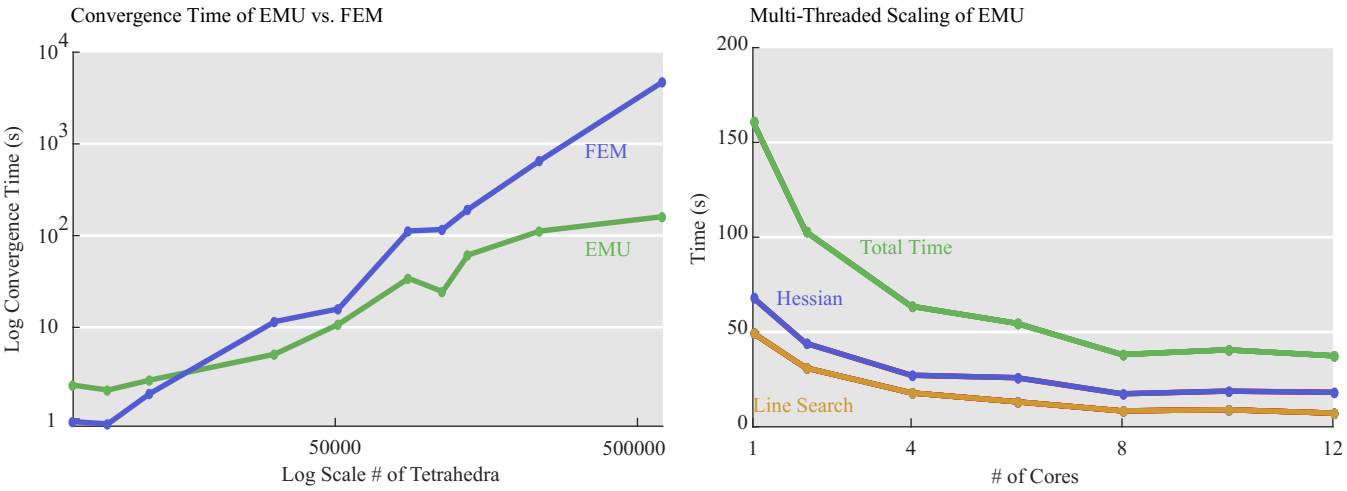

Figure 20: *Left: single core scaling of EMU and standard FEM as a function of mesh size. Right: multi-core scaling. EMU is faster than standard FEM for all but the smallest meshes and exhibits better asymptotic behavior. The EMU quasi-static solve scales well with number of cores. When using 12 cores the bottleneck in the code is the sparse back solve required for ACAP energy calculation which could be optimized further using more efficient solvers.*

References

[AKJ08] AN S. S., KIM T., JAMES D. L.: Optimizing cubature for efficient integration of subspace deformations. *ACM Trans. Graph. 27*, 5 (Dec. 2008), 165:1–165:10. URL: http://doi.acm.org/10.1145/1409060.1409118, doi:10.1145/1409060.1409118. 2

[ARM*19] ANGLES B., REBAIN D., MACKLIN M., WYVILL B., BARTHE L., LEWIS J., VON DER PAHLEN J., IZADI S., VALENTIN J., BOUAZIZ S., ET AL.: Viper: Volume invariant position-based elastic rods. *Proceedings of the ACM on Computer Graphics and Interactive Techniques 2*, 2 (2019), 1–26. 3, 4

[BC80] BATHE K. J., CIMENTO A. P.: Some practical procedures for the solution of nonlinear finite element equations. *Computer Methods in Applied Mechanics and Engineering 22*, 1 (1980), 59 – 85. URL: http://www.sciencedirect.com/science/article/pii/0045782580900511, doi:https://doi.org/10.1016/0045-7825(80)90051-1. 5

[BEH18] BRANDT C., EISEMANN E., HILDEBRANDT K.: Hyper-reduced projective dynamics. *ACM Trans. Graph. (Proc. of SIGGRAPH) 37*, 4 (2018), 80:1–80:13. URL: http://graphics.tudelft.nl/Publications-new/2018/BEH18. 2

[BML*14] BOUAZIZ S., MARTIN S., LIU T., KAVAN L., PAULY M.: Projective dynamics: Fusing constraint projections for fast simulation. *ACM Trans. Graph. 33*, 4 (July 2014), 154:1–154:11. URL: http://doi.acm.org/10.1145/2601097.2601116, doi:10.1145/2601097.2601116. 3

[BPGK06] BOTSCH M., PAULY M., GROSS M., KOBBELT L.: Primo: Coupled prisms for intuitive surface modeling. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing* (Aire-la-Ville, Switzerland, Switzerland, 2006), SGP '06, Eurographics Association,

pp. 11–20. URL: http://dl.acm.org/citation.cfm?id=1281957.1281959. 2

[Bra77] BRANDT A.: Multi-level adaptive solutions to boundary-value problems. *Mathematics of computation 31*, 138 (1977), 333–390. 2

[BZ11] BARBIČ J., ZHAO Y.: Real-time large-deformation substructuring. *ACM Trans. on Graphics (SIGGRAPH 2011) 30*, 4 (2011), 91:1–91:7. 2

[CLMK17] CHEN D., LEVIN D. I. W., MATUSIK W., KAUFMAN D. M.: Dynamics-aware numerical coarsening for fabrication design. *ACM Trans. Graph. 36*, 4 (July 2017), 84:1–84:15. URL: http://doi.acm.org/10.1145/3072959.3073669, doi:10.1145/3072959.3073669. 2

[DAA*07] DELP S. L., ANDERSON F. C., ARNOLD A. S., LOAN P., HABIB A., JOHN C. T., GUENDELMAN E., THELEN D. G.: Opensim: open-source software to create and analyze dynamic simulations of movement. *IEEE transactions on biomedical engineering 54*, 11 (2007), 1940–1950. 2

[DCDBK*16] DE CONINCK A., DE BAETS B., KOUROUNIS D., VERBOSIO F., SCHENK O., MAENHOUT S., FOSTIER J.: Needles: Toward large-scale genomic prediction with marker-by-environment interaction. 543–555. URL: http://dx.doi.org/10.1534/genetics.115.179887, arXiv:http://www.genetics.org/content/203/1/543.full.pdf, doi:10.1534/genetics.115.179887. 9

[FGBP11] FAURE F., GILLES B., BOUSQUET G., PAI D. K.: Sparse meshless models of complex deformable solids. *ACM Trans. Graph. 30*, 4 (July 2011), 73:1–73:10. URL: http://doi.acm.org/10.1145/2010324.1964968, doi:10.1145/2010324.1964968. 2

[FLP14] FAN Y., LITVEN J., PAI D. K.: Active volumetric musculoskeletal systems. *ACM Trans. Graph. 33*, 4 (July 2014), 152:1–152:9. URL: http://doi.acm.org/10.1145/2601097.2601215, doi:10.1145/2601097.2601215. 2

[FMD*19] FULTON L., MODI V., DUVENAUD D., LEVIN D., JACOBSON A.: Latent-space dynamics for reduced deformable simulation. 2

[GBFP11] GILLES B., BOUSQUET G., FAURE F., PAI D. K.: Frame-based elastic models. *ACM Trans. Graph. 30*, 2 (Apr. 2011), 15:1–15:12. URL: http://doi.acm.org/10.1145/1944846.1944855, doi:10.1145/1944846.1944855. 2

[GJ*19] GUENNEBAUD G., JACOB B., ET AL.: Eigen v3. http://eigen.tuxfamily.org, 2019. 6

[GvdPvdS13] GEIJTENBEEK T., VAN DE PANNE M., VAN DER STAPPEN A. F.: Flexible muscle-based locomotion for bipedal creatures. *ACM Trans. Graph. 32*, 6 (Nov. 2013), 206:1–206:11. URL: http://doi.acm.org/10.1145/2508363.2508399, doi:10.1145/2508363.2508399. 2

[HTK*04] HEIDELBERGER B., TESCHNER M., KEISER R., MÜLLER M., GROSS M. H.: Consistent penetration depth estimation for deformable collision response. In *VMV* (2004), vol. 4, Citeseer, pp. 339–346. 7

[HZG*18] HU Y., ZHOU Q., GAO X., JACOBSON A., ZORIN D., PANOZZO D.: Tetrahedral meshing in the wild. *ACM Trans. Graph. 37*, 4 (July 2018), 60:1–60:14. URL: http://doi.acm.org/10.1145/3197517.3201353, doi:10.1145/3197517.3201353. 8

[IKKP17] ICHIM A.-E., KADLEČEK P., KAVAN L., PAULY M.: Phace: Physics-based face modeling and animation. *ACM Trans. Graph. 36*, 4 (July 2017), 153:1–153:14. URL: http://doi.acm.org/10.1145/3072959.3073664, doi:10.1145/3072959.3073664. 2, 3

[JBK*12] JACOBSON A., BARAN I., KAVAN L., POPOVIĆ J., SORKINE O.: Fast automatic skinning transformations. *ACM Trans. Graph. 31*, 4 (July 2012), 77:1–77:10. URL: http://doi.acm.org/10.1145/2185520.2185573, doi:10.1145/2185520.2185573. 3

[JP99] JAMES D. L., PAI D. K.: Artdefo: Accurate real time deformable objects. In *Proc. SIGGRAPH* (1999). 6

[KDGI19] KIM T., DE GOES F., IBEN H.: Anisotropic elasticity for inversion-safety and element rehabilitation. *ACM Transactions on Graphics (TOG) 38*, 4 (2019), 1–15. 4

[KFS18] KOUROUNIS D., FUCHS A., SCHENK O.: Towards the next generation of multiperiod optimal power flow solvers. *IEEE Transactions on Power Systems PP*, 99 (2018), 1–10. URL: https://doi.org/10.1109/TPWRS.2017.2789187, doi:10.1109/TPWRS.2017.2789187. 9

[KMBG08] KAUFMANN P., MARTIN S., BOTSCH M., GROSS M.: Flexible simulation of deformable models using discontinuous galerkin fem. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2008), SCA '08, Eurographics Association, pp. 105–115. URL: http://dl.acm.org/citation.cfm?id=1632592.1632608. 2

[KPMP*17] KIM M., PONS-MOLL G., PUJADES S., BANG S., KIM J., BLACK M. J., LEE S.-H.: Data-driven physics for human soft tissue animation. *ACM Trans. Graph. 36*, 4 (July 2017), 54:1–54:12. URL: http://doi.acm.org/10.1145/3072959.3073685, doi:10.1145/3072959.3073685. 2

[LBK17] LIU T., BOUAZIZ S., KAVAN L.: Quasi-newton methods for real-time simulation of hyperelastic materials. *ACM Transactions on Graphics (TOG) 36*, 3 (2017), 1–16. 3, 4

[Lev19] LEVIN D.: Gauss, Mar. 2019. URL: https://github.com/dilevin/GAUSS. 9

[LGL*19] LI M., GAO M., LANGLOIS T., JIANG C., KAUFMAN D. M.: Decomposed optimization time integrator for large-step elastodynamics. *ACM Transactions on Graphics 38*, 4 (2019). 3, 4

[LGMP11] LEVIN D. I., GILLES B., MADLER B., PAI D. K.: Extracting skeletal muscle fiber fields from noisy diffusion tensor data. *Medical Image Analysis 15*, 3 (2011), 340 – 353. URL: http://www.sciencedirect.com/science/article/pii/S136184151100020X, doi:https://doi.org/10.1016/j.media.2011.01.005. 2

[LPKL14] LEE Y., PARK M. S., KWON T., LEE J.: Locomotion control for many-muscle humanoids. *ACM Trans. Graph. 33*, 6 (Nov. 2014), 218:1–218:11. URL: http://doi.acm.org/10.1145/2661229.2661233, doi:10.1145/2661229.2661233. 2

[LST09] LEE S.-H., SIFAKIS E., TERZOPOULOS D.: Comprehensive biomechanical modeling and simulation of the upper body. *ACM Trans. Graph. 28*, 4 (Sept. 2009), 99:1–99:17. URL: http://doi.acm.org/10.1145/1559755.1559756, doi:10.1145/1559755.1559756. 2

[LT06] LEE S.-H., TERZOPOULOS D.: Heads up!: Biomechanical modeling and neuromuscular control of the neck. *ACM Trans. Graph. 25*, 3 (July 2006), 1188–1198. URL: http://doi.acm.org/10.1145/1141911.1142013, doi:10.1145/1141911.1142013. 2

[LYP*18] LEE S., YU R., PARK J., AANJANEYA M., SIFAKIS E., LEE J.: Dexterous manipulation and control with volumetric muscles. *ACM Trans. Graph. 37*, 4 (July 2018), 57:1–57:13. URL: http://doi.acm.org/10.1145/3197517.3201330, doi:10.1145/3197517.3201330. 2, 3, 4

[MP99] MAGANARIS C. N., PAUL J. P.: In vivo human tendon mechanical properties. *The Journal of physiology 521*, 1 (1999), 307–313. 3

[NOB16] NARAIN R., OVERBY M., BROWN G. E.: ADMM ⊇ projective dynamics: Fast simulation of general constitutive models. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2016), SCA '16, Eurographics Association, pp. 21–28. URL: http://dl.acm.org/citation.cfm?id=2982818.2982822. 3

[PBH15] PAN Z., BAO H., HUANG J.: Subspace dynamic simulation using rotation-strain coordinates. *ACM Trans. Graph. 34*, 6 (Oct. 2015), 242:1–242:12. URL: http://doi.acm.org/10.1145/2816795.2818090, doi:10.1145/2816795.2818090. 2

[PMRMB15] PONS-MOLL G., ROMERO J., MAHMOOD N., BLACK M. J.: Dyna: A model of dynamic human shape in motion. *ACM Trans. Graph. 34*, 4 (July 2015), 120:1–120:14. URL: http://doi.acm.org/10.1145/2766993, doi:10.1145/2766993. 2

[RKSZ98] RHO J.-Y., KUHN-SPEARING L., ZIOUPOS P.: Mechanical properties and the hierarchical structure of bone. *Medical engineering & physics 20*, 2 (1998), 92–102. 3

[SCM*18] SELLÁN S., CHENG H. Y., MA Y., DEMBOWSKI M., JACOBSON A.: Solid geometry processing on deconstructed domains. *CoRR abs/1807.00866* (2018). URL: http://arxiv.org/abs/1807.00866, arXiv:1807.00866. 8

[SGK18] SMITH B., GOES F. D., KIM T.: Stable neo-hookean flesh simulation. *ACM Trans. Graph. 37*, 2 (Mar. 2018), 12:1–12:15. URL: http://doi.acm.org/10.1145/3180491, doi:10.1145/3180491. 2, 3, 5, 9

[SJLP11] SUEDA S., JONES G. L., LEVIN D. I., PAI D. K.: Large-scale dynamic simulation of highly constrained strands. *ACM Transactions on Graphics 30*, 4 (Aug 2011), 39:1–39:9. 12

[SKP08] SUEDA S., KAUFMAN A., PAI D. K.: Musculotendon simulation for hand animation. *ACM Transactions on Graphics 27*, 3 (Aug 2008), 83:1–83:8. 2

[SLST14] SI W., LEE S.-H., SIFAKIS E., TERZOPOULOS D.: Realistic biomechanical simulation and control of human swimming. *ACM Trans. Graph. 34*, 1 (Dec. 2014), 10:1–10:15. URL: http://doi.acm.org/10.1145/2626346, doi:10.1145/2626346. 2

[SSB*15] SACHDEVA P., SUEDA S., BRADLEY S., FAIN M., PAI D. K.: Biomechanical simulation and control of hands and tendinous systems. *ACM Trans. Graph. 34*, 4 (July 2015), 42:1–42:10. URL: http://doi.acm.org/10.1145/2766987, doi:10.1145/2766987. 2

[SSF08] SHINAR T., SCHROEDER C., FEDKIW R.: Two-way coupling of rigid and deformable bodies. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2008), SCA '08, Eurographics Association, pp. 95–103. URL: http://dl.acm.org/citation.cfm?id=1632592.1632607. 3

[TBHF03] TERAN J., BLEMKER S., HING V. N. T., FEDKIW R.: Finite volume methods for the simulation of skeletal muscle. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2003), SCA '03, Eurographics Association, pp. 68–74. URL: http://dl.acm.org/citation.cfm?id=846276.846285. 2, 3, 5

[TJM15] TAMSTORF R., JONES T., MCCORMICK S. F.: Smoothed aggregation multigrid for cloth simulation. *ACM Trans. Graph. 34*, 6 (Oct. 2015), 245:1–245:13. URL: http://doi.acm.org/10.1145/2816795.2818081, doi:10.1145/2816795.2818081. 2

[TPBF87] TERZOPOULOS D., PLATT J., BARR A., FLEISCHER K.: Elastically deformable models. In *Computer Graphics* (1987), vol. 21, pp. 205–214. 1

[TSB*05] TERAN J., SIFAKIS E., BLEMKER S. S., NG-THOW-HING V., LAU C., FEDKIW R.: Creating and simulating skeletal muscle from the visible human data set. *IEEE Transactions on Visualization and Computer Graphics 11*, 3 (May 2005), 317–328. URL: https://doi.org/10.1109/TVCG.2005.42, doi:10.1109/TVCG.2005.42. 2

[TSIF05] TERAN J., SIFAKIS E., IRVING G., FEDKIW R.: Robust quasistatic finite elements and flesh simulation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2005), SCA '05, ACM, pp. 181–190. URL: http://doi.acm.org/10.1145/1073368.1073394, doi:10.1145/1073368.1073394. 2

[VCKS17] VERBOSIO F., CONINCK A. D., KOUROUNIS D., SCHENK O.: Enhancing the scalability of selected inversion factorization algorithms in genomic prediction. *Journal of Computational Science 22*, Supplement C (2017), 99 – 108. URL: https://doi.org/10.1016/j.jocs.2017.08.013. 9

[WHDK12] WANG J. M., HAMNER S. R., DELP S. L., KOLTUN V.: Optimizing locomotion controllers using biologically-based actuators and objectives. *ACM Trans. Graph. 31*, 4 (July 2012), 25:1–25:11. URL: http://doi.acm.org/10.1145/2185520.2185521, doi:10.1145/2185520.2185521. 2

[WN99] WRIGHT S. J., NOCEDAL J.: *Numerical optimization*. 1999. 3, 5, 6, 8

[WPLS18] WEIDNER N. J., PIDDINGTON K., LEVIN D. I., SUEDA S.: Eulerian-on-Lagrangian cloth simulation. *ACM Trans. on Graph. 37*, 4 (August 2018), 50:1–50:11. 12

[WWB*19] WANG Y., WEIDNER N. J., BAXTER M. A., HWANG Y., KAUFMAN D. M., SUEDA S.: REDMAX: Efficient & flexible approach for articulated dynamics. *ACM Transactions on Graphics 38*, 4 (July 2019), 104:1–104:10. 3

[XB16] XU H., BARBIČ J.: Pose-space subspace dynamics. *ACM Trans. on Graphics (SIGGRAPH 2016) 35*, 4 (2016). 2, 6

[YZX*04] YU Y., ZHOU K., XU D., SHI X., BAO H., GUO B., SHUM H.-Y.: Mesh editing with poisson-based gradient field manipulation. In *ACM SIGGRAPH 2004 Papers*. 2004, pp. 644–651. 4

[ZBK18] ZHU Y., BRIDSON R., KAUFMAN D. M.: Blended cured quasi-newton for distortion optimization. *to appear ACM Trans. on Graphics (SIGGRAPH 2018)* (2018). 3, 4, 5

[ZSTB10] ZHU Y., SIFAKIS E., TERAN J., BRANDT A.: An efficient multigrid method for the simulation of high-resolution elastic solids. *ACM Trans. Graph. 29*, 2 (Apr. 2010), 16:1–16:18. URL: http://doi.acm.org/10.1145/1731047.1731054, doi:10.1145/1731047.1731054. 2