Simulation of Complex Nonlinear Elastic Bodies using Lattice Deformers

Taylor Patterson
University of Wisconsin-Madison

Nathan Mitchell
University of Wisconsin-Madison

Eftychios Sifakis
University of Wisconsin-Madison

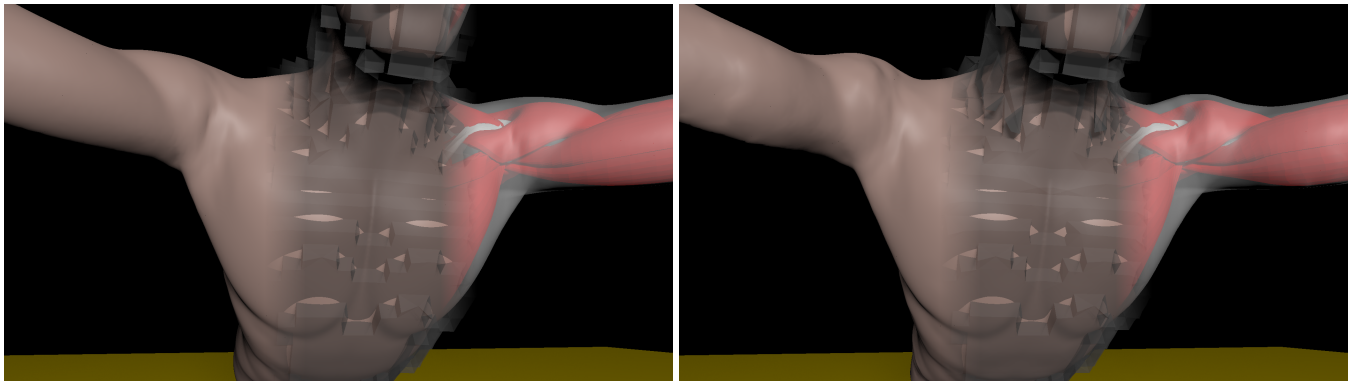

Figure 1: *Anatomic simulation with skin, deformer lattice and embedded muscles shown. Left: Muscles inactive, Right: Muscles fully flexed.*

Abstract

Lattice deformers are a popular option for modeling the behavior of elastic bodies as they avoid the need for conforming mesh generation, and their regular structure offers significant opportunities for performance optimizations. Our work expands the scope of current lattice-based elastic deformers, adding support for a number of important simulation features. We accommodate complex nonlinear, optionally anisotropic materials while using an economical one-point quadrature scheme. Our formulation fully accommodates near-incompressibility by enforcing accurate nonlinear constraints, supports implicit integration for large time steps, and is not susceptible to locking or poor conditioning of the discrete equations. Additionally, we increase the accuracy of our solver by employing a novel high-order quadrature scheme on lattice cells overlapping with the model boundary, which are treated at sub-cell precision. Finally, we detail how this accurate boundary treatment can be implemented at a minimal computational premium over the cost of a voxel-accurate discretization. We demonstrate our method in the simulation of complex musculoskeletal human models.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

Keywords: nonlinear elasticity, incompressibility, cut-cell methods

Links: DL PDF

1 Introduction

Simulation of elastic deformable models is ubiquitous in computer graphics and remains a vibrant area of research. Algorithmic techniques for deformable body simulation, pioneered by Terzopoulos et al [1987] have attained a significant level of maturity, leading to broad adoption in visual effects, games, virtual environments and biomechanics applications. However, numerous theoretical and technical challenges remain. Research efforts often emphasize improved computational performance for cost-conscious interactive applications. Simulation of complex materials and concerns about accuracy and fidelity, especially in biomechanics applications, place additional strain on simulation techniques. Finally, ease of use and deployment in production environments is an important trait that scholarly research work needs to be sensitive to. Our paper proposes a grid-based simulation technique with a number of original components that enhance performance and parallelism, natively accommodate complex materials (including skin, flesh and muscles) while offering the simple and familiar front-end of a lattice deformer for easy integration into an animation pipeline.

Lattice-based volumetric deformers are popular components in both physics-based and procedural animation techniques. In the case of physics-based simulation, one of their key advantages is that they avoid having to construct a simulation-ready conforming volume mesh, which is a delicate preprocessing task often requiring supervision and fine-tuning. Another crucial benefit is that the regularity of such data structures enables aggressive performance optimizations as vividly demonstrated by shape matching techniques [Rivers and James 2007]. Cartesian lattices have also been leveraged to accelerate performance in physics-based approaches, albeit predominantly for simple models such as linear or corotated elasticity [Müller et al. 2004; Georgii and Westermann 2008; McAdams et al. 2011]. Prior graphics work, however, has not demonstrated such aggressive performance gains from lattice-based discretizations when highly nonlinear, anisotropic or incompressible materials are involved. In part, this is attributed to the fact that simulation of complex materials commands an increased level of attention to issues of robust convergence and reliable treatment of incompressibility. Mature solutions to these concerns have predominantly been demonstrated in the context of specific discretizations (e.g. explicit tetrahedral meshes) where regularity of data structures, compactness of memory footprint and parallelization/vectorization potential were not inherently emphasized. Furthermore, as applications requiring the use of complex materials are also likely to emphasize geometric accuracy, they often opt for conforming mesh discretizations due to their superior performance in capturing intricate boundary features, even if their computational cost is higher.

ACM Reference Format
Patterson, T., Mitchell, N., Sifakis, E. 2012. Simulation of complex nonlinear elastic bodies using lattice deformers. *ACM Trans. Graph. 31* 6, Article 197 (November 2012), 10 pages.
DOI = 10.1145/2366145.2366216 http://doi.acm.org/10.1145/2366145.2366216.

We propose a lattice-based formulation that jointly addresses the aforementioned challenges. Our method accommodates complex nonlinear materials, including incompressible and anisotropic variants, without sacrificing robustness or performance. Intricate, irregular model geometries are accommodated by treating boundary cells at sub-element precision. Nevertheless, this is performed in a way that preserves regularity of data structures and at a minimal performance premium over voxel-accurate discretizations. We demonstrate a multithreaded and vectorized solver on complex musculoskeletal simulations of human anatomy. Our contributions include:

- A robust method for handling materials with any degree of incompressibility, based on a mixed variational formulation. Specific to our approach is support for true nonlinear volume preservation constraints instead of simplified approximations.
- A novel second order accurate, yet efficient and vectorizable quadrature scheme for volume integrals, enabling a sub-voxel accurate discretization of elasticity at the model boundary.
- A defect correction procedure for solving the sub-voxel accurate discrete elasticity equations while practically paying only the cost necessary for a voxel-accurate discretization.
- A new data organization scheme for storing state variables and intermediate solver data, facilitating aggressive SIMD accelerations and lock-free, load balanced multithreading.

The technical portion of our paper is structured as follows: In section 2 we detail how the discrete form of the governing equations is obtained and explain our treatment of incompressibility. In section 3 we replace complex integrals in the discrete equations with simpler numerical expressions, better suited for computer implementation; this section introduces our sub-voxel accurate treatment of boundaries. In section 4 we solve the nonlinear discrete equations using a high-order defect correction procedure and a symmetric indefinite Krylov solver for the linearized system. Section 5 lists several crucial implementation considerations, including our SIMD- and thread-optimized data organization scheme. We note that we shall defer the discussion of relevant existing research until later in our technical exposition, where such contributions can be more appropriately contrasted with our proposed approach.

2 Elasticity and discretization

We start by reviewing the physical principles that govern the motion of an elastic deformable body. Let $\phi : \Omega \rightarrow \mathbf{R}^3$ be the deformation function which maps a material point $\vec{X} = (X, Y, Z)$ to its deformed location $\vec{x} = (x, y, z) = \phi(\vec{X})$, and $\mathbf{F}(\vec{X}) = \partial\phi(\vec{X})/\partial\vec{X}$ denote the deformation gradient. In order to simulate the deformation of a body with a specific material composition, we need a quantitative description of how this material reacts to a given deformation. For hyperelastic materials this is derived from a strain energy density function $\Psi(\mathbf{F})$ which can be integrated over the entire body to measure the total energy $E[\phi] = \int_\Omega \Psi(\mathbf{F}) d\vec{X}$. In these expressions $\phi(\vec{X})$ is an arbitrary deformation field; however, for numerical simulation we only encode the deformation map via discrete values $\vec{x}_i = \phi(\vec{X}_i)$ sampled at prescribed locations $\{\vec{X}_i\}_{i=1\ldots N}$. Using those, we reconstruct discretized versions of the deformation field, the deformation gradient and the elastic energy, as follows:

$$\phi(\vec{X}; \boldsymbol{x}) = \sum_i \vec{x}_i \mathcal{N}_i(\vec{X}) \quad (1)$$

$$\mathbf{F}(\vec{X}; \boldsymbol{x}) = \partial\phi(\vec{X}; \boldsymbol{x})/\partial\vec{X} \quad (2)$$

$$E(\boldsymbol{x}) = \int_\Omega \Psi(\mathbf{F}(\vec{X}; \boldsymbol{x})) d\vec{X} \quad (3)$$

In the definitions above, $\boldsymbol{x} = (\vec{x}_1, ..., \vec{x}_N)$ is a vector containing all nodal degrees of freedom and conveys the state of our discrete model. The symbol $\mathcal{N}_i(\vec{X})$ denotes the interpolation basis functions associated with each node $\vec{X}_i$. In our approach those will be trilinear interpolating basis functions, associated with the vertices of a cubic lattice as detailed in section 4. For our subsequent discussion we will not restrict ourselves to a particular material model. Instead, we will treat $\Psi(\mathbf{F})$ as a placeholder for any material-specific strain energy definition. Specific energy formulas for common material models are provided in the supplemental technical document. Note that both the deformation gradient $\mathbf{F}(\vec{X}; \boldsymbol{x})$ as well as the energy density $\Psi(\mathbf{F}(\vec{X}; \boldsymbol{x}))$ are spatially varying functions (of the material location $\vec{X}$). This should be contrasted with tetrahedral discretizations where such quantities are constant on each element, as a consequence of the linear basis functions used in that setting. In any case, once a discrete energy $E(\boldsymbol{x})$ has been defined, the discrete nodal forces are readily computed as $\vec{f}_i = -\partial E/\partial\vec{x}_i$.

The remainder of this section addresses certain adjustments to the discrete energy definition (3) including modifications to performed approximations and a reformulation of the discrete state variables. Our objective is to support a spectrum of materials from compressible to highly-incompressible, accommodate true nonlinear volume preservation constraints and avoid locking or poor numerical conditioning problems that often stem from incompressible materials.

2.1 Quasi-incompressibility

We model response to volume change using the formulation referred to as *quasi-incompressibility*. In this approach, instead of enforcing incompressibility as a hard constraint we append a penalty-like volume preservation term to the definition of the deformation energy, with a tunable stiffness that allows a range of compressible to highly incompressible behaviors. The energy density function has the general form $\Psi(\mathbf{F}) = \Psi_0(\mathbf{F}) + \kappa M^2(\mathbf{F})/2$, where $M(\mathbf{F})$ measures the deviation from a volume-preserving configuration and κ is the stiffness of the incompressibility constraint which is related (or identified) with material properties such as the bulk modulus or the second Lamé coefficient (λ). For example, linear elasticity defines $M(\mathbf{F}) = \mathrm{tr}(\mathbf{F} - \mathbf{I})$ which seeks to make the displacement field divergence free. Corotated elasticity uses $M(\mathbf{F}) = \mathrm{tr}(\mathbf{\Sigma} - \mathbf{I})$ (where $\mathbf{F} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ is the SVD of $\mathbf{F}$) essentially enforcing that the average of principal stretch ratios is equal to one. Both measures provide an adequate approximation of volume change in the small strain regime, but become very inaccurate under large deformation (see Figure 2) . Thus, certain models (e.g. Neohookean elasticity) use the true volume change ratio $J = \det(\mathbf{F}) = \det(\mathbf{\Sigma})$ and define $M(\mathbf{F}) = \log(J)$ or $M(\mathbf{F}) = J - 1$, properly enforcing that the *product* of principal stretch ratios remains close to one. Although we recommend the use of the latter model types, we seek to accommodate any definition of $M(\mathbf{F})$ as even the less accurate formulations may be quite acceptable in appropriate deformation scenarios.

For discretization we superimpose a Cartesian lattice on the reference model shape Ω, naturally defining a partitioning $\Omega = \cup\Omega_k$ of the elastic domain into sub-domains $\Omega_k = \Omega \cap C_k$ within each lattice cell C_k. No restriction on the shape of Ω is imposed. Thus, each sub-domain Ω_k is either an entire cell of our cubic lattice (for cells fully interior to the deforming model) or a *fractional* cell when C_k overlaps with the model boundary. The discrete energy can also be split into a sum of local terms $E(\boldsymbol{x}) = \sum_k E_k(\boldsymbol{x})$, integrated over the respective Ω_k. We define the energy of each cell as follows:

$$E_k := \int_{\Omega_k} \Psi_0(\mathbf{F}) d\vec{X} + \tfrac{1}{2}\kappa W_k \overline{M}^2(\Omega_k) \quad (4)$$

$$\text{where } W_k := \int_{\Omega_k} d\vec{X} \text{ and } \overline{M}(\Omega_k) := \tfrac{1}{W_k} \int_{\Omega_k} M(\mathbf{F}) d\vec{X} \quad (5)$$

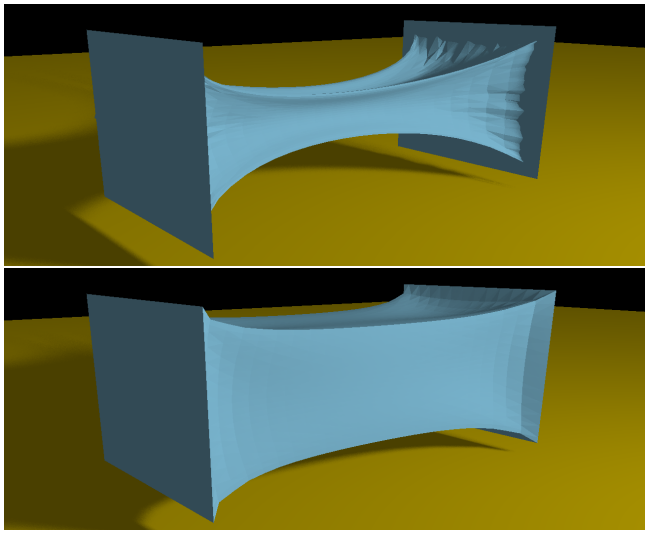

Figure 2: *Simulation of corotated (top) and neohookean (bottom) materials at high Poisson's ratio ($\nu = .498$). The corotated model loses more than 50% of the original volume due to its inaccurate incompressibility term. The neohookean model stays within $.1\%$ of its original volume, with less than 1% volume variation per element.*

Note that the volume W_k of Ω_k is simply h^3 if Ω_k is a fully interior cell. The liberties we took in this formulation should be apparent if we examine the second term of equation (4): instead of integrating the squared term $M^2(\mathbf{F})$ over Ω_k, we first compute an average $\overline{M}$ of the constraint measure $M(\mathbf{F})$ and proceed to integrate the square of this average. We do so in order to mitigate locking; a continuous material may be expected to be incompressible at any scale, however, our discrete deformation field $\phi(\vec{X};\boldsymbol{x})$ is only a product of interpolation. Thus, even if the total volume of Ω_k is exactly preserved, it is possible that interior locations may exhibit small local expansions or contractions that cancel each other out. Integrating the square of the measure $M(\mathbf{F})$ and using a high stiffness κ for this volume penalty would unnecessarily force continuous volume conservation at any interior point of Ω_k, rather than asking that Ω_k remains incompressible on the aggregate. The consequence is a parasitic stiffening of the material (locking) even on deformation modes that preserve the total volume. One can verify that the approximations reflected in equation (4) are sound; under refinement both the energy and resulting forces converge (to second order) to the respective continuous quantities . Finally we clarify using equation (4) does not automatically eliminate locking. Although this works well with our trilinear lattice elements, certain other discretizations (e.g. linear tetrahedra) would necessitate further modifications.

Accurate volume change penalization For those material models that use the local volume change ratio $J = \det \mathbf{F}$ in $M(\mathbf{F})$ we propose an optional modification which can penalize volume change even more accurately. For these materials we can write $M(J)$ for the volume change measure, and we modify our definition for the average measure $\overline{M}(\Omega_k)$ as

$$\overline{M}(\Omega_k) := M(\overline{J}) = M(\tfrac{1}{W_k}\textstyle\int_{\Omega_k} J d\vec{X}) \qquad (6)$$

Equation (6) formulates the incompressibility energy penalty based on the aggregate volume change, while the original formulation of equation (5) averages out the penalty itself over Ω_k. We note that this optional modification is not a prerequisite for locking resilience. In the following sections, we detail how either definition for $\overline{M}(\Omega_k)$ can be easily incorporated into our discrete formulation.

2.2 Mixed formulation

Although the actions of section 2.1 prevent locking, numerical simulation of near-incompressible materials is still hindered by poor conditioning of the governing equations. In the incompressible limit ($\kappa \to \infty$) the energy of equation (4) is dominated by the $M^2(\mathbf{F})$ term. Thus, there is great discrepancy in the stiffness associated with the incompressibility term, compared to the nominal elastic stiffness that volume-preserving modes are subject to. As an unfortunate consequence, the convergence of iterative algorithms (e.g. Conjugate Gradients) is significantly decelerated. We treat this problem by adopting a mixed position/pressure discretization, avoiding problematic conditioning even at the incompressible limit.

We introduce a new state variable, in addition to the deformation $\vec{x}=\phi(\vec{X})$: we refer to this as a (pseudo-)pressure $p(\vec{X})$. Discretely, we represent the pressure field as piecewise constant on each cell, associating a single value p_k with each Ω_k. We now consider the alternative energy expression $\hat{E}(\boldsymbol{x},\boldsymbol{p}) = \sum_k \hat{E}_k(\boldsymbol{x},p_k)$. Here we denote the set of all cell pressures with $\boldsymbol{p} = (p_1,\ldots,p_M)$ and:

$$\begin{aligned}\hat{E}_k(\boldsymbol{x},p_k) &:= \int_{\Omega_k}\left[\Psi_0(\mathbf{F}) + \alpha p M(\mathbf{F}) - \frac{\alpha^2 p^2}{2\kappa}\right] d\vec{X} \\ &= \int_{\Omega_k}\Psi_0(\mathbf{F})d\vec{X} + W_k\left[\alpha p_k \overline{M}(\Omega_k) - \frac{\alpha^2 p_k^2}{2\kappa}\right] \qquad (7)\end{aligned}$$

where α is an arbitrary constant. This modified energy expression has the following important properties: First, while $E(\boldsymbol{x})$ is expected to be generally convex (the deformation energy has a global minimum, although local maxima or saddle points may be present) the modified energy $\hat{E}(\boldsymbol{x},\boldsymbol{p})$ is concave with respect to $\boldsymbol{p}$, due to the negative sign of the term $-\alpha^2p^2/(2\kappa)$. Our second observation is that $E(\boldsymbol{x})$ and $\hat{E}(\boldsymbol{x},\boldsymbol{p})$ have the same critical points: by differentiating equations (4) and (7) we can show that when $\partial\hat{E}/\partial\boldsymbol{p} = 0$ is satisfied, then $\partial E/\partial\boldsymbol{x} = \partial\hat{E}/\partial\boldsymbol{x}$. Thus, if $\partial\hat{E}/\partial\boldsymbol{p} = 0$, the discrete forces computed by either energy are identical. This relation has an intuitive consequence in the context of a quasistatic (steady-state) simulation: find a configuration $\boldsymbol{x}$ such that the elastic forces are at equilibrium $\boldsymbol{f}(\boldsymbol{x}) = \mathbf{0}$, or equivalently, the energy $E(\boldsymbol{x})$ is minimized. In this setting, if one finds a *saddle point* $(\boldsymbol{x}_*,\boldsymbol{p}_*)$ for the modified energy $\hat{E}(\boldsymbol{x},\boldsymbol{p})$, then $\boldsymbol{x}_*$ is a critical point (generally a minimum) for $E(\boldsymbol{x})$ and thus a static equilibrium configuration.

These observations make it possible to compute discrete forces as $\hat{f}_i = \partial\hat{E}(\boldsymbol{x},\boldsymbol{p})/\partial\vec{x}_i$ and simply append $\partial\hat{E}/\partial\boldsymbol{p} = 0$ to the governing equations arising from this force definition; the physical response of the material will be identical to that computed from the original formulation of section 2.1. The practical benefit of this formulation is that it is, numerically, very resilient to high incompressibility; where setting $\kappa\to\infty$ in the original formulation would yield a very stiff energy, the expression of equation (7) remains well conditioned even in the limit case (where the quadratic pressure term would simply vanish in a smooth fashion). Lastly, we adopt the notation $\boldsymbol{q} = -\partial\hat{E}/\partial\boldsymbol{p}$ which we call a "pressure force" in analogy to $\hat{\boldsymbol{f}} = -\partial\hat{E}/\partial\vec{x}_i$, i.e. the "positional" force due to $\hat{E}$. We also write:

$$\hat{\Psi}(\mathbf{F},p) = \Psi_0(\mathbf{F}) + \alpha p M(\mathbf{F}) - \alpha^2p^2/(2\kappa) \qquad (8)$$

as an alternative *augmented energy* definition including p as a state variable. Equation (7) follows directly by substituting the energy definition (8) into equation (3), without any further approximation. Finally, we note that the specific value of α has no impact on the computed solution, provided that discrete equations are solved to reasonable precision. But, if solvers with low accuracy are used (e.g. CG with few iterations), large α values will prioritize volume conservation, while small values promote smooth approximations.

Relation with existing approaches Our formulation draws inspiration from the rich applied mathematics literature on mixed finite element methods [Brezzi and Fortin 1991; Arnold 1990] which includes many references to near-incompressible elasticity as a target domain. The most relevant graphics work [Zhu et al. 2010] also employs a pressure-based mixed formulation, yet is explicitly limited to linear and corotated materials. Furthermore, their formulation is based on finite-differences and voxel-accurate only. Our formulation is given at the energy level, spans arbitrary materials and can leverage finite element techniques to yield symmetric, sub-voxel accurate discretizations. In contrast with solutions on tetrahedral meshes [Irving et al. 2007] we handle nonlinear volume constraints and need not presume any implicit or semi-implicit time integration scheme. Deformation constraints have also been tackled via constrained dynamics [Goldenthal et al. 2007] and nonconforming discretizations [English and Bridson 2008]. We move away from incompressibility as a hard constraint, and we treat materials with any degree of incompressibility in a uniform fashion.

3 Quadrature and boundary treatment

Equations (4) and (7) are already discretized (i.e. they are fully defined by the discrete state variables) but not in a form that facilitates direct implementation, due to the presence of continuous integrals in their formulas. This is trivial for tetrahedral discretizations, as the integrands are constant on each element. Although one might similarly envision computing such integrals analytically for our trilinear elements, this is not a practical option for nonlinear materials since the integrand is a complex multivariate expression. Even for linear elasticity, analytic integration would be cumbersome for boundary cells, where Ω_k has an irregular shape, and is not a perfect cube. Thus, we use properly structured numerical quadrature rules, for practical evaluation of these integrals. An m-point quadrature rule would take the form $\int_{\Omega_k} f d\vec{X} \approx W_k \sum_{i=1}^{m} w_i f(\vec{X}_i)$, for appropriately chosen weights w_i (adding up to one) and quadrature points $\{\vec{X}_i\}_{i=1}^{m} \subset \Omega_k$. Using this rule, we can approximate:

$$\int_{\Omega_k} \Psi_0(\mathbf{F}) d\vec{X} \approx W_k \sum w_i \Psi_0(\mathbf{F}(\vec{X}_i))$$
$$\overline{M}(\Omega_k) \approx \sum w_i M(\mathbf{F}(\vec{X}_i))$$

From the theory of finite element methods [Hughes 1987] it is known that trilinear interpolating functions are capable, in principle, of discretizations that converge to the continuous solution with second-order accuracy, i.e. with an error that diminishes like $O(h^2)$ on a grid with spacing h. In this section we present discretization approaches based on (a) a second-order quadrature method which is expected to retain $O(h^2)$ solution accuracy and (b) a first-order quadrature scheme combined with a stabilization technique. Using the latter approach limits the observed accuracy of our scheme to $O(h)$, i.e. first order; however this less accurate (yet inexpensive) quadrature is ultimately leveraged in section 4 only for the purpose of accelerating the numerical solution of the second-order scheme.

3.1 Second order method

We propose a novel numerical quadrature scheme which achieves second-order accuracy (i.e. it integrates exactly polynomials of degree up to 2) on arbitrarily integration domains. For reference, we highlight one of the options that is broadly used for conforming hexahedral meshes: the 8-point Gauss quadrature scheme achieves second-order accurate integration on (skew) hexahedra, and is actually third-order accurate on axis-aligned rectangular parallelepipeds. Unfortunately, this approach is not easily adapted to integration domains that are not hexahedral, such as the fractional

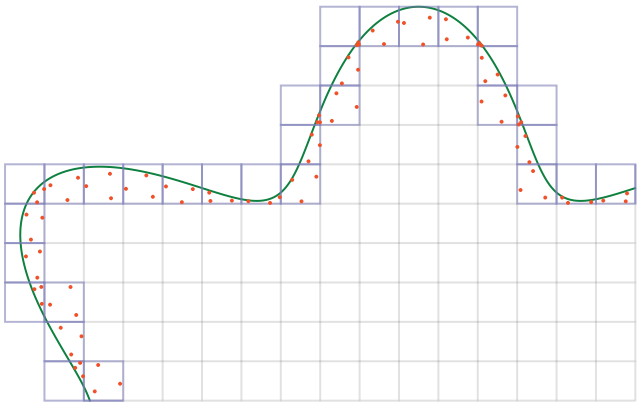

Figure 3: *A 2D illustration of our boundary quadrature. In each boundary cell 3 quadrature points (4 in 3D) are selected in a way that they match the moments of the respective material fragment.*

cells Ω_k covering the boundary of the deforming model. In contrast, our method achieves second order with just four quadrature points, for any shape of the domain of integration.

Our quadrature rule will be second order accurate if and only if it integrates exactly all monomials $X^p Y^q Z^r$ with $0 \le p+q+r \le 2$. Thus, we must have $\int_{\Omega_k} X^p Y^q Z^r d\vec{X} = W_k \sum_{i=1}^{4} w_i X_i^p Y_i^q Z_i^r$ for all $0 \le p+q+r \le 2$. This is equivalent to the matrix equation:

$$\int_{\Omega_k} \begin{pmatrix} 1 & X & Y & Z \\ X & X^2 & XY & XZ \\ Y & XY & Y^2 & YZ \\ Z & XZ & YZ & Z^2 \end{pmatrix} d\vec{X} = W_k \sum_{i=1}^{4} w_i \begin{pmatrix} 1 & X_i & Y_i & Z_i \\ X_i & X_i^2 & X_iY_i & X_iZ_i \\ Y_i & X_iY_i & Y_i^2 & Y_iZ_i \\ Z_i & X_iZ_i & Y_iZ_i & Z_i^2 \end{pmatrix} \quad (9)$$

In other words the 4 points $\vec{X}_1, \ldots, \vec{X}_4$, when viewed as a discrete distribution, must match all first- and second-order moments of the continuous distribution of points $\vec{X} \in \Omega_k$. The top leftmost entry of this matrix equation further ensures that the quadrature weights will be a partition of unity. Equation (9) illustrates a procedure for generating the four quadrature points; the matrix on the left-hand side (which we symbolize as $\mathbf{C}$) is symmetric and positive definite matrix which can be precomputed as a one-time preprocessing step. In our implementation we use Monte-Carlo integration to compute the relevant moments of fractional cells Ω_k. Then, equation (9) can be written as $W_k^{-1}\mathbf{C} = \mathbf{M}\mathbf{M}^T$, where

$$\mathbf{M} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ X_1 & X_2 & X_3 & X_4 \\ Y_1 & Y_2 & Y_3 & Y_4 \\ Z_1 & Z_2 & Z_3 & Z_4 \end{pmatrix} \begin{pmatrix} \pm\sqrt{w_1} & & & \\ & \pm\sqrt{w_2} & & \\ & & \pm\sqrt{w_3} & \\ & & & \pm\sqrt{w_4} \end{pmatrix} \quad (10)$$

In fact, given *any* symmetric factorization $W_k^{-1}\mathbf{C} = \mathbf{M}\mathbf{M}^T$, we can always bring the matrix $\mathbf{M}$ into the form of equation (10), by simply pulling the first row of M into the diagonal of the right factor in (10), and scaling the columns accordingly. Then, the weights can be obtained by squaring the diagonal entries $\pm\sqrt{w_i}$, and are guaranteed to sum to one by virtue of equation (9). At the same time, the coordinates of the quadrature points can be read from the the left matrix factor of equation (10).

It thus appears that the quadrature points can be computed once any symmetric factorization of $W_k^{-1}\mathbf{C}$ becomes available; this can be obtained via the Cholesky method, or the SVD among other options. However, we need to address an important issue: it is not guaranteed that, for a given factorization, the points computed from (10) will be interior to Ω_k; in fact, they could be located arbitrarily far away from it. Nevertheless, if four quadrature points that satisfy

equation (9) and are interior to Ω_k do exist, they will be associated with a different factorization $W_k^{-1}\mathbf{C} = \mathbf{N}\mathbf{N}^T$. We can show that if two 4×4 matrices $\mathbf{M}$ and $\mathbf{N}$ satisfy $\mathbf{M}\mathbf{M}^T = \mathbf{N}\mathbf{N}^T$, then $\mathbf{M} = \mathbf{N}\mathbf{Q}$, where $\mathbf{Q}$ is an orthogonal matrix. Thus, we begin with *any* factorization $W_k^{-1}\mathbf{C} = \mathbf{M}\mathbf{M}^T$ (e.g. Cholesky), and proceed to sample the space of 4×4 orthogonal matrices $\mathbf{Q}$ from which different matrices $\mathbf{N}$ can be obtained, providing different choices of candidate quadrature points. Although we do not provide a theoretical proof, our practical experience indicated that it is always possible to find four quadrature points that satisfy the stated constraints for any *convex* Ω_k, such that the four points are *interior* to Ω_k (in fact, we found that it is possible to also find equally-weighted points, in this case). Even in the case where Ω_k is not convex, our experience shows that we can easily find four quadrature points that are, at least, interior to the lattice cell containing Ω_k (see Figure 3).

We use the quadrature rule constructed in this fashion to evaluate all integrals introduced in the previous section. For interior cells Ω_k which are perfect cubes, the selection of points can be made just once, and reused for all such entire cells; a possible choice of quadrature points that have equal weights ($w_i = 1/4$) are given below, with the domain of integration being the cube $[-1, 1]^3$:

$$(\sqrt{\frac{2}{3}}, \sqrt{\frac{1}{3}}, 0), (-\sqrt{\frac{2}{3}}, \sqrt{\frac{1}{3}}, 0), (0, -\sqrt{\frac{1}{3}}, \sqrt{\frac{2}{3}}), (0, -\sqrt{\frac{1}{3}}, -\sqrt{\frac{2}{3}})$$

Should the formulation of equation (6) be used, one may opt to use 8-point Gauss quadrature to compute $\overline{J}$ as the integrand is composed of monomials $X^pY^qZ^r$ with $\max\{p, q, r\} \leq 2$, which will be integrated exactly with the Gauss method. In our implementation we used our 4-point quadrature even for this case, as we observed a discrepancy of less than 0.1% between the $\overline{J}$ computed by either rule, even for cases of severe deformation. Finally, we note that in cases where voxels in our embedding lattices had less than a 2% coverage in material, we chose to omit these voxels altogether (as is visible in Figure 1), and associate surface points embedded in them with neighboring voxels (which had greater material coverage) using trilinear weights extending slightly beyond the $[0, 1]$ range. This had practically no effect on the simulated deformation, yet helped improve the numerical conditioning of our discrete systems.

3.2 First order method

We also implemented a first-order accurate method using a one-point quadrature rule placed at the center of the cell containing each Ω_k, in a fashion similar to [McAdams et al. 2011]. In addition, all cells intersected by the deforming body are treated as if they were fully covered with material, essentially modeling an approximation to Ω which has been quantized at voxel precision.Although not as accurate as the second-order treatment previously described, this approach yields a simpler, less computationally expensive implementation. This is attributed to both the regularity of computation (same quadrature scheme for all cells), as well as the fact that fractional cells are treated as whole, preventing conditioning issues.In section 4 we illustrate how we use this first-order discretization as a building block for a numerical solver of the second-order accurate scheme, achieving both performance and accuracy.

As previously observed [McAdams et al. 2011] a single-point quadrature is normally unstable, due to the fact that certain oscillatory deformation modes are invisible to the discrete energy thus formulated; notably this is not an issue with our second-order quadrature, since any stress-inducing deformation mode that is invisible at the location of a certain quadrature point (due to cancellation) will still be visible and penalized at one or more of the remaining quadrature points. McAdams et al [2011] suggested a stabilization scheme, based on the separation of the energy density for corotational elasticity into a term that corresponds to a Laplace operator,

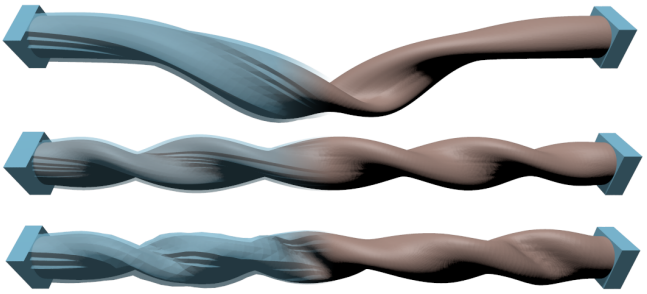

Figure 4: *A thin-walled elastic tube is compressed and twisted. Top: An 8×8×80 voxel accurate simulation under-resolves the thinness of the walls and fails to allow intricate corrugations. Middle: A voxel-accurate simulation at 16×16×160 resolution recovers deformation detail. Bottom: Our second-order method captures detailed deformation behaviors even at coarse lattice resolutions.*

plus a residual term accounting for the remainder of the corotational response. We also employ a stabilization step, which can be considered an extension of their approach, but state it in a way that allows its use with any material energy density function. Specifically, we add to the energy density $\hat{E}_k$ a stabilization term E^k_{stab} defined as

$$E^k_{\text{stab}}(\boldsymbol{x}) = \mu \int_{C_k} \left[\|\mathbf{F}(\vec{X}; \boldsymbol{x})\|_F^2 - \|\mathbf{F}(\vec{X}_c; \boldsymbol{x})\|_F^2 \right] d\vec{X}$$

where $\vec{X}_c$ is the location of the cell center and μ is the first Lamé parameter, which is either inherently present in many material models or easily approximated from their specific parameters. We compute the integral exactly over the entire cell C_k containing Ω_k; in fact, the nature of the integrand is such that 8-point Gauss quadrature performs an exact integration. The final result of the integration can be shown to be a quadratic, convex function of the nodal degrees of freedom, and thus expressed by $E_{\text{stab}}(\boldsymbol{x}) = \mu h^3 \boldsymbol{x}^T \mathbf{K}^k_{\text{stab}} \boldsymbol{x}$, where the constant matrix $\mathbf{K}_{\text{stab}}$ is block diagonal (across the X, Y, Z coordinates), symmetric and positive semi-definite. We proceeded to perform a Taylor analysis of this stabilization term, revealing that

$$E^k_{\text{stab}}(\boldsymbol{x}) \approx \left(\left\| \frac{\partial^2 \vec{\phi}(\vec{X}_c)}{\partial X \partial Y} \right\|^2 + \left\| \frac{\partial^2 \vec{\phi}(\vec{X}_c)}{\partial X \partial Z} \right\|^2 + \left\| \frac{\partial^2 \vec{\phi}(\vec{X}_c)}{\partial Y \partial Z} \right\|^2 \right) O(h^5)$$

Thus, this term penalizes the large mixed partial derivatives of $\vec{\phi}$ which are associated with oscillatory deformations that are invisible to the unmodified one-point quadrature scheme. Since this term scales proportionately to $O(h^5)$, it is equivalent to an $O(h^2)$ perturbation of the energy, and vanishes rapidly under refinement.

Relation to prior work There is significant prior graphics research on the topic of resolving simulation properties of embedded models with sub-element precision near boundaries [Nesme et al. 2006; Nesme et al. 2009; Kim and Pollard 2011] while others have investigated absorbing boundary detail and anisotropy into the constitutive model for boundary elements [Kharevych et al. 2009]. Sub-voxel accurate discretizations have been popular for fluids animation [Guendelman et al. 2005; Batty et al. 2007; Wojtan and Turk 2008], often in the context of solid-fluid coupling. Our use of a non-conforming lattice-derived basis also parallels the work of Chuang et al [2009] who discretize a surface PDE using a spline FEM basis over a non-conforming 3D lattice. There is also extensive computational physics literature on second or higher order accurate techniques for finite difference, finite element or XFEM discretizations on Cartesian grids [Fedkiw et al. 1999; Almgren et al. 1997; Daux et al. 2000]. Our original contribution is the new

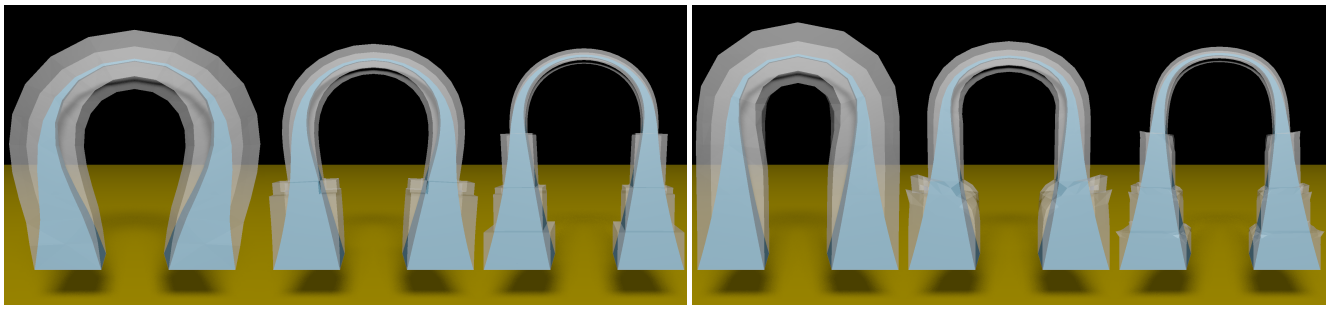

Figure 5: *Left: A 3D bow-tie model is bent, and simulated with the first order one-point quadrature scheme. Significant refinement is needed for convergence to the continuous behavior. Right: Using our second-order quadrature scheme, the correct asymptotic behavior is reached even on coarse resolutions. [Note: This example uses trilinear (not tricubic) interpolation to reconstruct the embedded object geometry.]*

numerical quadrature scheme which can compute the boundary integrals to second order accuracy on arbitrary domains Ω_k, while only requiring four quadrature points (in 3D). To contrast our novel approach with established variants, we note that XFEM methods routinely define quadrature rules by re-tessellating the boundary region. Other methods (see e.g. [Hellrung et al. 2012]) leverage the divergence theorem to evaluate such integrals exactly on polyhedral approximations of Ω_k, yet such approaches are significantly more cumbersome and impractical for nonlinear materials. Finally, our stabilization treatment for the first-order approach clearly draws inspiration from [McAdams et al. 2011], and our treatment is very similar (although not identical) for corotational materials. However, our method extends naturally to arbitrary nonlinear models.

4 Linearization and numerical solution

We now focus on the numerical solution of the discretized governing equations. For this exposition we focus on *quasistatic* simulation of elasticity, i.e. we generate a sequence of steady-state deformations that result from the kinematic positional constraints imposed at every frame of animation. In a conventional setting, the equation defining the quasistatic solution at each frame is simply $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$, stating that all nodal elastic forces needs to be zero at equilibrium. However, in light of the mixed formulation introduced in section 2, the equilibrium equation is replaced by the system

$$\boldsymbol{f}(\boldsymbol{x},\boldsymbol{p}) = \boldsymbol{0} \;\text{ and }\; \boldsymbol{q}(\boldsymbol{x},\boldsymbol{p}) = \boldsymbol{0}. \tag{11}$$

We use a Newton-Raphson method to generate an iterative solution method for this nonlinear system of state variables $\boldsymbol{x}$ and $\boldsymbol{p}$. After k steps of this iteration, we linearize $\boldsymbol{f}$ and $\boldsymbol{q}$ around the current approximation of the solution $(\boldsymbol{x}_k, \boldsymbol{p}_k)$ to obtain:

$$\begin{bmatrix} \boldsymbol{f}(\boldsymbol{x}_k+\delta\boldsymbol{x}, \boldsymbol{p}_k+\delta\boldsymbol{p}) \\ \boldsymbol{q}(\boldsymbol{x}_k+\delta\boldsymbol{x}, \boldsymbol{p}_k+\delta\boldsymbol{p}) \end{bmatrix} \approx \begin{bmatrix} \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{p}_k) \\ \boldsymbol{q}(\boldsymbol{x}_k, \boldsymbol{p}_k) \end{bmatrix} + \begin{bmatrix} \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} & \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}} \\ \frac{\partial \boldsymbol{q}}{\partial \boldsymbol{x}} & \frac{\partial \boldsymbol{q}}{\partial \boldsymbol{p}} \end{bmatrix} \begin{bmatrix} \delta\boldsymbol{x} \\ \delta\boldsymbol{p} \end{bmatrix}$$

where the partial derivatives are computed at $(\boldsymbol{x}_k, \boldsymbol{p}_k)$. Requesting that $\boldsymbol{f}$ and $\boldsymbol{q}$ approximate zero after corrections $\delta\boldsymbol{x}, \delta\boldsymbol{p}$ have been applied yields the Newton-Raphson update equation:

$$\underbrace{-\begin{bmatrix} \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}(\boldsymbol{x}_k, \boldsymbol{p}_k) & \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}}(\boldsymbol{x}_k, \boldsymbol{p}_k) \\ \frac{\partial \boldsymbol{q}}{\partial \boldsymbol{x}}(\boldsymbol{x}_k, \boldsymbol{p}_k) & \frac{\partial \boldsymbol{q}}{\partial \boldsymbol{p}}(\boldsymbol{x}_k, \boldsymbol{p}_k) \end{bmatrix}}_{\boldsymbol{K}(\boldsymbol{u}_k)} \underbrace{\begin{bmatrix} \delta\boldsymbol{x} \\ \delta\boldsymbol{p} \end{bmatrix}}_{\delta\boldsymbol{u}} = \underbrace{\begin{bmatrix} \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{p}_k) \\ \boldsymbol{q}(\boldsymbol{x}_k, \boldsymbol{p}_k) \end{bmatrix}}_{\boldsymbol{g}(\boldsymbol{u}_k)} \tag{12}$$

where we denote the combined state vector by $\boldsymbol{u} = (\boldsymbol{x}, \boldsymbol{p})$ and the combined force vector by $\boldsymbol{g} = (\boldsymbol{f}, \boldsymbol{q})$. Using the definitions of $\boldsymbol{f}$ and $\boldsymbol{q}$ as the negative gradients of $E(\boldsymbol{x}, \boldsymbol{p})$ with respect to position and pressure, respectively, the stiffness matrix $\boldsymbol{K}(\boldsymbol{u}_k)$ can be written:

$$\boldsymbol{K}(\boldsymbol{u}_k) = \begin{bmatrix} (\partial^2 E/\partial \boldsymbol{x}^2)(\boldsymbol{x}_k, \boldsymbol{p}_k) & (\partial^2 E/\partial \boldsymbol{x}\partial \boldsymbol{p})(\boldsymbol{x}_k, \boldsymbol{p}_k) \\ (\partial^2 E/\partial \boldsymbol{x}\partial \boldsymbol{p})(\boldsymbol{x}_k, \boldsymbol{p}_k) & (\partial^2 E/\partial \boldsymbol{p}^2)(\boldsymbol{x}_k, \boldsymbol{p}_k) \end{bmatrix}$$

This indicates that $\boldsymbol{K}(\boldsymbol{u}_k)$ is a symmetric matrix. Taking into consideration equation (7) we also infer that $\boldsymbol{K}(\boldsymbol{u}_k)$ is generally expected to be *indefinite*. Thus, we employ the Symmetric Quasi-Minimal Residual (QMR) method [Freund and Nachtigal 1994], a Krylov-subspace solver for symmetric indefinite systems. At the end of each Newton-Raphson iteration we incorporate the computed correction to obtain $\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k + \delta\boldsymbol{x}$ and $\boldsymbol{p}_{k+1} \leftarrow \boldsymbol{p}_k + \delta\boldsymbol{p}$.

High-order defect correction In section 3 we detailed two quadrature methods for computing boundary energy integrals. Depending on the specific choice of method there will be different formulas for the combined force $\boldsymbol{g}$ and combined stiffness $\boldsymbol{K}$. Let us denote by $\boldsymbol{g}_1, \boldsymbol{K}_1$ the force and stiffness definitions resulting from the first-order quadrature of section 3.2 and let $\boldsymbol{g}_2, \boldsymbol{K}_2$ be their second-order counterparts according to section 3.1. When using the QMR method to solve equation (12) we only need to evaluate $\boldsymbol{g}$ once, in order to generate the right hand side; in contrast, the stiffness matrix $\boldsymbol{K}$ is used once per QMR iteration, thus dominating the computational cost of the solution process. Solving the equilibrium problem to sub-voxel precision would normally require solving the version $\boldsymbol{K}_2(\boldsymbol{u}_k)\delta\boldsymbol{u} = \boldsymbol{g}_2(\boldsymbol{u}_k)$. Instead, we employ a high-order defect correction procedure [Trottenberg et al. 2001] by solving a modified Newton system with $\boldsymbol{K}_1$ on the left-hand side:

$$\boldsymbol{K}_1(\boldsymbol{u}_k)\delta\boldsymbol{u} = \boldsymbol{g}_2(\boldsymbol{u}_k) \tag{13}$$

At first, such a substitution may appear unjustified. However, it can be shown that if the spectral radius $\rho\left(\mathbf{I} - \boldsymbol{K}_1^{-1}\boldsymbol{K}_2\right)$ is less than one, then repeated applications of the defect correction iteration (13) will converge to the solution of the higher-order nonlinear system $\boldsymbol{g}_2(\boldsymbol{u}) = \boldsymbol{0}$. When dealing with discretizations of elliptic problems (such as our mixed formulation of elasticity) this spectral radius criterion is expected to hold true (it can be shown to do so in simple cases, e.g. when $\boldsymbol{K}$ is the Laplace or linear elasticity operator). The price one has to pay for the convenience of using a lower-order operator in a defect correction procedure is that the speed of convergence for Newton-Raphson typically degrades from quadratic to linear; this is very well tolerated in graphics applications, especially since the typical practice of terminating Krylov solvers after a fixed maximum number of iterations would typically result in the same speed compromise in the first place. In our experiments, we encountered no issues with using defect correction in the Newton-Raphson process, other than a very modest increase (less than 50%) in the number of Newton iterations required.

5 Implementation

Trilinear elements and gradients We start by deriving a concise expression for the deformation gradient $\mathbf{F}(\vec{X}; \boldsymbol{x})$. We use the notation $\{\vec{X}_{i_1 i_2 i_3}\}_{i_1,i_2,i_3 \in \{0,1\}}$ for the eight vertices of a given cell

with $\vec{X}_{i_1 i_2 i_3}=\vec{X}_0+(i_1,i_2,i_3)h$. Their respective interpolation basis functions are $\mathcal{N}_{i_1 i_2 i_3}(\vec{X})=\prod_k (\xi_k)^{i_k}(1-\xi_k)^{1-i_k}$ where $\vec{\xi}(\vec{X})=(\xi_1,\xi_2,\xi_3)=(X-X_0,Y-Y_0,Z-Z_0)/h$ are the trilinear coordinates of $\vec{X}$. Partial derivatives of the interpolating functions are readily computed as $\partial_j \mathcal{N}_{i_1 i_2 i_3}(\vec{X})=(-1)^{1-i_j}\prod_{k\neq j}(\xi_k)^{i_k}(1-\xi_k)^{1-i_k}$. From this point, we shall refer to the vertices of a specific cell simply as $\vec{X}_1 ... \vec{X}_8$, and $\mathcal{N}_1(\vec{X}) ... \mathcal{N}_8(\vec{X})$ for the respective trilinear basis functions, with the understanding that we know how to relate to the prior indexing convention. By equations (1,2) the deformation gradient at a location $\vec{X}_*$ is $\mathbf{F}(\vec{X}_*;\boldsymbol{x})=\sum_i \vec{x}_i \nabla\mathcal{N}_i(\vec{X}_*)^T = \mathbf{B}(\boldsymbol{x})\mathbf{G}(\vec{X}_*)^T$ where $\mathbf{B}(\boldsymbol{x}) = [\vec{x}_1\ \vec{x}_2\ \dots\ \vec{x}_8] \in \mathbf{R}^{3\times 8}$ is the cell shape matrix. The matrix $\mathbf{G}(\vec{X}_*)\in\mathbf{R}^{3\times 8}$ with $G_{ij}(\vec{X}_*) = \partial_i \mathcal{N}_j(\vec{X}_*)$ will be referred to as the gradient matrix at $\vec{X}_*$. Note that for any material point $\vec{X}_*$, $\mathbf{G}(\vec{X}_*)$ can be precomputed

Force computation We mimic the elastic force derivation from McAdams et al [2011] and write $\mathbf{H}_k(\boldsymbol{x},p_k)=[\vec{f}_1\ \vec{f}_2\ \dots\ \vec{f}_8]$ for the 3×8 matrix containing all nodal elastic forces in a cell Ω_k. If $\{\vec{X}_i\}_{i=1}^m$ are the m quadrature points used for integration, with associated weights $\{w_i\}_{i=1}^m$, the force matrix $\mathbf{H}_k$ is assembled as:

$$\mathbf{H}_k(\boldsymbol{x},p_k) = -W_k \textstyle\sum_i w_i \hat{\mathbf{P}}(\mathbf{F}(\vec{X}_i;\boldsymbol{x}),p_k)\mathbf{G}(\vec{X}_i)$$

$$\text{where } \hat{\mathbf{P}}(\mathbf{F},p) = \partial\hat{\Psi}(\mathbf{F},p)/\partial\mathbf{F} = \mathbf{P}_0(\mathbf{F}) + \alpha p \mathbf{Q}(\mathbf{F}).$$

In this last expression $\hat{\mathbf{P}}$ is the modified 1st Piola-Kirchhoff stress tensor, associated with our mixed formulation. $\mathbf{P}_0=\partial\Psi_0(\mathbf{F})/\partial\mathbf{F}$ is the stress component that excludes the incompressibility term, and $\mathbf{Q}=\partial M(\mathbf{F})/\partial\mathbf{F}$ is the gradient of the volume measure $M(\mathbf{F})$. Finally, the pressure force q_k is given by:

$$q_k(\boldsymbol{x},p_k) = -\tfrac{\partial \hat{E}_k}{\partial p_k} = \alpha W_k(\alpha p_k/\kappa - \textstyle\sum_i w_i M(\mathbf{F}(\vec{X}_i;\boldsymbol{x}))$$

Force differentials The QMR solver used to solve equation (12) does not require the matrix $\mathbf{K}(\boldsymbol{u}_k)$ to be explicitly constructed, as long as its action on an input vector $(\delta\boldsymbol{x},\delta\boldsymbol{p})$ can be evaluated. The result of this implicit matrix-vector multiplication are the force differentials $\delta\boldsymbol{f}$ and $\delta\boldsymbol{q}$ respectively. We perform this matrix-free operation on an element-by-element basis, adding the contribution of each Ω_k to the aggregate differentials. Once again the force differentials can be collectively computed as $\delta\mathbf{H}_k(\delta\boldsymbol{x},\delta p_k;\boldsymbol{x},p_k)=[\delta\vec{f}_1\ \delta\vec{f}_2\ \dots\ \delta\vec{f}_8]$, along with $\delta q_k(\delta\boldsymbol{x},\delta p_k;\boldsymbol{x},p_k)$ as follows:

$$\delta\mathbf{H}_k(\delta\boldsymbol{x},\delta p_k;\boldsymbol{x},p_k) = -W_k\delta\hat{\mathbf{P}}(\delta\mathbf{F},\delta p_k;\mathbf{F}(\vec{X}_c;\boldsymbol{x}),p_k)\mathbf{G}(\vec{X}_c)$$

$$\delta q_k(\delta\boldsymbol{x},\delta p_k;\boldsymbol{x},p_k) = \alpha W_k(\alpha\delta p_k/\kappa - \mathbf{Q}(\mathbf{F}(\vec{X}_c;\boldsymbol{x})) : \delta\mathbf{F})$$

$$\text{where } \delta\hat{\mathbf{P}}(\delta\mathbf{F},\delta p;\mathbf{F},p) = \mathcal{T}(\mathbf{F},p):\delta\mathbf{F} + \alpha\delta p\mathbf{Q}(\mathbf{F}),$$

$$\text{and } \delta\mathbf{F} = \mathbf{B}(\delta\boldsymbol{x})\mathbf{G}(\vec{X}_c)^T.$$

in this expression, the fourth order tensor $\mathcal{T}=\partial^2\hat{\Psi}/\partial\mathbf{F}^2$ is the stress derivative that we would obtain from equation (8) if p was simply treated as a constant value. We refer the reader to [Teran et al. 2005] for a discussion of how this tensor can be constructed via the SVD for isotropic materials. Also, note that due to our use of the defect correction iteration, only a single quadrature point (the cell center $\vec{X}_c$) was used in the equations above.

Definiteness fix A number of authors [Teran et al. 2005; McAdams et al. 2011] have emphasized the necessity to address the possible indefiniteness of the stiffness matrix $\partial\boldsymbol{f}/\partial\boldsymbol{x}$ for nonlinear materials. In our case, since we do not employ Conjugate Gradients as these methods do, the definiteness of the stiffness matrix is not a strict requirement. However, we observed that, in the absence of such modification, the Newton-Raphson procedure would sometimes converge to local minima, or even unstable force equilibrium configurations, which were seen as perfectly acceptable saddle points by our QMR algorithm. This issue was alleviated by performing a version of definiteness fix similar to the previously mentioned approaches. In particular, if we consider the original energy definition $E(\boldsymbol{x},\mu,\kappa)$ (prior to the introduction of pressures), the stiffness matrix is defined as $\mathbf{K}_{\mu,\kappa} = -\partial^2 E(\boldsymbol{x},\mu,\kappa)/\partial\boldsymbol{x}^2$, where we explicitly included a reference to two (representative) material parameters μ, and κ. Following the methodology of [Teran et al. 2005], we compute a modified stiffness matrix $\hat{\mathbf{K}}_{\mu,\kappa}$ by projecting $\mathbf{K}$ to its positive definite part. However, in order to avoid problematic conditioning, in case this positive definite correction has added a term proportional to the (very high in incompressible materials) parameter κ, we compute the following correction instead:

$$\mathbf{K}_{\text{corr}} = \hat{\mathbf{K}}_{\mu,\kappa^*} - \mathbf{K}_{\mu,\kappa^*}$$

where κ^* is a thresholded value of the incompressibility penalty, corresponding to a Poisson's ratio in the range of $\nu\approx 0.3-0.4$. Subsequently, we add the correction term $\mathbf{K}_{\text{corr}}$ to the top-left block of matrix $\mathbf{K}(\mathbf{u}_k)$ in equation (12). We observed excellent performance with this adjustment, which was fully adequate for all our academic and biomechanics examples. Similar to the defect correction procedure, this modification does not affect the computed solution, only changes the iterative procedure that converges to it.

Muscles and constraints Similar to the muscle model suggested in Lee et al [2009], we add an additional term Ψ_{muscle} to the deformation energy, in order to capture the contribution of contractile muscles. This term is defined as:

$$\Psi_{\text{muscle}}(\mathbf{F}) = \Psi_{\text{muscle}}(\lambda(\mathbf{F})),\ \text{ where } \lambda = \|\mathbf{F}\vec{w}\|_2$$

where $\vec{w}$ is the muscle fiber direction. Thus, λ measures the along-fiber elongation or compression. The corresponding Piola stress is:

$$\mathbf{P}(\mathbf{F}) = T(\lambda)\mathbf{F}\vec{w}\vec{w}^T$$

where $T(\lambda)$ is the muscle tension, specified by a Hill-type force-length relationship. As muscles are also sub-voxel features, the muscle-related component of the energy Ψ_{muscle} also requires a specific quadrature scheme. However, since this is not the only component of the energy, and stabilization is already handled by the term modeling the isotropic response of flesh, we only use a single quadrature point, placed at the centroid of the muscle volume within each cell. For the first-order approximation used in the defect correction procedure, we shift this quadrature point to the center of the cell, as with the other components of elasticity. Finally, bone attachments for our skeleton-driven simulations are handled via soft, embedded spring constraints as in [McAdams et al. 2011].

Dynamics Our formulations have so far been presented in the context of a quasistatic time evolution scheme. Nevertheless, our method extends naturally to implicit or semi-implicit time integration scenarios. As an example, we briefly outline the modifications necessary for combining our method with a fully implicit Backward Euler integration scheme, similar to the one detailed by Sifakis and Barbič [2012]. Due to our mixed formulation, the auxiliary pressure variable $\boldsymbol{p}$ as well as its time derivative $\dot{\boldsymbol{p}}$ need to be maintained as state variables. The Backward Euler scheme is formulated as:

$$\begin{aligned}
\boldsymbol{x}^{n+1} &= \boldsymbol{x}^n + \Delta t\boldsymbol{v}^{n+1} && (14)\\
\boldsymbol{p}^{n+1} &= \boldsymbol{p}^n + \Delta t\dot{\boldsymbol{p}}^{n+1} && (15)\\
\boldsymbol{v}^{n+1} &= \boldsymbol{v}^n + \Delta t\mathbf{M}^{-1}\left[\boldsymbol{f}(\boldsymbol{x}^{n+1},\boldsymbol{p}^{n+1}) + \right.\\
&\quad \left. + \boldsymbol{f}_d(\boldsymbol{x}^{n+1},\boldsymbol{p}^{n+1};\boldsymbol{v}^{n+1},\dot{\boldsymbol{p}}^{n+1})\right] && (16)\\
0 &= \boldsymbol{q}(\boldsymbol{x}^{n+1},\boldsymbol{p}^{n+1}) && (17)
\end{aligned}$$

Here, $\mathbf{M}$ is the mass matrix, and superscripts denote time steps. We define the term $\boldsymbol{f}_d$ according to a Rayleigh damping model as:

$$\boldsymbol{f}_d(\boldsymbol{x}, \boldsymbol{p}; \boldsymbol{v}, \dot{\boldsymbol{p}}) := \gamma \delta \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{p}; \delta \boldsymbol{x} \leftarrow \boldsymbol{v}, \delta \boldsymbol{p} \leftarrow \dot{\boldsymbol{p}})$$

where γ is the Rayleigh damping coefficient. Since the system of equations (14) through (17) is nonlinear as a whole, we solve it using an iterative Newton-Raphson procedure. Linearizing around an approximation $\boldsymbol{x}_k^{n+1}, \boldsymbol{p}_k^{n+1}, \boldsymbol{v}_k^{n+1}, \dot{\boldsymbol{p}}_k^{n+1}$ of the state variables at the end of the timestep (the k-th iterate of the Newton procedure), we obtain the equations:

$$\begin{gathered}\tfrac{1}{\Delta t^2}\mathbf{M}\Delta\boldsymbol{x} - \left(1 + \tfrac{\gamma}{\Delta t}\right)\delta\boldsymbol{f}(\boldsymbol{x}_k^{n+1}, \boldsymbol{p}_k^{n+1}; \Delta\boldsymbol{x}, \Delta\boldsymbol{p}) = \\ = \mathbf{M}(\boldsymbol{v}^n - \boldsymbol{v}_k^{n+1}) + \boldsymbol{f}(\boldsymbol{x}_k^{n+1}, \boldsymbol{p}_k^{n+1}) + \boldsymbol{f}_d(\boldsymbol{x}_k^{n+1}, \boldsymbol{p}_k^{n+1}; \boldsymbol{v}_k^{n+1}, \dot{\boldsymbol{p}}_k^{n+1})\end{gathered}$$

$$\text{and } -\left(1 + \tfrac{\gamma}{\Delta t}\right)\delta\boldsymbol{q}(\boldsymbol{x}_k^{n+1}, \boldsymbol{p}_k^{n+1}; \Delta\boldsymbol{x}, \Delta\boldsymbol{p}) = \boldsymbol{q}(\boldsymbol{x}_k^{n+1}, \boldsymbol{p}_k^{n+1})$$

This is a symmetric indefinite linear system of equations that can be solved with QMR or another appropriate solver, in a similar fashion as the system associated with our prior quasistatic time evolution. We can easily verify that these equations reduce to the quasistatic ones at the limit $\Delta t \to \infty$. Once the corrections $\Delta\boldsymbol{x}, \Delta\boldsymbol{p}$ have been computed, we proceed to update the state variables as $\boldsymbol{x}_{k+1}^{n+1} \leftarrow \boldsymbol{x}_k^{n+1} + \Delta\boldsymbol{x}$, $\boldsymbol{p}_{k+1}^{n+1} \leftarrow \boldsymbol{p}_k^{n+1} + \Delta\boldsymbol{p}$, $\boldsymbol{v}_{k+1}^{n+1} \leftarrow \boldsymbol{v}_k^{n+1} + \Delta t^{-1}\Delta\boldsymbol{x}$ and $\dot{\boldsymbol{p}}_{k+1}^{n+1} \leftarrow \dot{\boldsymbol{p}}_k^{n+1} + \Delta t^{-1}\Delta\boldsymbol{p}$.

Data organization By far, the most computationally expensive component of our technique is the force differential calculation, which occurs once at every QMR iteration. We are also confronted with a complex challenge if we choose to leverage parallelism and SIMD capabilities in modern processors: our models have irregular shapes only partially cover the embedding lattice, making optimal parallel iterators difficult to construct. In addition, we are faced with a data dependency issue: Computation, as currently structured is conducted on a cell-by-cell basis; however, the results of this computation $\boldsymbol{f}$ and $\boldsymbol{q}$ are stored on both cells and nodes, and we can have different (neighboring) cells that need to accumulate forces on the same common node, giving rise to dependencies and synchronization issues.

In order to streamline the force differential computation and generate the best conditions for parallelism and vectorization, we propose a new data organization scheme for our state variables and intermediate data. Our approach is illustrated in figure 7. We conceptually subdivide our 3-dimensional domain into blocks of 2^3 voxels each. Note that not all such voxels will participate in the grid, as some blocks near object boundaries will contain inactive voxels. However, due to the small size of the 2^3-sized blocks, the number of voxels included in all blocks that contain active voxels will be minimally larger than the number of active voxels. Prior to any computation (e.g. force differential calculation), we copy all nodal information from a grid-based array into a *flat* array of blocks, duplicating any shared values as necessary. Of course, this step entails creating multiple copies of data, but is not as expensive as if a separate copy of all nodal data was made for every individual voxel (the practical data overhead is $< 3x$ for this scheme, compared to $8x$ for a replication of all nodes for all cells). The cost of this data duplication is reduced by the fact that additional simulation metadata (such as pressures $\boldsymbol{c}$, material parameters, the matrix $\mathbf{Q}$, precomputed stress derivatives and Singular Value/Polar decompositions, if needed) which are conceptually cell-centered can be stored *persistently* in a flattened array of blocks. Once this translation is completed, fully balanced multi-threading is possible by simply subdividing the processing of this flattened array across computing threads. Within each thread, we leverage the 2^3 multiplicity of each block to compute differentials with SIMD instructions. The blocks of nodal and cell data are first copied from the heap-allocated flat arrays onto a stack-allocated copy. Then, we perform a final separation of cell and node data, creating one fully separate copy for each of the 8 voxels. Note that this operation does not incur memory bandwidth expense, since this local stack-allocated copy (typically less than $6 - 8KB$ in size) is expected to be cache-resident for the duration of the computation. We have leveraged the AVX instruction set of the Sandy Bridge architecture to process all 8 voxels of the block simultaneously. Upon completion of the local computation, the entire process is reversed: the voxel-specific forces are accumulated into nodal forces for the entire block (i.e. 8 sets of 8 nodal forces are accumulated into the 27 nodal force variables of the containing block) and copied to the flattened array of block data. Lastly, in a scatter/accumulate step, the forces from the flattened block array are added back onto the grid; naturally, this step requires attention to ordering to avoid data dependencies, but this very specific task is significantly easier to manage.

		1 core (AVX) 1 thread	2 cores (AVX) 2 threads	4 cores (AVX) 4 threads	4 cores (AVX) 8 threads
Human model (101K cells)	Force Differentials	0.041	0.023	0.015	0.015
	One QMR Iteration	0.067	0.037	0.025	0.024
	Newton Iteration (including 100 QMR iterations)	7.035	4.066	2.727	2.645
	Typical frame (4-6 Newton iterations)	31.825	18.186	12.193	11.919
Human model (13.5K cells)	Force Differentials	0.005	0.003	0.002	0.002
	One QMR Iteration	0.007	0.005	0.003	0.003
	Newton Iteration (including 50 QMR iterations)	0.422	0.244	0.184	0.177
	Typical frame (3-5 Newton iterations)	1.466	0.842	0.640	0.626

Figure 6: *Runtimes of our human body simulation examples, in seconds. All times are in seconds, reported on an Intel Core i7-2600 CPU and include the use of AVX instructions. (Note: The last column reflects an 8-thread execution, but only on 4 physical cores, via hyperthreading.)*

6 Examples

We demonstrate the use of our system in soft-tissue musculoskeletal simulations using the material model and parameters presented in [Lee et al. 2009]. Figure 1 depicts a keyframed walking sequence, where the deformation of flesh, skin and muscles was generated by our technique. As our focus was not on the derivation of realistic muscle activation sequences, we demonstrated just the two extreme cases in our simulations: all muscles being inactive (the passive component is still present), or all muscles contracted to maximum activation. Two discretization resolutions were used: a lattice with a grid size of $10mm$, yielding a lattice with approximately $101K$ voxels, and a coarser one with grid size $20mm$ containing approximately $13.5K$ voxels. Note that, for comparison with tetrahedral discretizations, our high-resolution grid would be comparable (in number of degrees of freedom) with a tetrahedral discretization containing about $500K - 600K$ tetrahedra. The supplemental submission video demonstrates additional motion sequences, and highlights instances where our use of a sub-voxel accurate method results in much pronounced muscle definition, and enables bending and folding of flesh near skin creases. Figure 6 reports the runtime cost on a 4-core Sandy Bridge CPU. Please note that the times given for full frames, or full Newton-Raphson iterations are highly variable and depend on the complexity of the simulated model; the numbers given here were characteristic of our specific simulations.

As previously noted by Zhu et al. [2010] reconstructing the embedded object surface using trilinear interpolation from lattice nodes yields visible grid artifacts due to discontinuous gradients at voxel boundaries. Consequently, our examples employed tricubic inter-

polation [Lekien and Marsden 2005] to generate the embedded object surface for rendering, with the exception of Figure 5 where standard trilinear interpolation was used, to demonstrate this effect and emphasize that the trilinear basis is still capable of rapid convergence under refinement when coupled with a sub-voxel accurate quadrature scheme.

7 Limitations and future work

Our proposed approach sought to bridge the feature sets of embedded lattice deformers and conforming discretizations. There is a number of aspects, however, in which our lattice-based approach remains less versatile than conforming meshes. Explicit tetrahedral models offer direct control over their nodal degrees of freedom, for the purposes of contact/collision resolution and enforcement of kinematic constraints. Although it would be possible to handle such tasks to a certain extent via embedding and soft constraints [Sifakis et al. 2007] within a lattice deformer, such hybrid treatments may compromise the regularity of our data structures and affect performance accordingly. In addition, although our method is able to resolve boundary geometry at sub-voxel resolution, the use of an implicitly defined embedding lattice restricts its ability to resolve small scale *topology*. Examples of this limitation would be face models where the two would effectively be tied together, if spanned by the same lattice cell, or fingers in human models that cannot move independently unless separated by a layer of whole voxels. It is relatively straightforward to leverage adaptivity in conforming discretizations, while lattice-based discretizations require nontrivial modifications to be effective and functional in an adaptive setting.

Our method handles the discretization of boundary equations (or *Neumann* conditions in PDE terminology) with sub-voxel precision, and would justifiably be considered a "second order" discretization with respect to free boundary treatment. We note however that *Dirichlet* conditions (i.e. kinematic constraints) are not currently handled to second order (sub-voxel) accuracy. Instead, we either specify kinematic constraints directly on lattice nodes, or implement soft embedded kinematic constraints via spring attachments. Techniques do exist for the embedded enforcement of Dirichlet conditions in ways that preserve second-order convergence [Hellrung et al. 2012] but they have not been exhaustively tested or even demonstrated in graphics applications. Our paper has focused on quasi-static (equilibrium) simulation and omitted collision detection/processing in its current form. The methodology would be expected to extend trivially to dynamic simulation, subject to careful determination of mass and damping properties near boundaries. Collision processing however merits special attention in future work; although penalty-based collision handling can be used, this treatment could affect the convergence behavior of numerical solvers, and compromise the regularity of the parallel data structures we employ. Self-collisions would introduce nonlocal forces which would need to be handled separately from our SIMD-optimized data organization methodology.

Additionally, a notable suboptimal trait of our current approach (and a promising thread for future development) is the fact that the Krylov methods we employ to solve the discrete equations are not the best solvers one could use for this purpose. Solvers (or preconditioners) based on multigrid or domain decomposition principles [Quarteroni and Valli 1999; Trottenberg et al. 2001] have demonstrated excellent potential for near-linear scalability on elliptic problems (such as elasticity) and are very well suited for parallel execution. In addition some of our techniques, such as the high-order defect correction procedure, originated from multigrid theory and achieve their ideal efficiency in that context.

Our use of Krylov subspace solvers is associated with another important limitation: The cost per iteration in a solver such as QMR is partly attributed to the computation of force differentials, while a fraction of execution time is spent on streaming vector operations, such as constant scaling, vector additions and inner product computations. In many of our tests, these memory bandwidth intensive streaming operations became a performance bottleneck after the force-related computations had been aggressively optimized via vectorization and multithreading. In the test runs documented in Figure 6 we observed that in a parallel 4-core execution most computational kernels (and certainly the streaming vector operations in QMR) were constrained by memory bandwidth. Although it may be possible to increase scalability by further optimizing our implementation for memory bandwidth, we believe that this issue would be addressed more appropriately by moving towards multigrid or domain decomposition solvers (or preconditioners) which provide greater flexibility for trading CPU load for reduced bandwidth demands. As an example, we point to the work of McAdams et al. [2010] who demonstrated parallel speedup in excess of $8\times$ on a multigrid-preconditioned Krylov solver. Lastly, our implementation focused on multicore desktop platforms, but we did not demonstrate a GPU port; a proper port and analysis is left to future work.

Acknowledgements

We would like to thank Ben Recht for many insightful discussions and his valuable feedback on our numerical quadrature scheme.

References

ALMGREN, A., BELL, J., COLELLA, P., AND MARTHALER, T. 1997. A Cartesian grid projection method for the incompressible Euler equations in complex geometries. *SIAM Journal on Scientific Computing 18*, 5, 1289–1309.

ARNOLD, D. N. 1990. Mixed finite element methods for elliptic problems. *Computer Methods in Applied Mechanics and Engineering 82*, 1-3, 281–300.

BATTY, C., BERTAILS, F., AND BRIDSON, R. 2007. A fast variational framework for accurate solid-fluid coupling. *ACM Transactions on Graphics (SIGGRAPH Proceedings) 26*, 3 (July).

BREZZI, F., AND FORTIN, M. 1991. *Mixed and hybrid finite element methods*. Springer-Verlag: New York.

CHUANG, M., LUO, L., BROWN, B. J., RUSINKIEWICZ, S., AND KAZHDAN, M. 2009. Estimating the Laplace-Beltrami operator by restricting 3D functions. In *Proceedings of the Symposium on Geometry Processing*, 1475–1484.

DAUX, C., MOES, N., DOLBOW, J., SUKUMAR, N., AND BELYTSCHKO, T. 2000. Arbitrary branched and intersecting cracks with the extended finite element method. *International Journal for Numerical Methods in Engineering 48*, 12.

ENGLISH, E., AND BRIDSON, R. 2008. Animating developable surfaces using nonconforming elements. In *ACM Transactions on Graphics (SIGGRAPH Proceedings)*, vol. 27, 66.

FEDKIW, R., ASLAM, T., MERRIMAN, B., AND OSHER, S. 1999. A non-oscillatory eulerian approach to interfaces in multimaterial flows (the ghost fluid method). *Journal of Computational Physics 152*, 2, 457–492.

FREUND, R., AND NACHTIGAL, N. 1994. A new Krylov-subspace method for symmetric indefinite linear systems. In *Proceedings of the 14th IMACS World Congress on Computational and Applied Mathematics*, IMACS: New Brunswick, NJ, 1253–1256.

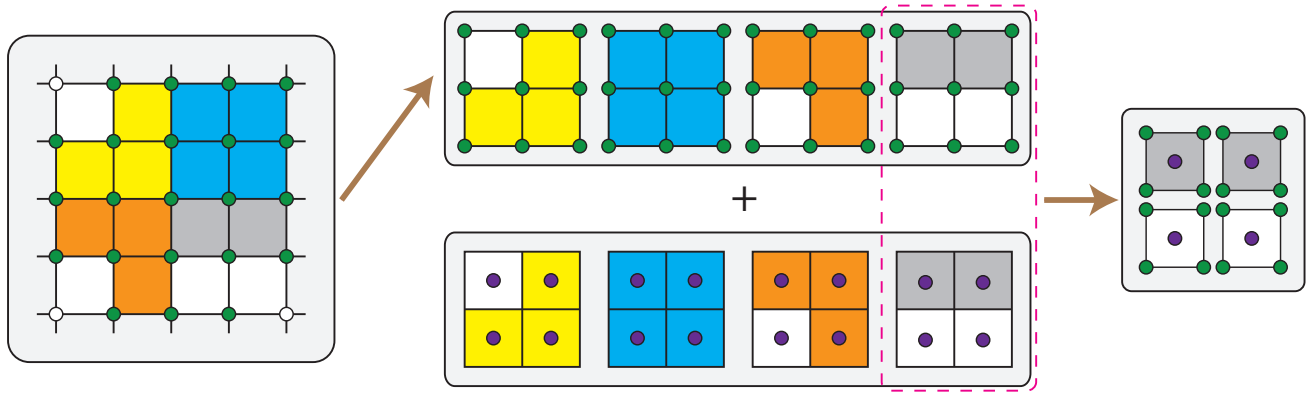

Figure 7: *A 2D illustration of our simulation data structures. Left: Nodal (deformation) data stored on a grid. Middle: On demand, nodal data is copied to an array of 2^d-sized blocks and combined with cell-based data which are persistently stored in arrays of blocks. Right: Nodal and cell-centered data for a single block are copied to a stack-allocated structure, and duplicated for each voxel for SIMD computation.*

GEORGII, J., AND WESTERMANN, R. 2008. Corotated finite elements made fast and stable. In *Proceedings of the 5th Workshop On Virtual Reality Interaction and Physical Simulation*.

GOLDENTHAL, R., HARMON, D., FATTAL, R., BERCOVIER, M., AND GRINSPUN, E. 2007. Efficient simulation of inextensible cloth. In *ACM Transactions on Graphics (TOG)*, vol. 26, 49.

GUENDELMAN, E., SELLE, A., LOSASSO, F., AND FEDKIW, R. 2005. Coupling water and smoke to thin deformable and rigid shells. *ACM Transactions on Graphics (SIGGRAPH Proceedings) 24*, 3, 973–981.

HELLRUNG, JR., J. L., WANG, L., SIFAKIS, E., AND TERAN, J. M. 2012. A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions. *Journal of Computational Physics 231*, 4, 2015–2048.

HUGHES, T. 1987. *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*. Prentice Hall.

IRVING, G., SCHROEDER, C., AND FEDKIW, R. 2007. Volume conserving finite element simulations of deformable models. *ACM Transactions on Graphics (SIGGRAPH Proc.) 26*, 3.

KHAREVYCH, L., MULLEN, P., OWHADI, H., AND DESBRUN, M. 2009. Numerical coarsening of inhomogeneous elastic materials. *ACM Trans. on Graphics (SIGGRAPH Proc.) 28*, 3, 51.

KIM, J., AND POLLARD, N. 2011. Fast simulation of skeleton-driven deformable body characters. *ACM Transactions on Graphics (TOG) 30*, 5, 121.

LEE, S.-H., SIFAKIS, E., AND TERZOPOULOS, D. 2009. Comprehensive biomechanical modeling and simulation of the upper body. *ACM Transactions on Graphics 28*, 4, 1–17.

LEKIEN, F., AND MARSDEN, J. 2005. Tricubic interpolation in three dimensions. *Journal of Numerical Methods and Engineering 63*, 455–471.

MCADAMS, A., SIFAKIS, E., AND TERAN, J. 2010. A parallel multigrid Poisson solver for fluids simulation on large grids. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 65–74.

MCADAMS, A., ZHU, Y., SELLE, A., EMPEY, M., TAMSTORF, R., TERAN, J., AND SIFAKIS, E. 2011. Efficient elasticity for character skinning with contact and collisions. *ACM Transactions on Graphics (SIGGRAPH Proceedings) 30*, 4, 37.

MÜLLER, M., TESCHNER, M., AND GROSS, M. 2004. Physically-based simulation of objects represented by surface meshes. In *Proc. Computer Graphics International*, 156–165.

NESME, M., PAYAN, Y., AND FAURE, F. 2006. Animating shapes at arbitrary resolution with non-uniform stiffness. In *Eurographics Workshop in Virtual Reality Interaction and Physical Simulation (VRIPHYS)*.

NESME, M., KRY, P., JEŘÁBKOVÁ, L., AND FAURE, F. 2009. Preserving topology and elasticity for embedded deformable models. In *ACM Transactions on Graphics (SIGGRAPH Proceedings)*, vol. 28, 52.

QUARTERONI, A., AND VALLI, A. 1999. *Domain decomposition methods for partial differential equations*, vol. 10. Clarendon Press.

RIVERS, A., AND JAMES, D. 2007. FastLSM: fast lattice shape matching for robust real-time deformation. *ACM Transactions on Graphics (SIGGRAPH Proc.) 26*, 3.

SIFAKIS, E., AND BARBIČ, J., 2012. FEM simulation of 3D deformable solids: A practitioner's guide to theory, discretization, and model reduction. ACM SIGGRAPH 2012 Courses. `http://www.femdefo.org/`.

SIFAKIS, E., SHINAR, T., IRVING, G., AND FEDKIW, R. 2007. Hybrid simulation of deformable solids. In *Proc. of ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 81–90.

TERAN, J., SIFAKIS, E., IRVING, G., AND FEDKIW, R. 2005. Robust quasistatic finite elements and flesh simulation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM, 181–190.

TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. 1987. Elastically deformable models. *Computer Graphics (Proc. SIGGRAPH 87) 21*, 4, 205–214.

TROTTENBERG, U., OOSTERLEE, C., AND SCHULLER, A. 2001. *Multigrid*. San Diego: Academic Press.

WOJTAN, C., AND TURK, G. 2008. Fast viscoelastic behavior with thin features. In *ACM Transactions on Graphics (SIGGRAPH Proc.)*, vol. 27, ACM, 47.

ZHU, Y., SIFAKIS, E., TERAN, J., AND BRANDT, A. 2010. An efficient multigrid method for the simulation of high-resolution elastic solids. *ACM Transactions on Graphics (TOG) 29*, 2, 16.