An efficient parallelizable multigrid framework for the simulation of elastic solids

Figure 1: *Lookit! Lookit!*

Abstract

TBD

CR Categories: K.6.1 [Management of Computing and Information Systems]: Project and People Management—Life Cycle; K.7.m [The Computing Profession]: Miscellaneous—Ethics

Keywords: multigrid, elasticity, parallelization

1 Introduction

TBD

2 Background

We review certain fundamental concepts of elasticity theory, initially focusing on a static linear elasticity formulation, followed by a general outline of a multigrid solver for this elliptic problem.

2.1 Linear elasticity

We represent the deformation of a elastic volumetric object using a *deformation function* $\boldsymbol{\phi}$ which maps any material point $\boldsymbol{X}^*$ of the undeformed configuration of the object, to its position $\boldsymbol{x}^*$ in the deformed configuration, i.e. $\boldsymbol{x}^* = \boldsymbol{\phi}(\boldsymbol{X}^*)$. For simplicity, we will use the symbol $\boldsymbol{\phi}$ to denote both this deformation function, as well as the deformed position of a material point as $\boldsymbol{\phi}^* = \boldsymbol{\phi}(\boldsymbol{X}^*)$. A deformation of an object gives rise to elastic forces aiming to restore the object to an equilibrium configuration [Bonet and Wood 1997]. These forces are analytically given by the divergence form

$$\boldsymbol{f} = -\boldsymbol{\nabla}^T\mathbf{P} \quad \text{or, componentwise} \quad f_i = -\textstyle\sum_j \partial_j Pij. \qquad (1)$$

where $\mathbf{P}$ is the first Piola-Kirchhoff stress tensor. We note that the partial derivatives in this and all subsequent formulas are taken with respect to the material (undeformed) coordinates. The stress tensor $\mathbf{P}$ is computed from the deformation map $\boldsymbol{\phi}$; the analytic expression that defines the dependence of $\mathbf{P}$ on $\boldsymbol{\phi}$, known as the *constitutive equation*, is an instrinsic property of every elastic material.

We will henceforth adopt the common conventions of using subscripts after a comma to denote partial derivatives, and omit certain summation symbols by implicitly summing over any right-hand side indices that do not appear on the left-hand side of a given equation. Consequently, equation (1) is compactly written as $f_i = -P_{ij,j}$. The constitutive equation of linear elasticity is

$$\mathbf{P} = 2\mu\boldsymbol{\epsilon} + \lambda\text{tr}(\boldsymbol{\epsilon})\mathbf{I} \quad \text{or} \quad P_{ij} = 2\mu\epsilon_{ij} + \lambda\epsilon_{kk}\delta_{ij} \qquad (2)$$

In this equation, μ and λ are the Lamé parameters of the linear material, and are computed from Young's modulus E (a measure of material stiffness) and Poisson's ratio ν (a measure of material incompressibility) as $\mu = E/(2+2\nu)$, $\lambda = E\nu/((1+\nu)(1-2\nu))$. Also, δ_{ij} is the Kronecker delta, $\boldsymbol{\epsilon}$ is the small strain tensor

$$\boldsymbol{\epsilon} = \tfrac{1}{2}(\mathbf{F} + \mathbf{F}^T) - \mathbf{I} \quad \text{or} \quad \epsilon_{ij} = \tfrac{1}{2}(\phi_{i,j} + \phi_{j,i}) - \delta_{ij} \qquad (3)$$

and $\mathbf{F}$ is the deformation gradient tensor, defined as $F_{ij} = \phi_{i,j}$. Using (1,2,3) we derive the differential equation of linear elasticity

$$f_i = -\mu\phi_{i,jj} - (\mu+\lambda)\phi_{j,ij} = \mathcal{L}_{ij}\phi_j \qquad (4)$$

In this equation $\mathcal{L} = -\mu\Delta\mathbf{I} - (\mu+\lambda)\boldsymbol{\nabla}\boldsymbol{\nabla}^T$ is the partial differential operator of linear elasticity. A static elasticity problem amounts to determining the deformation map $\boldsymbol{\phi}$ that leads to an equilibrium of the total forces on a deformable object, i.e. $\mathcal{L}\boldsymbol{\phi} + \boldsymbol{f}_{\text{ext}} = 0$, where $\boldsymbol{f}_{\text{ext}}$ are the external forces applied on the object. Substituting $\boldsymbol{f} = -\boldsymbol{f}_{\text{ext}}$, the static elasticity problem becomes equivalent to the linear partial differential equation $\mathcal{L}\boldsymbol{\phi} = \boldsymbol{f}$.

2.2 Multigrid correction scheme

Multigrid techniques have been predominatly targeted towards elliptic partial differential equations, such as the preceeding formulatin of elasticity. Although their is a broad gamut of multigrid cycles and schemes, the overall philosophy is well reflected in the multigrid V-cycle correction scheme which will be described in this section. Alternative schemes will be discussed in section 13.

Multigrid methods are based on the concept of a *smoother* which is a procedure resigned to smooth, and at the same time reduce the magnitude of the residual $\boldsymbol{r} = \boldsymbol{f} - \mathcal{L}\boldsymbol{\phi}$ of the differential equation, by modifying the current estimate of the unknown function $\boldsymbol{\phi}$. For example, if the differential equation in question has been discretized into a system of linear equations, Gauss-Seidel or Jacobi iteration could be two candidates for a simple smoother. An inherent property of elliptic systems is that a smooth distribution of residuals generally implies a certain degree of smoothness in the *error* $\boldsymbol{e} = \boldsymbol{\phi} - \boldsymbol{\phi}_{\text{exact}}$ as well, although a careful discretization is often needed to guarantee that this property is reflected in the discrete form of the problem. Smoothers are typically simple, local and relatively inexpensive routines, which are quite efficient at eliminating high frequencies of the residual (and, as a consequence, of the error). Nevertheless, once the high frequency component of the error has been eliminated as a result of a few applications of the smoother, subsequent iterations are characterized by rapidly decelerated convergence towards the solution. Multigrid methods seek to remediate this effect of smoother *stagnation*, using the smoother as a building block to construct a solver that achieves constant rate of convergence towards the solution, regardless of the prevailing frequencies of the residual or error. This property is acomplished by observing that any lower frequency error that persists after a few smoothing iterations will appear to be higher frequency if the problem is resampled using a coarser discretization step. By transitioning to ever coarser discretizations the smoother retains it ability to make significant progress towards the solution of the problem.

The components of a multigrid algorithm are:

- The discretization of the continuous operator $\mathcal{L}$ at a number of different resolutions, denoted as $\mathcal{L}^h, \mathcal{L}^{2h}, \mathcal{L}^{4h}$ and so on.
- The Smoothing subroutine, defined at each resolution.
- The Prolongation and Restriction subroutines. These implement an upsampling and downsampling operation respectively, between two different levels of resolution.

- An exact solver, used for solving the partial differential equation at the coarsest discretization resolution. Any reasonable solver may be used, as the small size of the coarsest problem is expected to lead to a negligible runtime for this subroutine.

Table 1 Multigrid Correction Scheme – V(1,1) Cycle

procedure MULTIGRID($\boldsymbol{\phi}, \boldsymbol{f}, L$) ▷ $\boldsymbol{\phi}$ is the current estimate
  $\boldsymbol{u}^h \leftarrow \boldsymbol{\phi}, \boldsymbol{b}^h \leftarrow \boldsymbol{f}$ ▷ total of $L{+}1$ levels
  for $l = 0$ **to** $L{-}1$ **do**
    Smooth($\mathcal{L}^{2^l h}$,$\boldsymbol{u}^{2^l h}$,$\boldsymbol{b}^{2^l h}$)
    $\boldsymbol{r}^{2^l h} \leftarrow \boldsymbol{b}^{2^l h} - \mathcal{L}^{2^l h}\boldsymbol{u}^{2^l h}$
    $\boldsymbol{b}^{2^{l+1} h} \leftarrow$Restrict($\boldsymbol{r}^{2^l h}$)
    $\boldsymbol{u}^{2^{l+1} h} \leftarrow 0$
  end for
  Solve $\boldsymbol{u}^{2^L h} \leftarrow (\mathcal{L}^{2^L h})^{-1}\boldsymbol{b}^{2^L h}$
  for $l = L{-}1$ **down to** 0 **do**
    $\boldsymbol{u}^{2^l h} \leftarrow \boldsymbol{u}^{2^l h}$+Prolongate($\boldsymbol{u}^{2^{l+1} h}$)
    Smooth($\mathcal{L}^{2^l h}$,$\boldsymbol{u}^{2^l h}$,$\boldsymbol{b}^{2^l h}$)
  end for
  $\boldsymbol{\phi} \leftarrow \boldsymbol{u}^h$
end procedure

Table 1 gives the pseudocode for a V(1,1) cycle of the Multigrid correction scheme, Notably, this description does not presume a specific discretization, or a particular implementation of the smoothing, restriction or prolongation operators. The following sections detail our specific implementation of these components, and the factors that motivated these design decisions.

3 Discretization

Our method uses a staggered finite difference discretization on uniform grids. This is a familiar concept in the field of computational fluid dynamics (e.g. [Fedkiw et al. 2001]) where staggered finite difference methods are commonplace. In contrast, these formulations are less widespread in the simulation of solids, especially for computer graphics applications, where unstructured meshes coupled with finite element or mass-spring methods are more common. This trend is generally justified by the all-around geometric and algorithmic versatility of these formulations. Nevertheless, finite difference based approaches to elasticity have been investigated [Terzopoulos et al. 1987; Terzopoulos and Fleischer 1988], and a number of authors [Müller et al. 2004; James et al. 2004; Rivers and James 2007] have turned to regular grid representations for reasons of efficiency, even for discretizations other than finite differences.

Regular vs. Unstructured grids Our main motivation for using a discretization based on regular grids, is avoiding the storage and access overhead of an unstructured mesh. Consider the example of a deformable model, discretized using 100K vertices. Representing the state (e.g. nodal positions) of this model would require 1.2MB of storage, using single-precision. As a rule of thumb, a tetrahedral representation using the same number of vertices would typically have more than 400K tetrahedra and require at least 6.4MB of storage to represent the mesh alone. In a parallel, streaming computing platform, this explicit representation of topology would compete with the representation of the state variables for limited cache and bandwidth resources. Additionally, unstructured meshes generally require indirect memory access or scatter-gather mechanisms which may lead to suboptimal utilization of the available bandwidth. Furthermore, uniform grids allow the use of constant stencils for the restriction and prolongation operations (and in some cases, for the discrete PDE operator $\mathcal{L}^h$ itself), whereas this data would have to be precomputed and streamed from memory in the case of unstructured grids. In section 12 we give further details on the working set size reductions enabled by our uniform discretization. Of course, a different measure of comparison should be established between uniform grids and unstructured, yet highly *adaptive* meshes, a topic discussed in more detail in section 13. Overall, our approach delivers very good performance at high resolution levels that compensate for the lack of a conforming or adaptive geometry.

Finite differences vs. Finite elements In principle, either finite difference or finite element discretizations would have been viable options for our system, assuming that either method would be used with a regular grid. Our framework is based on finite differences as this method enables us to obtain a sparser, more compact discrete system of equations. This property enables computational savings and more efficient utilization of resources such as memory bandwidth and stream buffers. In 3D, for example, each of the equations of linear elasticity $\mathcal{L}_i = -\mu\partial_{jj} - (\mu + \lambda)\partial_{ij}$ can be discretized to second-order accuracy using a stencil of 15 nonzero coefficients. For comparison, the finite element discretization of [Kazhdan and Hoppe 2008] yields 25 nonzero coefficients per equation for the much simpler 2D Poisson problem. Discretizing the 3D linear elasticity equations using trilinear finite elements and 8-point Gauss quadrature would typically require 81 nonzero coefficients per equation. Care needs to be taken however since certain useful properties of finite elements, such as symmetry and convenient treatment of certain boundaries, are not automatically guaranteed in a finite difference scheme. A significant fraction of our paper is dedicated to the implementation strategies necessary to retain these desired properties within our finite diffence scheme.

Staggered vs. Collocated grids The deformation map $\boldsymbol{\phi}$ is a vector-valued function of the material coordinate vector $\boldsymbol{X}$. Thus $\boldsymbol{\phi}(\boldsymbol{X})=(\phi_1(\boldsymbol{X}), \phi_2(\boldsymbol{X}), \phi_3(\boldsymbol{X}))$ where each ϕ_i is a scalar-valued function. When discretizing these quantities, it would be most intuitive to use *collocated* grids, where all components of $\boldsymbol{\phi}$ are specified at the same location, for example at the nodes of a background grid. Unfortunately, for the equations of elasticity such a discretization may result in grid-scale oscillations, especially for near-incompressible materials. A comprehensive study of the causes and consequences of this behavior specifically for the equations of elasticity is beyond the scope of our current exposition. It is however qualitatively analogous to an artifact observed in the simulation of fluids with non-staggered grids, where spurious oscillations may be left over in the pressure field after a Poisson solver has been used to project a velocity field to its divergence-free component. In the context of multigrid methods, such oscillatory discretizations can be far more problematic, as they may not respect the fundamental property of elliptic PDEs that a low residual implies a smooth error, requiring more elaborate and expensive smoothers to compensate. We avoid this issue altogether by adopting a staggered discretization, which is free of this oscillatory behavior, and aligns naturally with the rest of our theoretical formulations.

Our staggered discretization is illustrated in Figure 2. A background cartesian grid serves as a reference for the placement of the unknown variables. Each component ϕ_i of the deformation function $\boldsymbol{\phi}$ is stored in a separate cartesian lattice, which is offset from the nodes of the background reference grid. Specifically, ϕ_i variables are stored at the centers of the background grid faces perpendicular to the cartesian axis vector $\boldsymbol{e}_i$. For example, ϕ_1 values are stored on grid faces perpendicular to $\boldsymbol{e}_1$, i.e. those parallel to the yz-plane. The same strategy is followed in 2D, where faces of grid cells are now identified with grid edges, thus ϕ_1 values are stored

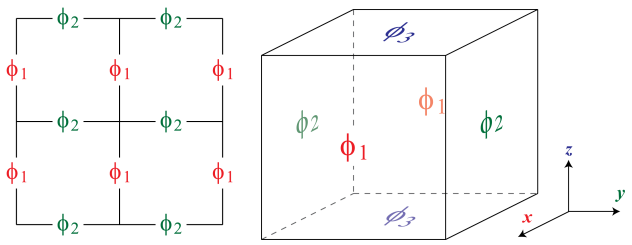

Figure 2: *Staggering of variables in 2D(left) and 3D(right). Equations $\mathcal{L}_1$,$\mathcal{L}_2$,$\mathcal{L}_3$ are also stored on ϕ_1,ϕ_2,ϕ_3 locations respectively.*

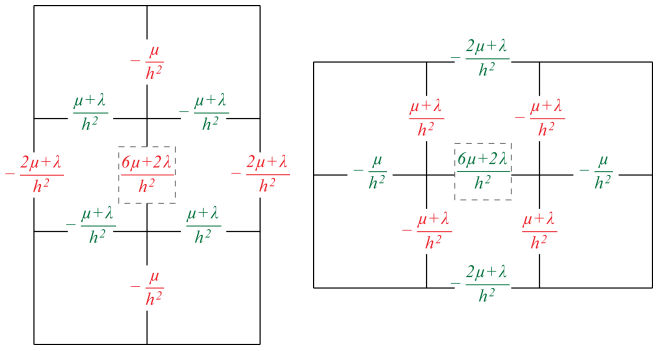

Figure 3: *Discrete stencils for operators $\mathcal{L}_1$(left) and $\mathcal{L}_2$(right) of the PDE system (4). The red and green nodes of the stencil correspond to ϕ_1 and ϕ_2 values respectively. The dashed square indicates the center of the stencil, where the equation is evaluated.*

at the center of y-oriented edges, and ϕ_2 values at the center of x-oriented edges. We proceed to define the discrete approximations to *first-order* derivatives using central finite differences

$$D_1 u[x,y,z] = u[x+\tfrac{1}{2}h_x, y, z] - u[x-\tfrac{1}{2}h_x, y, z]$$
$$D_2 u[x,y,z] = u[x, y+\tfrac{1}{2}h_y, z] - u[x, y-\tfrac{1}{2}h_y, z]$$
$$D_3 u[x,y,z] = u[x, y, z+\tfrac{1}{2}h_z] - u[x, y, z-\tfrac{1}{2}h_z]$$

where (h_x, h_y, h_z) are the dimensions of the cells of the background grid. *Second-order* derivative stencils are simply defined as the composition of two first-order stencils, i.e. $D_{ij} = D_i D_j$. An implication of these definitions is that the discrete first derivative of a certain quantity will *not* be collocated with it. For example, all derivatives of the form $D_i\phi_i$ are naturally defined at cell centers, while $D_1\phi_2$ is located at centers of z-oriented edges in 3D, and at grid nodes in 2D. This is not a problem, however, as all second-order derivatives are centered at the appropriate locations for a convenient discretization of (4). In particular, all stencils involved in the discretization of equation $\mathcal{L}_i$ are naturally centered on the location of variable ϕ_i. Thus, the staggering of the unknown variables implies a natural staggering of the discretized differential equations. Figure 3 illustrates this fact in 2D, where the discrete stencils for the operators $\mathcal{L}_1$ and $\mathcal{L}_2$ form the system (4) are shown to be naturally centered at ϕ_1 and ϕ_2 variable locations, respectively.

4 Construction of the Smoothing operator

The staggered discretization described in the previous section could be used essentially unmodified in the case of a relatively compressible material, i.e. with a Poisson's ratio ν not exceeding 0.2-0.3. However, a majority of materials of interest to computer graphics, including the muscles and flesh of animated characters, are highly incompressible. A number of authors [Irving et al. 2007; English and Bridson 2007; Kaufmann et al. 2008] have discussed the challenges presented by the simulation of near-incompressible materials and proposed techniques to cope with this problem. For a multigrid solver, naive use of standard smoothers (e.g. Gauss-Seidel) in the presence of high incompressibility could lead to complications such as slow convergence or even loss of stability. We present a comprehensive and computationally inexpensive solution to this problem for linear elasticity, practically achieving performance independent of the material parameters. Moreover, in sections 7 and 8 we show how this treatment generalizes to nonlinear constitutive models.

4.1 Augmentation and stable discretization

Simulation of near-incompressible materials is known to cause numerical complications to a range of solvers, including techniques based on finite element, finite difference or other discretizations. In particular, our PDE formulation also requires special treatment in this case. When Poisson's ratio approaches the incompressible limit $\nu \to 0.5$, the Lamé parameter λ becomes several orders of magnitude larger than μ. As a consequence, the dominant term of the elasticity operator $\mathcal{L} = -\mu\Delta\mathbf{I} - (\mu+\lambda)\boldsymbol{\nabla}\boldsymbol{\nabla}^T$ is the rank deficient operator $-(\mu+\lambda)\boldsymbol{\nabla}\boldsymbol{\nabla}^T$; thus $\mathcal{L}$ becomes *near-singular*. More specifically, we see that any divergence-free field $\boldsymbol{\phi}$ will be in the nullspace of the dominant term, i.e. $-\lambda\boldsymbol{\nabla}\boldsymbol{\nabla}^T\boldsymbol{\phi} = 0$. Thus, a solution to the elasticity PDE $\mathcal{L}\boldsymbol{\phi} = \boldsymbol{f}$ could be perturbed by a divergence-free displacement of substantial amplitude, without introducing a large residual for the differential equation. In the context of a multigrid scheme, this observation has far deeper implications other than indicating that the discretized system of equations will have a high condition number; here, the eigenspace corresponding to the smallest eigenvalues of the operator $\mathcal{L}$ contains the entire class of divergence-free functions. These can be arbitrarily oscillatory, and lead to high-frequency errors that the multigrid method cannot smooth efficiently or correct using information from a coarser grid. Fortunately, this complication is not a result of inherently problematic material behavior, but rather an artifact of the form of the governing equations chosen to describe this physical phenomenon. Our solution is to reformulate the PDEs of elasticity into an equivalent system, which does not suffer from the near-singularity of the original differential operator. This stable differential description of near-incompressible elasticity is adapted from the theory of mixed variational formulations [Brezzi and Fortin 1991] and was demonstrated by [Gaspar et al. 2008] in simple academic problems of linear elasticity. Our work extends these formulations to nonlinear materials and domains with arbitrary boundaries.

We introduce a new auxiliary variable p (which we will refer to as the *pressure*) defined as $p = -(\lambda/\mu)\boldsymbol{\nabla}^T\boldsymbol{\phi} = -(\lambda/\mu)\mathbf{div}\boldsymbol{\phi}$. Note that, although p is a scaled divergence of the deformation field, we do not pursue or depend on any explicit associations of this variable with any "physical" pressure quantity. We can now write

$$\begin{aligned}\mathcal{L}\boldsymbol{\phi} &= -\mu\Delta\boldsymbol{\phi} - (\mu+\lambda)\boldsymbol{\nabla}\boldsymbol{\nabla}^T\boldsymbol{\phi} \\ &= -\mu(\Delta\mathbf{I} + \boldsymbol{\nabla}\boldsymbol{\nabla}^T)\boldsymbol{\phi} - \lambda\boldsymbol{\nabla}(\boldsymbol{\nabla}^T\boldsymbol{\phi}) \\ &= -\mu(\Delta\mathbf{I} + \boldsymbol{\nabla}\boldsymbol{\nabla}^T)\boldsymbol{\phi} + \mu\boldsymbol{\nabla}p \qquad (5)\end{aligned}$$

As a result, the equilibrium equation $\mathcal{L}\boldsymbol{\phi} = \boldsymbol{f}$ can be equivalently written as the system

$$\begin{pmatrix} -\mu(\Delta\mathbf{I}+\boldsymbol{\nabla}\boldsymbol{\nabla}^T) & \mu\boldsymbol{\nabla} \\ \mu\boldsymbol{\nabla}^T & \frac{\mu^2}{\lambda} \end{pmatrix} \begin{pmatrix} \boldsymbol{\phi} \\ p \end{pmatrix} = \begin{pmatrix} \boldsymbol{f} \\ 0 \end{pmatrix} \qquad (6)$$

The top part of system (6) follows directly from equation (5), while the bottom equation is simply a rescaled version of the definition of

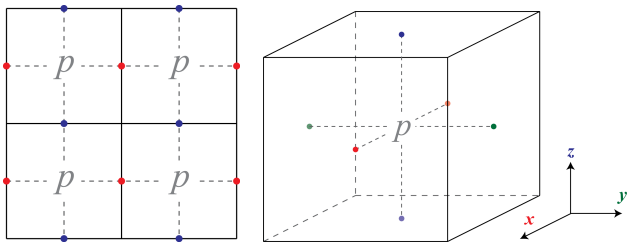

Figure 4: *Placement of pressures in 2D (left) and 3D (right).*

pressure p. Conversely, the original differential equation (4) can be obtained from (6) by simply eliminating the pressure variable. Thus the *augmented* differential equation system of (6) is equivalent to the governing equations of linear elasticity (e.g. the two systems agree in the value of $\boldsymbol{\phi}$ when solved). The important consequence of this manipulation is that this new discretization is *stable*, in the sense that the eigenspace corresponding to the smaller eigenvalues of this augmented differential operator does not contain high frequency deformation modes. This property can be rigorously proven via Fourier analysis; we can verify however that as λ tends to infinity, the term μ^2/λ vanishes, and the resulting limit system is now *non-singular*. This stability property indicates that it is possible to smooth the error of this system efficiently with inexpensive, local smoothers. Such a smoother is described in detail in section 4.2.

The newly introduced pressure variables are also discretized on an offset cartesian lattice, with each pressure begin stored in the center of a cell of the background lattice (see Figure 4). Pressure *equations*, i.e. the last row of system (6), are also centered and stored at cell centers. As was the case with the non-augmented elasticity equations, the staggering of deformation ($\boldsymbol{\phi}$) and pressure (p) variables is such that all discrete fist and second order differential operators are well defined where they are naturally needed. Finally, as a consequence of our staggered discretization, equations (4) and (6) are equivalent at the differential *and* at the discretization level, i.e. the discrete forms of the equations are algebraically identical, after eliminating pressures from the augmented form.

4.2 Distributive smoothing

Although the augmented system (6) has the necessary stability to admit, in principle, efficient local smoothing, this cannot be accomplished with a standard Gauss-Seidel or Jacobi iteration, as the discrete augmented equations lack the formal convergence guarantees of these methods. First, we note that the discrete form of system (6) is not a symmetric definite matrix; the upper leftmost block $-\mu(\Delta\mathbf{I}+\boldsymbol{\nabla}\boldsymbol{\nabla}^T)$ *is* symmetric positive definite but the discrete form of the off-diagonal blocks ($\mu\boldsymbol{\nabla}$ and $\mu\boldsymbol{\nabla}^T$) is actually *skew-symmetric* as the discrete first order operators satisfy $D_i^T = -D_i$. Also, negating the pressure equation to make the system symmetric, would still render it indefinite. Secondly, in the incompressible limit the diagonal constant term μ^2/λ vanishes, so neither Gauss-Seidel nor Jacobi methods would be usable.

Apart from these technical difficulties, it is generally known that for a differential equation such as (6) exhibiting nontrivial coupling between the variables ϕ_1, ϕ_2, ϕ_3 and p, a smoothing scheme which updates several variables at once is often the optimal choice in terms of efficiency [Trottenberg et al. 2001]. We note that this is not the same as a more costly block smoother where a larger number of *equations* are solved simulatenously; we still process one equation at a time, but the residual is eliminated by changing the value of several variables, rather than just one. We adopt the *distributive smoothing* scheme introduced by [Gaspar et al. 2008] for linear elasticity, which we later extend to nonlinear problems. Let us redefine $\mathcal{L}$ to denote the augmented differential operator of equation (6), and write $\boldsymbol{u} = (\boldsymbol{\phi}, p)$ for the augmented set of unknowns and $\boldsymbol{b} = (\boldsymbol{f}, 0)$ for the right-hand side vector. Thus, system (6) is written as $\mathcal{L}\boldsymbol{u} = \boldsymbol{b}$. Consider the following change of variables

$$\begin{pmatrix} \boldsymbol{\phi} \\ p \end{pmatrix} = \begin{pmatrix} \mathbf{I} & -\boldsymbol{\nabla} \\ \boldsymbol{\nabla}^T & -2\Delta \end{pmatrix} \begin{pmatrix} \boldsymbol{\psi} \\ q \end{pmatrix} \quad \text{or} \quad \boldsymbol{v} = \mathcal{M}\boldsymbol{u} \tag{7}$$

where $\boldsymbol{v} = (\boldsymbol{\psi}, q)$ is the vector of auxiliary unknown variables, and $\mathcal{M}$ is called the *distribution* matrix. In accordance with our staggered formulation, the components ψ_1, ψ_2, ψ_3 of the auxiliary vector $\boldsymbol{\psi}$ will be collocated with ϕ_1, ϕ_2, ϕ_3 respectively, while q and p values are collocated as well. Using the change of variables of equation (7), our augmented system $\mathcal{L}\boldsymbol{u} = \boldsymbol{b}$ is equivalently written as $\mathcal{L}\mathcal{M}\boldsymbol{v} = \boldsymbol{b}$. Composing the operators $\mathcal{L}$ and $\mathcal{M}$ yields

$$\mathcal{L}\mathcal{M} = \begin{pmatrix} -\mu\Delta\mathbf{I} & 0 \\ \mu(1+\frac{\mu}{\lambda})\boldsymbol{\nabla}^T & -\mu(1+\frac{2\mu}{\lambda})\Delta \end{pmatrix} \tag{8}$$

That is, the composed system is lower triangular, and its diagonal elements are simply Laplacian operators. This system can be smoothed with any scheme that works for the Poisson equation, including the Gauss-Seidel or Jacobi methods. In fact, the entire system can be smoothed practically with the same efficiency as the Poisson equation, following a forward substitution approach, i.e. we smooth all ψ_1-centered equations across the domain first, followed by sweeps of ψ_2, ψ_3, and q-centered equations in sequence.

One seeming obstacle to realizing these benefits, is that we do not have the auxiliary variables $(\boldsymbol{\psi}, q)$ at our disposal. As a matter of fact, computing $(\boldsymbol{\psi}, q)$ from $(\boldsymbol{\phi}, p)$ would necessitate solving system (7). Fortunately, such an explicit transformation is not necessary. We start by reviewing the standard Gauss-Seidel iteration for solving (or smoothing) the system $\mathcal{L}\boldsymbol{u} = \boldsymbol{b}$. At every step of the iteration, we focus on a different equation $\mathcal{L}_i$ (here i indicates a single discrete equation, as opposed to the three coordinate components of $\mathcal{L}$). Each Gauss-Seidel step amounts to calculating a pointwise correction to the variable u_i collocated with the equation $\mathcal{L}_i$, such that the residual of $\mathcal{L}_i$ vanishes. In more detail, we seek to replace variable u_i with $u_i + \delta$, or equivalently $\boldsymbol{u}$ with $\boldsymbol{u} + \delta\boldsymbol{e}_i$. As a result of this value change, the residual of the equation becomes $\boldsymbol{r} = \boldsymbol{b} - \mathcal{L}(\boldsymbol{u} + \delta\boldsymbol{e}_i)$. The unknown variable δ is determined by requiring that $r_i = \boldsymbol{e}_i^T\boldsymbol{r}$ becomes zero after this correction, thus

$$\boldsymbol{e}_i^T(\boldsymbol{b} - \mathcal{L}(\boldsymbol{u} + \delta\boldsymbol{e}_i)) = 0 \Rightarrow (\boldsymbol{e}_i^T\mathcal{L}\boldsymbol{e}_i)\delta = \boldsymbol{e}_i^T(\boldsymbol{b} - \mathcal{L}\boldsymbol{u})$$

The last equation is equivalent to $\mathcal{L}_{ii}\delta = r_i^{\text{old}}$ or $\delta = r_i^{\text{old}}/\mathcal{L}_{ii}$, where $\mathcal{L}_{ii}$ is the i-th diagonal element of the discrete operator and r_i^{old} denotes the i-th component of the residual vector *before* the correction. Operating in an analogous fashion, a Gauss-Seidel step on the *distributed* system $\mathcal{L}\mathcal{M}\boldsymbol{v} = \boldsymbol{b}$ amounts to changing ψ_i into $\psi_i + \delta$, or equivalently $\boldsymbol{v}$ into $\boldsymbol{v} + \delta\boldsymbol{e}_i$, such that the i-th residual of the distributed equation is annihilated, as follows

$$\boldsymbol{e}_i^T(\boldsymbol{b} - \mathcal{L}\mathcal{M}(\boldsymbol{v} + \delta\boldsymbol{e}_i)) = 0 \Rightarrow \boldsymbol{e}_i^T(\boldsymbol{b} - \mathcal{L}(\boldsymbol{u} + \delta\mathcal{M}\boldsymbol{e}_i)) = 0$$
$$\Rightarrow (\boldsymbol{e}_i^T\mathcal{L}\mathcal{M}\boldsymbol{e}_i)\delta = \boldsymbol{e}_i^T(\boldsymbol{b} - \mathcal{L}\boldsymbol{u}) \Rightarrow \delta = r_i^{\text{old}}/(\mathcal{L}\mathcal{M})_{ii}$$

In this derivation we leveraged the fact that the auxiliary vector $\boldsymbol{v}$ is only used in the form $\mathcal{M}\boldsymbol{v}$ which is equal to the value of the original variable $\boldsymbol{u}$. After the value of δ has been determined, $\boldsymbol{u}$ can be updated to $\boldsymbol{u} + \delta\mathcal{M}\boldsymbol{e}_i$. This update can be put in a more convenient form by observing that the discrete operator $\mathcal{M}$ is symmetric (discrete first-order derivative operators are skew-symmetric), thus $\mathcal{M}\boldsymbol{e}_i$ can be taken form either the i-th column *or* row of the discretized operator. Therefore, the correction $\boldsymbol{u} \leftarrow \boldsymbol{u} + \delta\mathcal{M}\boldsymbol{e}_i$ is

simply implemented by adding a δ-scaled version of the finite difference stencil of the i-th row of $\mathcal{M}$ to the current solution $\boldsymbol{u}$. Finally, the diagonal element $\boldsymbol{e}_i^T \mathcal{L}\mathcal{M}\boldsymbol{e}_i$ can be precomputed as the inner product of the stencils for the i-th row of $\mathcal{L}$ and the i-th row of $\mathcal{M}$ (again, due to symmetry). The computational cost of distributive smoothing is comparable to that of simple Gauss-Seidel iteration, yet it allows efficient smoothing of the equations of linear elasticity, independent of Poisson's ratio. We summarize the distributive smoothing process in pseudo-code in Table 2.

Table 2 Distributive Smoothing

```
1:  procedure DistributiveSmoothing(L,M,u,b)
2:      for v in {φ1, φ2, φ3, p} do      ▷ Must be in this exact order
3:          for i in Lattice[v] do        ▷ i is an equation index
4:              DistributiveSmoothingStep(L,M,u,b,i)
5:          end for
6:      end for
7:  end procedure
8:  procedure DistributiveSmoothingStep(L,M,u,b,i)
9:      r ← b_i − L_i · u                 ▷ L_i is the i-th row of L
10:     δ ← r/(LM)_ii
11:     u += δ m_i^T                      ▷ m_i is the i-th row of M
12: end procedure
```

5 Treatment of boundaries

The discretization and smoothing procedures discussed in the previous sections did not address the effect of boundaries, focusing on the treatment of the interior region of the simulated deformable body. In fact, we have been able to evaluate the validity and efficiency of the preceding formulations using a periodic domain, which is devoid of any boundaries. Our findings are reported in section 12 and can serve as a theoretical upper bound of multigrid efficiency. Multigrid theory suggests that a general boundary value problem can be solved at the same efficiency as a periodic problem, at the expense of more intensive smoothing effort at the boundary. In theoretical studies of multigrid efficiency, the computational overhead of this additional boundary smoothing is often overlooked, as the cost of interior smoothing is *asymptotically* expected to dominate. Nevertheless, in our experience, practical problem sizes may never reach this asymptotic regime, and generic boundary smoothing approaches could become a performance bottleneck, even for problems with millions of degrees of freedom. In this section, we develop a boundary discretization strategy, including a novel treatment of traction boundary conditions, that facilitates the design of efficient and inexpensive boundary smoothers.

5.1 Domain description

Our geometrical description of the computational domain is based on a partitioning of the cells of the background grid. Initially, cells that have an overlap with the simulated deformable body are characterized as *interior* cells, otherwise they are designated *exterior* cells. Additionally, any cell can be user-specified to be a *constrained* (or *Dirichlet*) cell. Specifying a Dirichlet cell overrides any interior/exterior designation it may otherwise carry. Geometrically, the interface between interior and Dirichlet cells corresponds to the boundary of the computational domain where Dirichlet boundary conditions are given, while traction (or free) boundary conditions are imposed on the interface between interior and exterior cells. Intuitively, Dirichlet cells correspond to kinematically constrained parts of the object, such as the skeleton of an articulated character. This partitioning of the domain is illustrated in Figure 5.

This definition provides an intuitive way to specify the degrees of

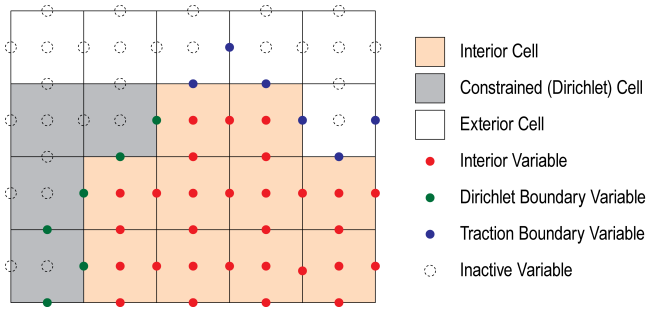

Figure 5: *Classification of cells and variables near the boundary*

freedom of our problem and their associated equations. Any of the variables ϕ_1, ϕ_2, ϕ_3 or p located strictly inside the interior region (i.e. either on a interor cell center, or on the face between two interior cells) is designated an *interior* variable. For every interior variable we also introduce in our system the discrete equation of (6) centered at the same location. Any variable which is included in the discrete stencil of an interior equation, but is not interior itself, is a *boundary* variable; such a variable will be further designated a Dirichlet boundary variable if it touches a Dirichlet cell (either inside or on the boundary of one), otherwise it is designated a traction boundary variable. Variables not appearing in the stencil of any interior equation are labeled *inactive* and can generally be ignored.

5.2 A general-purpose box smoother

We first describe a general-purpose treatment of the boundary equations and variables. In order to allow for a unique solution to our discrete problem, one additional equation must be provided for each boundary variable. Note that, as a result of our domain description, there will be no boundary *pressure* variables, i.e. no exterior pressure ever appears on the discrete stencil of an interior equation. For every Dirichlet boundary variable ϕ_i we need to specify a Dirichlet condition of the form $\phi_i(\boldsymbol{X}^*) = \phi_i^*$, while every traction boundary variable ϕ_i will be naturally matched with a traction condition $\boldsymbol{e}_i^T \mathbf{P}(\boldsymbol{X}^*)\boldsymbol{N} = t^*$, where $\boldsymbol{N}$ is the normal vector to the object surface (t^*=0 would be the zero-traction or *free* boundary condition). The point $\boldsymbol{X}^*$ used in these boundary equations need not coincide with the location of the boundary variables, allowing the flexibility to use a boundary condition defined on the object surface even when the associated boundary variable is offset from it due to staggering. As a consequence, averaging (for Dirichlet equations) or one-sided finite differences (for traction equations) may be needed in the discrete formulation of these boundary equations. In general, any first-order accurate (or better) finite difference approximation for either type of boundary condition is acceptable for our purposes.

Although a well-posed system can be constructed as described, the distributive smoothing scheme cannot be used in the immediate vicinity of the boundary. In that region, the distributive corection for certain variables extends outside the domain, affecting boundary and even inactive variables. Moreover, the boundary equations themselves need to be smoothed, and that cannot be accomplished by simply substituting a different smoother for the specific equations where distributive smoothing is not applicable. In such situations a *box smoother* is a broadly applicable solution. This process amounts to collectively solving a number of equations in a rectangular box, simultaneously adjusting the values of all variables within that region. More formally, we compute a collective correction $\boldsymbol{\delta} = (\delta_{i_1},..,\delta_{i_N})$ such that the N equations $i_1,..,i_N$ will be simultaneously satisfied after the correction has been applied. The correction vector can be obtained as the solution of the equation

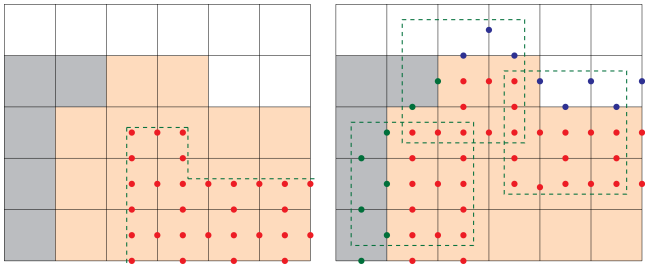

Figure 6: *Left: Extent of distributive smoothing (interior region), Right: Boundary region with some boxes used by the box smoother.*

$\mathcal{L}^*\boldsymbol{\delta} = \boldsymbol{r}^*$, where $\mathcal{L}^*$ is the $N\times N$ submatrix of $\mathcal{L}$ corresponding to the rows and columns indexed i_1 through i_N, and $\boldsymbol{r}^*$ contains the corresponding N entries of the residual vector before the correction. Our complete smoothing subroutine starts with a boundary box smoothing sweep, proceeds with a sweep of interior distributive smoothing and finishes with a last boundary pass. The criterion for performing distributive smoothing on a certain interior equation is that the discrete stencil of the corresponding *distributed* equation, i.e. the i-th equation of the composed system $(\mathcal{L}\mathcal{M})\boldsymbol{v} = \boldsymbol{b}$ needs to contain only interior variables, as illustrated in Figure 6 (left). During the boundary sweep, we collectively solve all equations in overlapping boxes that are two grid cells wide, and centered at the centers of all the outermost layer of interior cells, as seen in Figure 6 (right). The local system $\mathcal{L}^*\boldsymbol{\delta} = \boldsymbol{r}^*$ can be pre-factorized using a pivoted LU decomposition and solved by means of forward and back substitution. In our experiements the box smoother performed very well, generally allowing the entire multigrid scheme to converge at the interior efficiency (i.e. achieving the convergence rates observed for a periodic problem).

5.3 A fast symmetric Gauss-Seidel smoother

Although the box smoother described in section 5.2 was effective in achieving a good convergence rate, the runtime overhead associated with it was enormous. For a problem with 32K vertices, the execution time of boundary smoothing was about 60 – 80 times longer than that of the interior distributive smoothing. Asymptotically, the cost of boundary smoothing is $O(N^{2/3})$, while the distributive smoothing has an asymptotic $O(N)$ complexity, where N is the number of vertices in the simulation. These asymptotics, however, will not materialize into a tangible befit for realistic problem sizes. Even for models with 2M vertices, boundary smoothing would still require more than 15 times the runtime of interior smoothing. Furthermore, when using a direct solver for box smoothing in a parallel machine we would waste substantial memory bandwidth to streaming the precomputed LU factors, and tolerating on-the-fly factorization would likely be an even costlier alternative.

We propose a novel formulation that enables equation-by-equation smoothing that is both efficient and inexpensive. The main obstacle to designing efficient equation-by-equation boundary smoothing schemes, is that the discrete system of equations near the boundary lacks properties such as symmetry, definiteness or diagonal dominance. In particular, loss of definiteness is predominantly a side-effect of the augmented discretization (6) the distributive smoother owes its efficiency to. Additionally, even natural discretizations of the boundary conditions (especially for the traction boundary) can easily result in loss of symmetry, even in our non-augmented system (4). An alternative local smoother would be the Kaczmarz method [Trottenberg et al. 2001], which does not require symmetry or definiteness of the boundary system; we have nevertheless found the Kaczmarz smoother to converge extremely slowly (a fact well documented in the literature) and consequently requiring a very large number of iterations, making it a very sub-optimal solution. Our proposed solution stems from a novel perspective of the constitutive equations *and* the boundary conditions that results in a well conditioned, symmetric positive definite boundary system.

First, we revisit the constitutive equation of linear elasticity (2). The scalar coefficient $\mathrm{tr}(\boldsymbol{\epsilon})$ appearing in equation (3) is equivalently written as $\mathrm{tr}(\boldsymbol{\epsilon})=\sum_i \epsilon_{ii}=\sum_i \phi_{i,i} - d$, where $d=\mathrm{tr}(\mathbf{I})$ equals the number of spatial dimensions. Similarly, the last equation of system (6) is equivalent to $-(\mu/\lambda)p = \boldsymbol{\nabla}^T\boldsymbol{\phi} = \sum_i \phi_{i,i}$. Thus, we have $\mathrm{tr}(\boldsymbol{\epsilon}) = -(\mu/\lambda)p - d$, and equation (2) becomes

$$\mathbf{P} = \mu(\mathbf{F} + \mathbf{F}^T) - \mu p\mathbf{I} - (2\mu + d\lambda)\mathbf{I} \qquad (9)$$

The difference between equations (2) and (9) is that the original definition of stress is physically valid for any given deformation field $\boldsymbol{\phi}$ while the formulation of equation (9) will correspond to the real value of stress only when the augmented system (6) is solved exactly. We can verify that the *position* equations $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3$ of system (9) are equivalent to the divergence form $\mathcal{L}_i\boldsymbol{u}=-\partial_j P_{ij}$, where $\mathbf{P}$ is now given by the new definition of equation (9). The discrete stencils for these equations can be constructed as a two-step process. First, we construct a finite difference stencil for the expression $-\partial_j P_{ij}$, treating every value P_{ij} appearing in this stencil as a separate variable (see Figure 7, left). As a second step, finite difference approximations are substituted in place of the P_{ij} values.

For interior equations, this process yeilds exactly the same results as the direct discretization of system (6). Certain interior equations near the boundary, however, are special in the sense that their stencil extends onto *boundary* variables. For those equations, we turn our attention to the stress variables appearing in their discrete divergence form. Such a stress variable P_{ij} will be characterized as *interior* if the discrete stencil for P_{ij} uses only interior or Dirichlet variables, and *exterior* if its stencil uses at least one traction boundary variable (see Figure 8, left). In our proposed formulation, any exterior stresses will not be evaluated by means of a finite difference stencil; instead a specific value will be substituted for them, using an appropriate traction boundary condition. More specifically:

- Stress variables of the form P_{ij} $(i \neq j)$ are centered on grid edges in 3D (see Figure 7, right) and on grid nodes in 2D. This stress variable appeared in the finite difference approximation of the term $-\partial_j P_{ij}$ in equation $\mathcal{L}_i$. Let $\boldsymbol{X}^*$ be the location where equation $\mathcal{L}_i$ is centered. The stress variable P_{ij} is located one half of a grid cell away from $\boldsymbol{X}^*$, *along the direction* $\boldsymbol{e}_j$. Without loss of generality, assume P_{ij} is located at $\boldsymbol{X}^*+\frac{h_j}{2}\boldsymbol{e}_j$. P_{ij} neighbors exactly four cells; out of those, the two centered at $\boldsymbol{X}^* \pm \frac{h_i}{2}\boldsymbol{e}_i$ are *interior* cells, since we assumed that $\mathcal{L}_i$ was an interior equation. The two other neighbor cells of P_{ij} are centered at $\boldsymbol{X}^* \pm \frac{h_i}{2}\boldsymbol{e}_i + h_j\boldsymbol{e}_j$. We

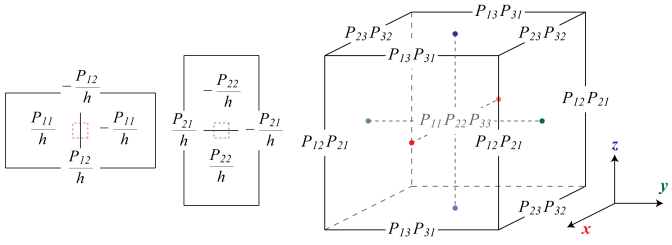

Figure 7: *Left: Equations $\mathcal{L}_1$,$\mathcal{L}_2$ expressed as divergence stencils. Right: Placement of the components of stress tensor $\mathbf{P}$ in 3D.*

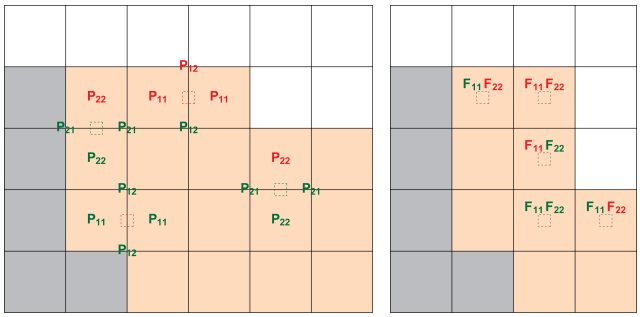

Figure 8: *Left: Stress variables used in the divergence form of certain interior equations. Boundary stress variables are colored red, interior stresses are green. All boundary stresses can be set to a specific value using a traction condition from a nearby boundary. Right: Interior and boundary gradients used in pressure equations.*

can verify that if those two cells were interior or Dirichlet, P_{ij} would have been an *interior* stress, i.e. its finite different stencil would have only included interior or Dirichlet variables. Thus, if P_{ij} is an exterior stress, one of the cells centered at $\boldsymbol{X}^* \pm \frac{h_i}{2}\boldsymbol{e}_i + h_j\boldsymbol{e}_j$ must be exterior. This means that P_{ij} is incident on a traction boundary face perpendicular to the direction $\boldsymbol{e}_j$, thus the traction condition $\mathbf{P}\boldsymbol{e}_j = \boldsymbol{t}$ associated with this part of the boundary specifies a value $P_{ij} = t_i$ for this component of the stress. In the common case of a free boundary the traction value is simply zero, giving $P_{ij} = 0$.

- Stress variables of the form P_{ii} are located at cell centers, and appear in the finite difference approximation of $-\partial_i P_{ii}$ in equation $\mathcal{L}_i$. Similar to the previous case, P_{ii} is located one half grid away from the location $\boldsymbol{X}^*$ of $\mathcal{L}_i$ *along the direction* $\boldsymbol{e}_i$. Without loss of generality, assume P_{ij} is located at $\boldsymbol{X}^* + \frac{h_i}{2}\boldsymbol{e}_i$. From (9) we have $P_{ii} = 2\mu\partial_i\phi_i - \mu p - (2\lambda + d\mu)$, thus the stencil for P_{ii} uses variables $\phi_i(\boldsymbol{X}^*), p(\boldsymbol{X}^* + \frac{h_i}{2}\boldsymbol{e}_i)$ which are both *interior* (since $\mathcal{L}_i$ is interior) and the one additional variable $\phi_i(\boldsymbol{X}^* + h_i\boldsymbol{e}_i)$. P_{ii} would be an exterior stress only if $\phi_i(\boldsymbol{X}^* + h_i\boldsymbol{e}_i)$ was an exterior variable; in this case P_{ii} would have been "near" (specifically half a cell away from) a traction boundary face normal to $\boldsymbol{e}_i$. Once again, we will use the traction condition associated with this boundary to set $P_{ii} = t_i$ (or $P_{ii} = 0$ for a free boundary). The subtlety of this formulation is that the stress variable P_{ii} is not located exactly on the boundary; nevertheless the discrete stencil for P_{ii} is still a valid *first-order* approximation of the continuous quantity P_{ii} *at* the boundary. At the worst case, our solver merely loses second-order accuracy, which is a widely acceptable compromise for computer graphics purposes, and creates hardly any visible artifacts in our experience.

In summary, we have justified that all *exterior* stress variables can be eliminated (and replaced with known constants) from the divergence form of interior position equations. A similar treatment is performed on the discretiztion of the pressure equation $\mathcal{L}_p = \mu \sum_i F_{ii} + \frac{\mu^2}{\lambda} p$. Similar to stresses, the deformation gradients F_{ii} are also characterized as interior or exterior, based on whether they touch traction boundary variables. Since $P_{ii} = 2\mu F_{ii} - \mu p - (2\lambda + d\mu)$, we observe that F_{ii} is exterior if and only if the stress P_{ii} is exterior (see figure 8, right). For such exterior gradients or stresses we can use the traction condition $P_{ii} = t_i$ to eliminate F_{ii} from the pressure equation. This is accomplished by replacing $\mathcal{L}_p \leftarrow \mathcal{L}_p - \frac{1}{2}(P_{ii} - t_i)$ for every exterior gradient F_{ii}. If more than one gradients are exterior, we can annihilate all of them from the stencil for $\mathcal{L}_p$ in a similar fashion. Note that in the process of annihilating these exterior gradients, we modify the original coefficient μ^2/λ of the pressure variable at the center of $\mathcal{L}_p$, yet the modified coefficient will retain a positive sign regardless of how many gradients are eliminated.

Our manipulations effectively remove all traction boundary variables from the discretization of the interior equations. For every Dirichlet boundary variable, we assume a Dirichlet condition of the form $\phi_i = c_i$ is provided. Thus, we can substitute a given value for every Dirichlet variable in the stencil of every interior equation that uses it. As a result, our overall discrete system can be written as $\mathcal{L}^*\boldsymbol{u}^* = \boldsymbol{b} - \boldsymbol{b}^D = \boldsymbol{b}^*$, where $\boldsymbol{u}^*$ only contains interior variables, and $\boldsymbol{b}^D$ results from moving the known Dirichlet variables to the right-hand side. The discrete system matrix $\mathcal{L}^*$ has as many rows and columns as interior variables, and will differ from $\mathcal{L}$ near the boundaries, as it incorporates the effect of the boundary conditions. An analysis of our formulation can verify that $\mathcal{L}^*$ has the form

$$\mathcal{L}^* = \begin{pmatrix} \mathbf{L}_\phi & \mathbf{G} \\ -\mathbf{G}^T & \mathbf{D}_p \end{pmatrix}$$

In this formulation $\mathbf{L}_\phi$ is symmetric, positive definite, and $\mathbf{D}_p$ is a diagonal matrix with positive diagonal elements. The skew-symmetry of the off-diagonal blocks is anticipated, since $\mathbf{G}$ originates from a discretization of first-order derivatives. As a final step, we define the *substitution* matrix $\mathcal{U}$

$$\mathcal{U} = \begin{pmatrix} \mathbf{I} & -\mathbf{G}\mathbf{D}_p^{-1} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}$$

and use it to pre-multiply our equation as

$$\mathcal{U}\mathcal{L}^*\boldsymbol{u}^* = \begin{pmatrix} \mathbf{L}_\phi + \mathbf{G}\mathbf{D}_p^{-1}\mathbf{G}^T & \mathbf{0} \\ -\mathbf{G}^T & \mathbf{D}_p \end{pmatrix} \boldsymbol{u}^* = \mathcal{U}\boldsymbol{b}^* \qquad (10)$$

Equation (10) is the basis of our boundary smoother. The top left block $\mathbf{L}_\phi + \mathbf{G}\mathbf{D}_p^{-1}\mathbf{G}^T$ is a symmetric and positive definite matrix and can be smoothed via Gauss-Seidel iteration. One may recognize that this matrix is qualitatively similar to the discretization of our non-augmented system (4) and, in fact, the two are identical away from boundaries. In section 4.1 we discussed the problematic conditioning of this matrix in the near-incompressible regime. In our boundary smoother, however, this formulation is only used for a very narrow region around the boundary, specifically the same interior equations affected by our previous box smoother formulation (depicted as red interior nodes in Figure 6, right). The boundary and interior regions are smoothed in separate sweeps; during the sweep of the boundary smoother, all interior variables not being smoothed are effectively treated as Dirichlet values. The fact that the boundary smoother is confined in a narrow region between boundary conditions has a strong stabilizing effect, as compared to its use for interior smoothing. In practice, we found that 2 Gauss-Seidel boundary sweeps for every sweep of the distributive interior smoother are sufficient for Poisson's ratio up to $\nu = .45$, while 3-4 Gauss-Seidel sweeps will accommodate values as high as $\nu = .495$.

The last step in the boundary smoothing process is the treatment of the pressure and boundary variables, which were not included in the Gauss-Seidel update. Since the lower right block of equation (10) is diagonal, all pressure equations can be satisfied exactly via a simple Gauss-Seidel sweep (which is essentially equivalent to forward-substitution). Note that this update of pressures only needs to occur once *after* the position variables have been smoothed by a given number of sweeps. Lastly, the boundary traction variables can be updated using the traction conditions $P_{ij} = t_i$ that have been imposed. For simplicity we will assume that no interior cell

is trapped between two exterior cells which are adjancent on two opposing faces, i.e. the active domain does not contain a one-cell wide isthmus between traction boundaries. This assumption is only required of the finest level of discretization in the multigrid scheme and can be procedurally accomplished by minimal thickening of the active domain. In this case all external diagonal stress components P_{ii} have a stencil that touches only a single boundary variable, located at the surface of the active domain. Thus, if the equation $P_{ii} = t_i$ is stored at the location of this specific boundary variable, a Gauss-Seidel step will compute the exact value of that boundary variable. Similarly, all off-diagonal stress components P_{ij} $(i \neq j)$ will contain only a single boundary variable, located *outside* the active domain, whose value has not been already determined, and we store the boundary condition $P_{ij} = t_i$ on the location of that variable. In summary, after the smoothing of interior position variables, all our smoother has to do to is perform Gauss-Seidel relaxation on pressure equations, diagonal stress equations $P_{ii} = t_i$ and off-diagonal stress equations $P_{ij} = t_i$, in this specific order. At the end of the process, all boundary equations will be satisfied *exactly* (i.e. they will have zero residuals), which we exploit next.

6 Construction of the Transfer operators

We designed the Restriction ($\mathcal{R}$) and Prolongation ($\mathcal{P}$) operators employed by the algorithm of Table 1 aiming to keep implementation as inexpensive as possible, while conforming to the textbook accuracy requirements for full multigrid efficiency (see [Trottenberg et al. 2001]). We define the following 1D averaging operators

$$\mathcal{B}^1 u[x] = \tfrac{1}{2}u[x-\tfrac{h}{2}] + \tfrac{1}{2}u[x+\tfrac{h}{2}]$$
$$\mathcal{B}^2 u[x] = \tfrac{1}{4}u[x-h] + \tfrac{1}{2}u[x] + \tfrac{1}{4}u[x+h]$$
$$\mathcal{B}^3 u[x] = \tfrac{1}{8}u[x-\tfrac{3h}{2}] + \tfrac{3}{8}u[x+\tfrac{h}{2}] + \tfrac{3}{8}u[x-\tfrac{h}{2}] + \tfrac{1}{8}u[x+\tfrac{3h}{2}]$$

The restriction and prolongation operators will be defined as tensor product stencils of the preceeding 1D operators as

$$\mathcal{R}_1 = \mathcal{B}^2 \otimes \mathcal{B}^1 \otimes \mathcal{B}^1 \qquad \mathcal{P}_i^T = 8\,\mathcal{B}^2 \otimes \mathcal{B}^3 \otimes \mathcal{B}^3$$
$$\mathcal{R}_2 = \mathcal{B}^1 \otimes \mathcal{B}^2 \otimes \mathcal{B}^1 \qquad \mathcal{P}_2^T = 8\,\mathcal{B}^3 \otimes \mathcal{B}^2 \otimes \mathcal{B}^3$$
$$\mathcal{R}_3 = \mathcal{B}^1 \otimes \mathcal{B}^1 \otimes \mathcal{B}^2 \qquad \mathcal{P}_3^T = 8\,\mathcal{B}^3 \otimes \mathcal{B}^3 \otimes \mathcal{B}^2$$
$$\mathcal{R}_p = \mathcal{B}^1 \otimes \mathcal{B}^1 \otimes \mathcal{B}^1 \qquad \mathcal{P}_p^T = 8\,\mathcal{B}^1 \otimes \mathcal{B}^1 \otimes \mathcal{B}^1$$

where $\mathcal{R}_i$,$\mathcal{P}_i$ are the restriction and prolongation operators used for variable $\boldsymbol{u}_i$, respectively. We opted to define the prolongation operator in terms of its transpose, since $\mathcal{P}_i^T$ has the same stencil everywhere, while $\mathcal{P}_i$ is composed of several different stencils. We can easily verify that $\mathcal{P}$ simply corresponds to trilinear interpolation.

Our domain description for the finest grid was based on a partitioning of the cells into interior, exterior and Dirichlet. The coarse background grid is derived by the natural 8-to-1 coarsening of the cartesian background lattice. Furthermore, a coarse cell is designated a Dirichlet cell if *any* of its eight fine sub-cells is Dirichlet. If any of the fine sub-cells are interior and none is Dirichlet, the coarse cell will be considered interior. Otherwise, the coarse cell is exterior. Thus, the coarse active domain is geometrically a superset of the fine domain, while its Dirichlet parts are more extensive. Despite this geometrical discrepancy, which is in any case no larger than the grid size, we were still able to retain the interior efficiency of the multigrid scheme, by taking a few simple additional steps.

In our treatment of boundary equations in section 5.3 we effectively forced all boundary conditions to be satisfied *exactly* after every application of the smoother, by absorbing boundary equations into the interior system. In general, if a smoother by design leaves a residual

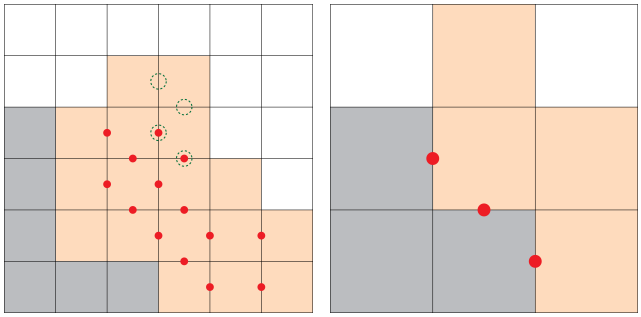

Figure 9: *Fine grid domain description (left) and its coarse grid form (right). Red dots indicate (a) coarse Dirichlet equations that were interior in the fine grid, and (b) the fine grid equations they restrict residuals from. Green circles indicate fine interior variables that prolongate their correction from boundary coarse variables.*

on the boundary equations, this residual has to be restricted (separately from interior residuals) onto the boundary equations of the fine grid. In our case all boundary residuals in the fine grid are zero, thus all coarse boundary equations will be *homogeneous*; for Dirichlet equations they will have the form $\boldsymbol{u}_i^{2h} = 0$ (i.e. the coarse grid incurs no correction), while traction equations will be of the form $\hat{P}_{ij}^{2h} = 0$, where $\hat{\mathbf{P}}$ is the homogeneous part of $\mathbf{P}$ (i.e. we omit any constant terms). We also note that, due to the possible geometrical change of the Dirichlet region, certain coarse Dirichlet equations will be centered on locations that were interior in the fine grid (shown as red dots in Figure 9, right). The fine grid interior equations (red dots in Figure 9, left) that would restrict their residuals onto these (now Dirichlet) coarse locations, will not have their residuals well represented on the coarse grid. We compensate for this inaccuracy by performing an extra 2-3 sweeps of our boundary Gauss-Seidel smoother over these equations, driving their residuals very close to zero, just before the Restriction operation takes place.

In section 5.3 we stated that the interior domain must not have a one-cell wide isthmus, where an interior cell is neighbored by two exterior cells across two opposing faces, in order for the traction boundary variables to be exactly computable using a sequence of Gauss-Seidel updates. Although this property can be procedurally enforced in the fine grid, the coarse grid could easily violate it, as illustrated in Figure 9. Instead of attempting to compute these coarse boundary variables in some more elaborate way, we simply elect to not use them. This is possible, since the formulation of section 5.3 defines a form of the equations that completely excludes the traction boundary variables. The consequence of this approach is that certain fine grid variables (depicted as green circles in Figure 9, left) cannot get an accurate correction, since it would have to be prolongated from an omitted coarse boundary variable. Again, we found that performing 2-3 Gauss-Seidel smoother sweeps over these equations immediately after the prolongation of the correction will be enough to compensate for this inaccurate correction.

7 Co-rotational linear elasticity

Our discussion so far was based on the constitutive equations of linear elasticity. This model allowed us to detail the theoretical and engineering subtleties associated with designing a highly efficient multigrid scheme, and provides excellent conditions for the implementation of real-time multigrid solvers for models with hundreds of thousands of degrees of freedom. The physical validity, however, of linear elasticity is primarily limited to scenarios of moderate deformation. In the large deformation regime, and in the presence

of large rotational deformations in particular, the linear elasticity model develops artifacts such as volumetric distortions in parts of the domain with large rotations. Interestingly, our experiments with linear elasticity seemed to indicate that visually plausible deformations were attainable even for certain examples with higher deformation, where conventional wisdom would lead us to expect more visible artifacts; we traced this effect to our ability to use a high Poisson's ratio which helped with volume conservation in the moderate deformation regime. Nevertheless, it is easy to demonstrate cases where linear elasticity produces visibly nonphysical results, and where a nonlinear treatment of deformation is required.

Our first extension is the co-rotational linear elasticity model, which has been used in slightly different forms by a number of authors in computer graphics [Müller et al. 2002; Hauth and Strasser 2004; Müller and Gross 2004], and has also been employed in the context of a finite element based multigrid framework by [Georgii and Westermann 2006]. The co-rotational formulation extracts the rotational component of the local deformation at a specific part of the domain by computing the polar decomposition of the deformation gradient tensor $\mathbf{F} = \mathbf{RS}$ into the rotation $\mathbf{R}$ and the symmetric tensor $\mathbf{S}$. The stress is then computed as $\mathbf{P} = \mathbf{R}\mathbf{P}_L(\mathbf{S})$, where $\mathbf{P}_L$ denotes the stress of a *linear* material, as described in equation (2). Thus, the co-rotational formulation computes stress by applying the constitutive equation of linear elasticity in a frame of reference that is rotated with the material deformation, and subsequently rotating the result back to unrotated coordinates. After the necessary substitutions, the constitutive equation takes the following form

$$\begin{aligned} \mathbf{P} &= 2\mu(\mathbf{F}-\mathbf{R}) + \lambda \mathrm{tr}(\mathbf{R}^T\mathbf{F}-\mathbf{I})\mathbf{R} \\ &= 2\mu\mathbf{F} + \lambda \mathrm{tr}(\mathbf{R}^T\mathbf{F})\mathbf{R} - (2\mu + d\lambda)\mathbf{R} \\ &= 2\mu\mathbf{F} - \mu p\mathbf{R} - (2\mu + d\lambda)\mathbf{R} \end{aligned} \quad (11)$$

where the last form of the stress in equation (11) results from introducing an auxiliary pressure variable $p{=}-(\lambda/\mu)\mathrm{tr}(\mathbf{R}^T\mathbf{F})$ in a fashion similar to the augmentation used for the linear elasticity problem in section 4.1. As before, the augmented position equations are defined as $-\partial_j P_{ij}{=}f_i$. After combining with the equations defining the pressures and rearranging some terms we get

$$\begin{pmatrix} -2\mu\Delta\mathbf{I} & \mu(\boldsymbol{\nabla}^T\mathbf{R}^T)^T \\ \mu(\mathbf{R}\boldsymbol{\nabla})^T & \frac{\mu^2}{\lambda} \end{pmatrix} \begin{pmatrix} \boldsymbol{\phi} \\ p \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}+(2\mu+d\lambda)\boldsymbol{\nabla}\cdot\mathbf{R} \\ 0 \end{pmatrix} \quad (12)$$

which we use as the augmented PDE of co-rotational linear elasticity. The notation for the off-diagonal blocks of the matrix in equation (12) were used to indicate whether the operators $\boldsymbol{\nabla}, \boldsymbol{\nabla}^T$ operate or not on the rotation matrix $\mathbf{R}$. In index form, these operators equal $[\mu(\boldsymbol{\nabla}^T\mathbf{R}^T)^T]_i = \mu\partial_j R_{ij}$, and $[\mu(\mathbf{R}\boldsymbol{\nabla})^T]_i = \mu R_{ij}\partial_j$ respectively. *Discuss quasi-linear form here*

8 Neo-hookean elasticity

TBD

9 Dynamics

TBD

10 Collisions

TBD

11 Parallelization and Optimization

TBD

12 Results

TBD

13 Discussion and Limitations

TBD

14 Conclusion

TBD

References

BONET, J., AND WOOD, R. 1997. *Nonlinear Continuum Mechanics for Finite Element Analysis*. Cambridge University Press.

BREZZI, F., AND FORTIN, M. 1991. *Mixed and hybrid finite element methods*. Springer: Berlin.

ENGLISH, E., AND BRIDSON, R. 2007. Animating developable surfaces using nonconforming elements. *ACM Trans. Graph. (SIGGRAPH Proc.) 26*, 3.

FEDKIW, R., STAM, J., AND JENSEN, H. 2001. Visual simulation of smoke. In *Proc. of ACM SIGGRAPH 2001*, 15–22.

GASPAR, F., GRACIA, J., LISBONA, F., AND OOSTERLEE, C. 2008. Distributive smoothers in multigrid for problems with dominating grad-div operators. *Numerical Linear Algebra with Applications 15*, 8, 661–683.

GEORGII, J., AND WESTERMANN, R. 2006. A multigrid framework for real-time simulation of deformable bodies. *Computers and Graphics 30*, 3, 408 – 415.

HAUTH, M., AND STRASSER, W. 2004. Corotational Simulation of Deformable Solids. In *In Proc. of WSCG*, 137–145.

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

JAMES, D., BARBIC, J., AND TWIGG, C. 2004. Squashing cubes: Automating deformable model construction for graphics. In *SIGGRAPH 2004 Sketches & Applications*, ACM Press.

KAUFMANN, P., MARTIN, S., BOTSCH, M., AND GROSS, M. 2008. Flexible Simulation of Deformable Models Using Discontinuous Galerkin FEM. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.

KAZHDAN, M., AND HOPPE, H. 2008. Streaming Multigrid for Gradient-Domain Operations on Large Images. *ACM Transactions on Graphics 27*, 3.

MÜLLER, M., AND GROSS, M. 2004. Interactive virtual materials. In *Graph. Interface*, 239–246.

MÜLLER, M., DORSEY, J., MCMILLAN, L., JAGNOW, R., AND CUTLER, B. 2002. Stable real-time deformations. In *ACM SIGGRAPH Symp. on Comput. Anim.*, 49–54.

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

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

TERZOPOULOS, D., AND FLEISCHER, K. 1988. Deformable models. *The Visual Computer 4*, 6, 306–331.

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.