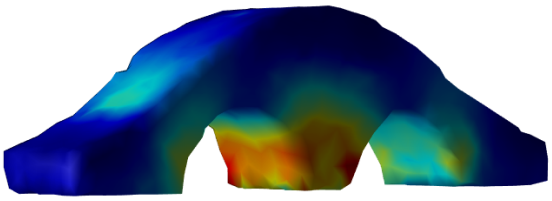
Linear Elasticity Simulation for Stress Visualization

Peiqi Wang, Eris Zhang

April 10, 2019



Contents

1 Introduction

In computational fabrication, it is crucial to quantify how a 3D printed objects might break or fail using linear elasticity simulations. [1] [2] [2] To enable interactive design and optimization of shapes such that failure cases are minimized, there has been work to increase the speed of simulation with novel interpolation algorithms [3] or with the help of modern parallel hardwares. [4]. A particularly interesting and relevant work uses matrix-free iterative methods to perform real time simulation on GPU. [5] [6]

The aim of this report is to explore potential methods to visualize failure cases, i.e. fractures, breakage, etc. for solid shapes of elastic material in real time. For our use case, precision of simulation is only important up to the visualization not deviating from the ground truth by human perception. To exploit this assumption, we explored different iterative methods to solve the linear system resulted from finite element discretization. We explored linear basis functions on both tetrahedron element and regular hexahedron element. In the latter case, we can avoid the costly operation of numerically computing integrals at each element, which opens up opportunity to use matrix-free iterative method with substantial reduction in memory and time cost.

2 Linear Elasticity

2.1 Background

Continuum mechanics deals with behavior of material modeled as a continuous mass. The study of elastic material is a subset of continuum mechanics that models the ability of a material to resist influence and return to its original configuration when the influence is removed. The following definitions are important to elasticity theory and provide important background for rest of the report. [7]

1. **(stress)** describes internal force over a differential area at a point, which can be represented as a symmetric second order tensor $\boldsymbol{\sigma}$,

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{bmatrix}$$

   where

$$\sigma_{ij} = \lim_{dA_i \to 0} \frac{F_j}{dA_i}$$

   represents force applied in j-th coordinate axis over a differential plane orthogonal to i-th coordinate axis.

2. **(strain)** describes deformation by relative displacements, which can be represented as a symmetric second order tensor $\boldsymbol{\varepsilon}$,

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_{11} & \varepsilon_{12} & \varepsilon_{13} \\ \varepsilon_{21} & \varepsilon_{22} & \varepsilon_{23} \\ \varepsilon_{31} & \varepsilon_{32} & \varepsilon_{33} \end{bmatrix}$$

   where ε_{ii} are axial strains and ε_{ij} where $i \neq j$ are shearing strains.

3. **(displacement)** describes how much a point moves from referenced to deformed configuration, which can be represented as component-wise displacement, $\mathbf{u}$

$$\mathbf{u} = (u_1, u_2, u_3)$$

4. **(yield)** a point on a stress-strain curve of a material that indicates the limit of elastic behavior. Material experiencing stress over the yield point will suffer from permanent deformations.

5. **(von Mises yielding criterion)** Ultimately, we are interested in computing the stress field over all points within a solid geometry, from which we can deduce material yielding by comparing the effective stress σ_e with material-specific yield stress σ_y. Simply, we use Von Mises' criterion to determine yielding,

$$\sigma_e = \frac{1}{\sqrt{2}}\sqrt{(\sigma_{11} - \sigma_{22})^2 + (\sigma_{22} - \sigma_{33})^2 + (\sigma_{33} - \sigma_{11})^2 + 6(\sigma_{23}^2 + \sigma_{31}^2 + \sigma_{12}^2)} > \sigma_y$$

2.2 Problem Assumptions

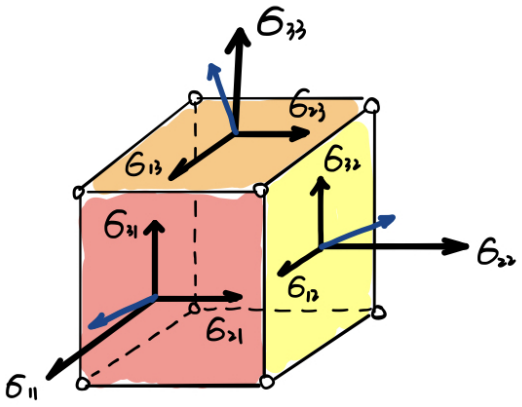


Figure 1: Strain tensor

Here, we lay down simplifying assumptions for our problem.

1. **(Analytic stress/strain)** If we assume that stress and strain field is analytic in small neighborhood of a point, we can guarantee convergent Taylor series expansion, which leads to simplier formulation of differential equation governing materials' elastic behavior. This assumption, for example, helped with derivation of Green-Lagrangian strain tensor

$$\varepsilon_{ij} = (u_{i,j} + u_{j,i} - u_{k,i}u_{k,j})$$

2. **(Small deformations)** Typically, elastic material exhibit a nonlinear relationship between stress and strain. However, if we assume for small deformation only, stress-strain is linear, hence the name *linear* elasticity. We can employ Hooke's law to model the relationship between stress and strain.

$$\boldsymbol{\sigma} = \mathbf{C} : \boldsymbol{\varepsilon}$$

where $\mathbf{C}$ is a fourth-order stiffness tensor.

3. **(Isotropic homogeneous material)** Typically, material exhibit direction or location specific behavior. If materials are characterized by properties independent of orientation and coordinate system, we can reduce the number of material-dependent constants to 2 (i.e. λ, μ),

$$\boldsymbol{\sigma} = \mathbf{C} : \boldsymbol{\varepsilon} \qquad \rightarrow \qquad \boldsymbol{\sigma} = 2\mu\boldsymbol{\varepsilon} + \lambda \operatorname{tr} \boldsymbol{\varepsilon}\mathbf{I}$$

Also note that the stiffness tensor is symmetric, i.e. $\mathbf{C}_{ijkl} = \mathbf{C}_{klij}$. This is true even for anisotropic materials.

4. **(Static equilibrium)** We are only interested in modeling static objects. With this assumption, we obtain a simpler formula from Newton's second law on conservation of linear and angular momentum.

$$\nabla \cdot \boldsymbol{\sigma} + \mathbf{f} = \rho\ddot{\mathbf{u}} \qquad \rightarrow \qquad \nabla \cdot \boldsymbol{\sigma} + \mathbf{f} = 0$$

where $\mathbf{f}$ are body forces, and ρ is density of the material.

2.3 Differential Equations

From physical laws and simplifying assumptions, we can derive a set of equations governing linear elasticity with respect to the stress field $\boldsymbol{\sigma}$, the strain field $\boldsymbol{\varepsilon}$, and the displacement field $\mathbf{u}$. We express the equations in tensor notations.

1. **(Equilibrium Equations)** express equilibirum condition imposed to the stress field by Newton's second law

$$\nabla \cdot \boldsymbol{\sigma} + \mathbf{f} = 0 \qquad\qquad \sigma_{ji,j} + f_i = 0$$

2. **(Kinematics Equations)** express deformation with respect to displacement without reference to forces that created them.

$$\boldsymbol{\varepsilon} = \frac{1}{2}\left(\nabla\mathbf{u} + \nabla\mathbf{u}^T\right) \qquad\qquad \varepsilon_{ij} = \frac{1}{2}\left(u_{i,j} + u_{j,i}\right)$$

3. **(Constitutive Law)** describes behavior of material under stress. We approximate behavior of linear elastic material with Hooke's Law

$$\boldsymbol{\sigma} = \mathbf{C} : \boldsymbol{\varepsilon} \qquad\qquad \sigma_{ij} = C_{ijkl}\varepsilon_{kl}$$

where $\mathbf{C}$ is a 4-th order stiffness tensor encoding material properties. Specifically, under the assumption of isotropic material, the relationship is simplified to

$$\boldsymbol{\sigma} = 2\mu\boldsymbol{\varepsilon} + \lambda \operatorname{tr} \boldsymbol{\varepsilon}\mathbf{I} \qquad\qquad \sigma_{ij} = 2\mu\varepsilon_{ij} + \lambda\varepsilon_{kk}\delta_{ij}$$

where λ, μ are Lamé's first and second parameter can be written as material-specific constants - Poisson's ratio ν and Young's modulus E,

$$\mu = \frac{E}{2(1+\nu)} \qquad\qquad \lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}$$

2.4 Boundary Value Problem

For our purposes, it is easier to eliminate stress and strain to seek a set of equations where the displacement field is the primary unknown. We can achieve this by writing the strain $\boldsymbol{\sigma}$ in the equilibrium equation as a function of displacement $\mathbf{u}$ using the constitutive law and kinematics equations. In other words, we seek displacement field $\mathbf{u}$ in $\mathbb{R}^3$ satisfying

$$\begin{aligned} -\nabla \cdot \mathbf{C}\left(\boldsymbol{\varepsilon}(\mathbf{u})\right) &= \mathbf{f} && \mathbf{x} \in \Omega \\ \mathbf{u} &= 0 && \mathbf{x} \in \Gamma \end{aligned}$$

where $\Omega \in \mathbb{R}^3$ is a closed and bounded region occupied by a linear elastic solid, with boundary $\partial\Omega$. $\Gamma \subset \partial\Omega$ is part of boundary that the object cannot be displaced. $\mathcal{V} := \left\{\mathbf{v} \in H(\Omega, \mathbb{R}^3) \mid \mathbf{v}|_\Gamma = \mathbf{0}\right\}$ is a first Sobolev space satisfying the zero boundary condition on Γ. $\boldsymbol{\varepsilon}$ is a linear differential operator over the displacement field $\mathbf{u}$, while $\mathbf{C}$ is a linear operator over the the strain field $\boldsymbol{\varepsilon}(\mathbf{u})$, which is positive definite for linear elastic isotropic material. It is easy to see that $\mathcal{L} := -\nabla \cdot (\mathbf{C}\boldsymbol{\varepsilon}(\cdot))$ is a linear differential operator.

3 Finite Elements

It is conventional to use finite element method to solve linear elasticity equations for 3-dimensional objects with complex geometry. We will formulate the boundary value problem in both the weak and variational form. The two formulations are equivalent in our setup. We will then discuss computer implementation strategies on tetrahedron elements, and briefly on regular hexahedron elements.

3.1 Weak Form

The goal is to estimate $\mathbf{u}$ in a finite dimensional function space $\mathcal{V}_m \subset \mathcal{V}$ spanned by a set of basis functions $\{\mathbf{u}_1, \cdots, \mathbf{u}_m\} \subset \mathcal{V}$. We can formulate linear elasticity weakly as

$$\int_\Omega \mathbf{C}\boldsymbol{\varepsilon}(\mathbf{u}) : \boldsymbol{\varepsilon}(\mathbf{v}) d\mathbf{x} = \int_\Omega \mathbf{f} \cdot \mathbf{u} d\mathbf{x}$$

for all $\mathbf{v} \in \mathcal{V}$. The left hand side is obtained via Green's first identity assuming a pure displacement formulation with no traction force on the boundary. [8] We can define a bilinear form $\tilde{a}(\cdot, \cdot)$

$$\tilde{a}(\mathbf{u}, \mathbf{v}) = \langle \mathcal{L}\mathbf{u}, \mathbf{v} \rangle = \int_\Omega \mathbf{C}\boldsymbol{\varepsilon}(\mathbf{u}) : \boldsymbol{\varepsilon}(\mathbf{v}) d\mathbf{x}$$

where $\langle \cdot, \cdot \rangle$ is L^2 norm over Ω.

3.2 Variational Form

By theorem 9.1 in [9] and the following proposition, $\mathcal{L}\mathbf{u} = \mathbf{f}$ is the Euler-Lagrange equation of following variational problem

$$\mathcal{J}(\mathbf{u}) := \tilde{a}(\mathbf{u}, \mathbf{u}) - 2\langle \mathbf{f}, \mathbf{u} \rangle$$

where $\mathcal{J} : \mathcal{V} \to \mathbb{R}$ is a functional and $\mathbf{u} \in \mathcal{V}$. The theorem also guarantees the uniqueness and existence of solution for the boundary value problem. Therefore, we can solve for an equivalent

optimization problem

$$\min_{\mathbf{u}\in\mathcal{V}} \ \mathcal{J}(\mathbf{u}) \quad \text{where} \quad \mathcal{J}(\mathbf{u}) = \underbrace{\frac{1}{2}\int_\Omega \mathbf{C}\boldsymbol{\varepsilon}(\mathbf{u}) : \boldsymbol{\varepsilon}(\mathbf{u}) d\mathbf{x}}_{\text{potential energy}} - \underbrace{\int_\Omega \mathbf{f}\cdot\mathbf{u} d\mathbf{x}}_{\text{work}}$$

This amounts to applying the principle of least work to mechanical systems. Simply, we are minimizing the internal potential energy and work done by external force (e.g. gravity) on the object.

3.3 Discretization

There are many ways to discretize the domain. We will clarify computer implementations on tetrahedron elements first, and then briefly explain how we can adapt similar strategies for regular hexahedron elements. We construct a mesh $\mathcal{T}$ that partitions the domain Ω where each $\Omega^e \in \mathcal{T}$ is a tetrahedron and n is number of vertices of the mesh. We define a linear nodal basis function that is 0 at all vertices of the mesh, except has value of 1 at one vertex . Note the boundary value equations are vector valued, so we compose the same scalar valued nodal basis function for each component of the vector field. Therefore, we arrive at $3n$ linear basis that form a finite dimensional subspace of $\mathcal{V}$,

$$\mathcal{V}_{3n} = \text{span}\left\{\varphi_1, \cdots, \varphi_{3n}\right\}$$

The variational formulation over the discretized domain is then as follows,

$$\min_{\mathbf{d}\in\mathbb{R}^{3n}} \ \mathcal{J}_{3n}(\mathbf{d}) \quad \text{where} \quad \mathcal{J}_{3n}(\mathbf{d}) := \mathcal{J}(\mathbf{u}) = \sum_{\Omega^e\in\mathcal{T}} \left[\frac{1}{2}\int_{\Omega^e} \mathbf{C}\boldsymbol{\varepsilon}(\mathbf{u}) : \boldsymbol{\varepsilon}(\mathbf{u}) d\mathbf{x} - \int_{\Omega^e} \mathbf{f}\cdot\mathbf{u} d\mathbf{x}\right]$$

where $\mathbf{d} \in \mathbb{R}^{3n}$ are coefficients to linear nodal basis functions and could be interpreted as displacements in 3 coordinate directions at each vertex of the mesh. The displacement field $\mathbf{u}$ and coefficient $\mathbf{d}$ are related as follows in the above equation,

$$\mathbf{u} = \begin{pmatrix} \sum_{i=1}^n \xi_i\varphi_i \\ \sum_{i=1}^n \eta_i\varphi_i \\ \sum_{i=1}^n \zeta_i\varphi_i \end{pmatrix} = \begin{bmatrix} \varphi_1 & 0 & 0 & \varphi_2 & 0 & 0 & \cdots & \varphi_n & 0 & 0 \\ 0 & \varphi_1 & 0 & 0 & \varphi_2 & 0 & \cdots & 0 & \varphi_n & 0 \\ 0 & 0 & \varphi_1 & 0 & 0 & \varphi_2 & \cdots & 0 & 0 & \varphi_n \end{bmatrix} \begin{bmatrix} \xi_1 \\ \eta_1 \\ \zeta_1 \\ \xi_2 \\ \eta_2 \\ \zeta_2 \\ \vdots \\ \xi_n \\ \eta_n \\ \zeta_n \end{bmatrix} = \boldsymbol{\varphi}\mathbf{d}$$

where $\boldsymbol{\varphi}_{ij}$ is nodal basis for i-component of vector field at vertex j. Here, we impose a node-based ordering, in particularly, we can write

$$\mathbf{d}^e = \big(\quad \underbrace{\xi_1 \quad \eta_1 \quad \zeta_1}_{\mathbf{d}_1^e} \quad \underbrace{\xi_2 \quad \eta_2 \quad \zeta_2}_{\mathbf{d}_2^e} \quad \underbrace{\xi_3 \quad \eta_3 \quad \zeta_3}_{\mathbf{d}_3^e} \quad \underbrace{\xi_4 \quad \eta_4 \quad \zeta_4}_{\mathbf{d}_4^e} \quad \big)$$

as the nodal displacements at each vertex of a single tetrahedron $\Omega^e \in \mathcal{T}$. It important to note

that when integrating over the volume of an element, the nodal basis is a function of the variable of integration $\mathbf{x}$, while the coefficients $\mathbf{d}$ is not a function of $\mathbf{x}$, i.e.

$$\mathbf{u}^e(\mathbf{x}) = \boldsymbol{\varphi}^e(\mathbf{x})\mathbf{d}^e$$

where

$$\boldsymbol{\varphi}^e(\mathbf{x}) = \begin{bmatrix} \varphi_1(\mathbf{x}) & 0 & 0 & \cdots & \varphi_4(\mathbf{x}) & 0 & 0 \\ 0 & \varphi_1(\mathbf{x}) & 0 & \cdots & 0 & \varphi_4(\mathbf{x}) & 0 \\ 0 & 0 & \varphi_1(\mathbf{x}) & \cdots & 0 & 0 & \varphi_4(\mathbf{x}) \end{bmatrix} \qquad \mathbf{d}^e = \begin{bmatrix} \xi_1 \\ \eta_1 \\ \zeta_1 \\ \xi_2 \\ \eta_2 \\ \zeta_2 \\ \xi_3 \\ \eta_3 \\ \zeta_3 \\ \xi_4 \\ \eta_4 \\ \zeta_4 \end{bmatrix}$$

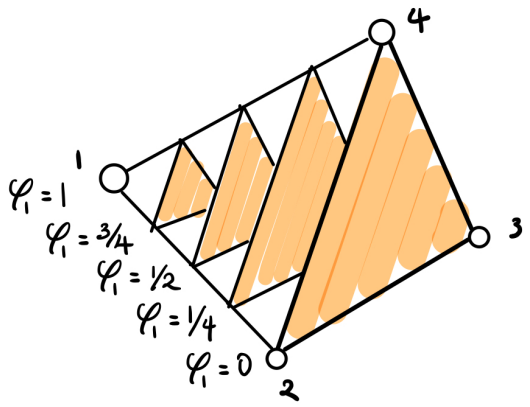


Figure 2: Visualization of φ_1 over a tetrahedron element

3.4 Element Stiffness Matrix

We will proceed with derivation of solution for the variational formulation of the problem on the discretized domain. In practice, it is more convenient to represent integrands in matrix notation. For a single tetrahedron,

$$\frac{1}{2}\int_{\Omega^e} \mathbf{C}\boldsymbol{\varepsilon}(\mathbf{u}^e) : \boldsymbol{\varepsilon}(\mathbf{u}^e)d\mathbf{x} - \int_{\Omega^e} \mathbf{f}\cdot\mathbf{u}^e d\mathbf{x} = \frac{1}{2}\int_{\Omega^e} \boldsymbol{\varepsilon}(\mathbf{u}^e)^T\mathbf{C}\boldsymbol{\varepsilon}(\mathbf{u}^e)d\mathbf{x} - \int_{\Omega^e} \mathbf{f}^T\mathbf{u}^e d\mathbf{x}$$

Evaluating linear basis function for any $\mathbf{x} = (x, y, z)$ over a tetrahedron with 4 vertices $\mathbf{x}_i$ for $i = 1, \cdots, 4$ is equivalent to determining the barycentric coordinate of $\mathbf{x}$. This is straight-forward

to compute by using the inverse of the following linear relationship,

$$\begin{bmatrix} 1 \\ x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \end{bmatrix} \begin{bmatrix} \varphi_1 \\ \varphi_2 \\ \varphi_3 \\ \varphi_4 \end{bmatrix}$$

Linear operators $\mathbf{C}$ and ε can be represented using matrices as follows

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \partial/\partial x & 0 & 0 \\ 0 & \partial/\partial x & 0 \\ 0 & 0 & \partial/\partial x \\ \partial/\partial y & \partial/\partial x & 0 \\ 0 & \partial/\partial z & \partial/\partial y \\ \partial/\partial z & 0 & \partial/\partial x \end{bmatrix} \qquad \mathbf{C} = \begin{bmatrix} \lambda+2\mu & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda+2\mu & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & \lambda+2\mu & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{bmatrix}$$

To simplify the expression, we want to write $\boldsymbol{\varepsilon}(\mathbf{u}^e)$ with respect to $\mathbf{d}^e$. We can define

$$\mathbf{B}^e = \boldsymbol{\varepsilon}\boldsymbol{\varphi}^e = \begin{bmatrix} \mathbf{B}_1^e & \mathbf{B}_2^e & \mathbf{B}_3^e & \mathbf{B}_4^e \end{bmatrix} \qquad \mathbf{B}_i^e = \begin{bmatrix} \partial\varphi_i/\partial x & 0 & 0 \\ 0 & \partial\varphi_i/\partial x & 0 \\ 0 & 0 & \partial\varphi_i/\partial x \\ \partial\varphi_i/\partial y & \partial\varphi_i/\partial x & 0 \\ 0 & \partial\varphi_i/\partial z & \varphi_i\partial/\partial y \\ \partial\varphi_i/\partial z & 0 & \partial\varphi_i/\partial x \end{bmatrix}$$

such that

$$\boldsymbol{\varepsilon}(\mathbf{u}^e) = \boldsymbol{\varepsilon}\boldsymbol{\varphi}^e\mathbf{d}^e = \mathbf{B}^e\mathbf{d}^e$$

Therefore the functional reduces to

$$\begin{aligned} \frac{1}{2}\int_{\Omega^e} \boldsymbol{\varepsilon}(\mathbf{u}^e)^T\mathbf{C}\varepsilon(\mathbf{u}^e)d\mathbf{x} - \int_{\Omega^e}\mathbf{f}^T\mathbf{u}^e d\mathbf{x} &= \frac{1}{2}\int_{\Omega^e}(\mathbf{B}^e\mathbf{d}^e)^T\mathbf{C}(\mathbf{B}^e\mathbf{d}^e)d\mathbf{x} - \int_{\Omega^e}\mathbf{f}^T\boldsymbol{\varphi}^e\mathbf{d}^e d\mathbf{x} \\ &= \frac{1}{2}(\mathbf{d}^e)^T\left(\int_{\Omega^e}(\mathbf{B}^e)^T\mathbf{C}\mathbf{B}^e d\mathbf{x}\right)\mathbf{d}^e - \left(\int_{\Omega^e}\mathbf{f}^T\boldsymbol{\varphi}^e d\mathbf{x}\right)\mathbf{d}^e \\ &= \frac{1}{2}(\mathbf{d}^e)^T\mathbf{K}^e\mathbf{d}^e - \mathbf{F}^e\mathbf{d}^e \end{aligned}$$

where $\mathbf{K}^e \in \mathbb{R}^{12x12}$ is termed *element stiffness matrix*,

$$\mathbf{K}^e = \int_{\Omega^e}(\mathbf{B}^e)^T\mathbf{C}\mathbf{B}^e d\mathbf{x} \qquad \mathbf{F}^e = \int_{\Omega^e}\mathbf{f}^T\boldsymbol{\varphi}^e d\mathbf{x}$$

Tetrahedron elements has the nice property that $\mathbf{B}^e$ is a constant function of $\mathbf{x}$. Therefore, $\mathbf{K}^e = (\mathbf{B}^e)^T\mathbf{C}\mathbf{B}^e|\Omega^e|$ is constant inside the element, where $|\cdot|$ denotes *volume of*. In general, this is not true. We need to resort to 3-dimensional Gaussian quadrature to numerically integrate the integral for elements like regular hexahedron.

3.5 Global Stiffness Matrix

Now return to the original functional,

$$\mathcal{J}_{3n}(\mathbf{d}) = \sum_{\Omega^e\in\mathcal{T}}\left[\frac{1}{2}(\mathbf{d}^e)^T\mathbf{K}^e\mathbf{d}^e - \mathbf{F}^e\mathbf{d}^e\right] = \frac{1}{2}\mathbf{d}^T\mathbf{K}\mathbf{d} - \mathbf{d}^T\mathbf{F}$$

for some $\mathbf{K} \in \mathbb{R}^{3nx3n}$, $\mathbf{F} \in \mathbb{R}^{3nx1}$. The last step is always possible, as $\mathcal{J}_{3n}$ is quadratic with respect to $\mathbf{d}$. Here $\mathbf{K}$ is termed *global stiffness matrix*. In practice, we need to keep a consistent indexing (node-based ordring) to map the contribution of element stiffness matrix $\mathbf{K}^e$ to corresponding location of global stiffness matrix $\mathbf{K}$. In spirit of Ritz Method, we set the gradient of $\mathcal{J}_{3n}$ to $\mathbf{0}$,

$$\nabla_{\mathbf{d}} \left(\frac{1}{2} \mathbf{d}^T \mathbf{K} \mathbf{d} - \mathbf{d}^T \mathbf{F} \right) = \mathbf{0} \quad \Rightarrow \quad \mathbf{K}\mathbf{d} = \mathbf{F}$$

Solving the boundary value problem is then reduced to solving a linear system $\mathbf{K}\mathbf{d} = \mathbf{F}$. In general, $\mathbf{K}$ is a symmetric positive definite matrix.

3.6 Linear Hexahedron Element

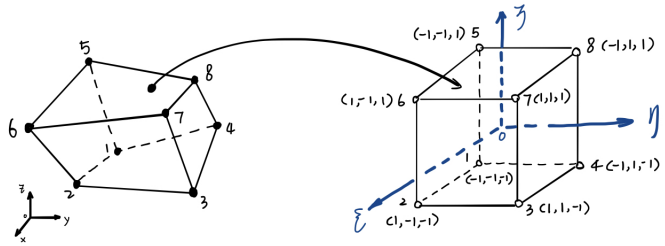


Figure 3: Change of coordinate system on hexahedron element

Apart from tetrahedral elements, we also experimented with regular hexahedral elements with linear basis functions. Specifically, regular hexahedronal elements are equal-sized unit cubes that are perfectly aligned with a cartesian grid, i.e. voxel-based discretization. For each hexahedron element, there are 8 nodes numbered in a counter-clockwise manner. To makes it easier to construct the basis functions and to evaluate the matrix integral, we define a natural coordinate system (ξ, η, ζ) with its origin at the centre of the transformed cube. Unlike tetrahedral elements, the basis functions are chosen to be tri-linear functions,

$$\varphi_i = \frac{1}{8}(1 + \xi\xi_i)(1 + \eta\eta_i)(1 + \zeta\zeta_i)$$

where (ξ_i, η_i, ζ_i) denotes the natural coordinates of each node i. Similar to tetrahedrons, the strain matrix $\mathbf{B}$ could be written as

$$\mathbf{B}^e = \begin{bmatrix} \mathbf{B}_1^e & \mathbf{B}_2^e & \mathbf{B}_3^e & \mathbf{B}_4^e & \mathbf{B}_5^e & \mathbf{B}_6^e & \mathbf{B}_7^e & \mathbf{B}_8^e \end{bmatrix} \qquad \mathbf{B}_i^e = \begin{bmatrix} \partial\varphi_i/\partial x & 0 & 0 \\ 0 & \partial\varphi_i/\partial x & 0 \\ 0 & 0 & \partial\varphi_i/\partial x \\ \partial\varphi_i/\partial y & \partial\varphi_i/\partial x & 0 \\ 0 & \partial\varphi_i/\partial z & \varphi_i\partial/\partial y \\ \partial\varphi_i/\partial z & 0 & \partial\varphi_i/\partial x \end{bmatrix}$$

However, since the basis functions are defined in terms of the natural coordinates, the chain rule of partial differentiation needs to be used to obtain the derivatives with respect to global coordinates,

$$\begin{bmatrix} \partial\varphi_i/\partial x \\ \partial\varphi_i/\partial y \\ \partial\varphi_i/\partial z \end{bmatrix} = \mathbf{J}^{-1} \begin{bmatrix} \partial\varphi_i/\partial\xi \\ \partial\varphi_i/\partial\eta \\ \partial\varphi_i/\partial\zeta \end{bmatrix} \qquad \text{where} \quad \mathbf{J} = \begin{bmatrix} \Sigma_{j=1}^8 x_j \partial\varphi_j/\partial\xi & \Sigma_{j=1}^8 y_j \partial\varphi_j/\partial\xi & \Sigma_{j=1}^8 z_j \partial\varphi_j/\partial\xi \\ \Sigma_{j=1}^8 x_j \partial\varphi_j/\partial\eta & \Sigma_{j=1}^8 y_j \partial\varphi_j/\partial\eta & \Sigma_{j=1}^8 z_j \partial\varphi_j/\partial\eta \\ \Sigma_{j=1}^8 x_j \partial\varphi_j/\partial\zeta & \Sigma_{j=1}^8 y_j \partial\varphi_j/\partial\zeta & \Sigma_{j=1}^8 z_j \partial\varphi_j/\partial\zeta \end{bmatrix}$$

Hence the element stiffness matrix for each voxel element is,

$$\mathbf{K}^e = \int_{\Omega^e} (\mathbf{B}^e)^T \mathbf{C} \mathbf{B}^e d\mathbf{x} = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} (\mathbf{B}^e)^T \mathbf{C} \mathbf{B}^e \det(\mathbf{J}) d\xi d\eta d\zeta$$

Note that the strain matrix $\mathbf{B}^e$ is a function of ξ, η and ζ and matrix C is the material constant. Since the integral cannot be easily solved analytically, we used a gaussian cubature for numerical integration

$$\mathbf{K}^e == \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} f(\xi,\eta,\zeta) d\xi d\eta d\zeta = \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N} \omega_i \omega_j \omega_k f(\xi_i, \eta_j, \zeta_k)$$

Most importantly, since all the voxels have exactly the same shape, only a single pre-computed element stiffness matirx is needed, which greatly reduces the memory requirements and computation time. This harmonizes well with matrix-free iterative methods.

3.7 Dirichlet Boundary Condition

We can enforce the dirichlet boundary condition, by considering a subset of linear equations. Specifically, let $\mathbf{I} = \mathbf{D} + \tilde{\mathbf{D}}$, where $\mathbf{D}_{ii} = 1$ for all i corresponding to vertices at Dirichlet boundary. We can rewrite the system of linear equations as follows,

$$\begin{aligned} \mathbf{K}(\mathbf{D} + \tilde{\mathbf{D}})\mathbf{d} &= \mathbf{F} \\ \mathbf{K}\tilde{\mathbf{D}}\mathbf{d} &= \mathbf{F} - \mathbf{K}\mathbf{D}\mathbf{x} \\ \tilde{\mathbf{D}}\mathbf{K}\tilde{\mathbf{D}}\mathbf{d} &= \tilde{\mathbf{D}}\mathbf{F} - \tilde{\mathbf{D}}\mathbf{K}\mathbf{D}\mathbf{x} \end{aligned}$$

where $\tilde{\mathbf{D}}\mathbf{K}\mathbf{D}\mathbf{x} = \mathbf{0}$, so we have

$$\tilde{\mathbf{D}}\mathbf{K}\tilde{\mathbf{D}}\mathbf{d} = \tilde{\mathbf{D}}\mathbf{F}$$

Effectively, we zero-ed out entries on both left and right hand side of the linear system at indices corresponding to vertices on Dirichlet boundary. We can collapse the matrix $\mathbf{K}$ by removing zero rows/columns and obtain a matrix that remains to be symmetric positive definite.

4 Iterative Methods

We are left with the task of solving the linear system

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

where $\mathbf{A}$ is a sparse symmetric positive definite matrix. Cholesky decomposition is the standard method to solve a symmetric positive definite linear system. However, exact solution is unnecessary for the purpose of visualization. In this section, we will explore a few iterative method for computing approximate solutions that are *good enough*.

4.1 Stationary Iteration

Consider the standard splitting of $\mathbf{A}$,

$$\mathbf{A} = \mathbf{D} - \mathbf{L} - \mathbf{U}$$

where $\mathbf{D}, \mathbf{L}, \mathbf{U}$ are the diagonal, negative of strictly lower-triangular, and negative of strictly upper-triangular parts of $\mathbf{A}$. We consider a linear one-step stationary scheme

$$\mathbf{x}^{(k+1)} = \mathbf{H}\mathbf{x}^{(k)} + \mathbf{v}$$

for some $\mathbf{H}, \mathbf{v}$. Specifically, we are interested in stationary methods that are convergent for symmetric positive definite $\mathbf{A}$. Initially, we started with Jacobi's method, where

$$\mathbf{H} = \mathbf{P}^{-1}(\mathbf{L} + \mathbf{U}) = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A} \qquad \mathbf{v} = \mathbf{P}^{-1}\mathbf{b}$$

and element-wise iteration

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_1 - \sum_{j \neq i} a_{ij} x_j^{(k)} \right)$$

Jacobi's iteration is embarrassingly parrallel, as $\mathbf{x}^{k+1}$ is only dependent upon previous $\mathbf{x}^k$ and so every iteration can be fully parallelised. However, positive definiteness of $\mathbf{A}$ is not sufficient for convergence. In general, linear elasticity problem does not satisfy $\rho(\mathbf{H}) < 1$, which is an equivalent condition for convergence given nonsingular $\mathbf{A}$,$\mathbf{P}$. [9] Instead, we used weighted or dampled Jacobi (ω-Jacobi)

$$\mathbf{H} = \mathbf{I} - \omega\mathbf{D}^{-1}\mathbf{A} \qquad \mathbf{v} = \omega\mathbf{D}^{-1}\mathbf{b}$$

and element-wise iteration

$$x_i^{(k+1)} = \frac{\omega}{a_{ii}} \left(b_i - \sum_{j=i} a_{ij} x_j^{(k)} \right) + (1-\omega) x_i^{(k)}$$

for some $\omega \in (0, 1)$. In practice, ω forces eigenvalues of $\mathbf{H}$ to approach 1, and allows for convergence for some problems that is otherwise divergent with Jacobi's method. Another stationary method we want to explore is Successive Over-Relaxation (SOR) method,

$$\mathbf{H} = (\mathbf{I} - \omega\mathbf{D}^{-1}\mathbf{L})^{-1} \left((1-\omega)\mathbf{I} + \omega\mathbf{D}^{-1}\mathbf{U} \right) \qquad \mathbf{v} = \omega(\mathbf{I} - \omega\mathbf{D}^{-1}\mathbf{L})^{-1}\mathbf{D}^{-1}\mathbf{b}$$

and element-wise iteration

$$x_i^{(k+1)} = \frac{\omega}{a_{ii}} \left(b_i - \sum_{j<i} a_{ij} x^{(k+1)} - \sum_{j>i} a_{ij} x_j^{(k)} \right) + (1-\omega) x_i^{(k)}$$

for some $\omega \in [1, 2)$. SOR reduces to Gauss-Seidel method for $\omega = 1$. Additionally, it can be proved that, given symmetric positive definite $\mathbf{A}$, SOR converges for any $\omega \in (0, 2)$. [10]. However, each iteration of SOR has data dependencies on both $\mathbf{x}^{(k)}$ and $\mathbf{x}^{(k+1)}$, and is therefore less inclined to be parallelised.

5 Numerical Experiments

The goal of this section is to experimentally investigate the performance of the few iterative methods described above and identify the viability of iterative methods in generating visually plausible simulation of the stress field. Briefly, we have implemented

1. FEM solver on linear tetrahedron and linear regular hexahedron

2. ω-Jacobi and SOR linear solvers for sparse matrices.

The solution to the linear system $\mathbf{Ku} = \mathbf{F}$ can be interpreted as a discrete displacement field sampled over vertices of the mesh. From the displacement field, we can compute the corresponding strain, stress, and von Mises' stress fields over the vertices. To visualize the result, we interpolate the discrete vector fields over the surface of the mesh using barycentric interpolation.

5.1 Mesh

We used a few mesh of interesting shapes and geometries to perform numerical experiments. Mainly, we used `archbridge` with a fixed number of vertices (1118) and tetrahedrons (3221). The tetrahedron elements are generated with `tetgen`, which is a delaunay-based tetrahedralization algorithm that generates mesh with good geometric properties. Regular hexahedron elements are constructed such that every vertex of the mesh is enclosed within some hexahedron elements, such that we can perform trilinear interpolation to derive the per-vertex vector fields over the original tetrahedron mesh.

5.2 Convergence

We use relative residual to measure convergence. We define residual at k-th iteration to be,

$$\mathbf{r}^{(k)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(k)}$$

The relative residual is therefore $\left\|\mathbf{r}^{(k)}\right\| / \left\|\mathbf{r}^{(0)}\right\|$, where $\mathbf{r}^{(0)} = \mathbf{b}$ if initial guess is a zero vector. In Figure 4, we experimentally measured the convergence of ω-Jacobi with respect to different values of weights ω. We noticed that the iterations diverges for $\omega = 0.6, 0.8, 1$, while converges for $\omega \leq 0.4$. This is expected, as value of ω gets smaller, the eigenvalues of Jacobi's iteration matrix $\mathbf{H}$ gets closeer to 1. We choose the best performing parameter $\omega = 0.4$ for subsequent experiments. In Figure 5, we experimentally measured the convergence of SOR with respect to different values of relaxation factor ω. As expected, every choice of ω leads to a convergent sequence of iterations. Although $\omega = 1.8$ converges faster in the long run, $\omega = 1.6$ converges more quickly for iterations less than approximately 500 iterations. As we are interested in using as few iterations for visualizing stress, we will use $\omega = 1.6$ for subsequent experiments.

5.3 Approximate Error

We define error at k-th iteration to be,

$$\mathbf{e}^{(k)} = \tilde{\mathbf{x}} - \mathbf{x}^{(k)}$$

where $\tilde{\mathbf{x}}$ is the real solution of the system. Although $\tilde{\mathbf{x}}$ is not known, we consider solution resulting from direct LDL/Cholesky solver (`mldivide`) as the ground truth, and use it in place of $\tilde{\mathbf{x}}$ to determine an estimate of true error at each iteration. In Figure 6, we quantified and compared the performance of ω-Jacobi ($\omega = 0.4$) and SOR ($\omega = 1.6$) by computing the error norm $\|\mathbf{e}\|$. We observed that SOR, with a good choice of ω, is much superior to ω-Jacobi.

5.4 Visualizing Stress On Tetrahedron

Ultimately, we are interested in getting an idea of how few iterations are needed to create a *sensible* visualization of the effective stress field. By *sensible*, we meant that the viewer should be able

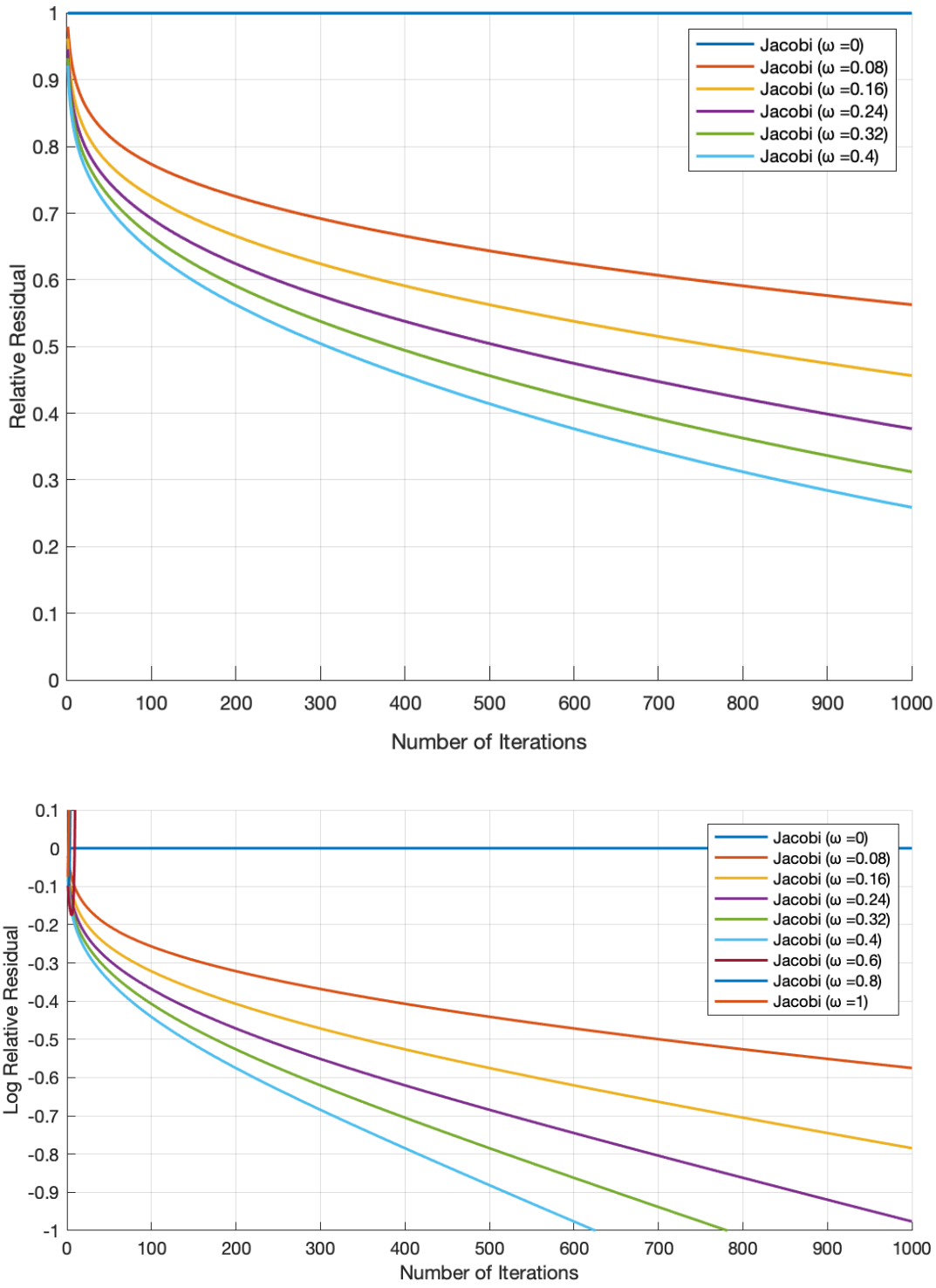


Figure 4: ω-Jacobi's relative residual (top) and log relative residual (bottom) plotted as a function of number of iterations. Each line represent Jacobi's Method with different weights

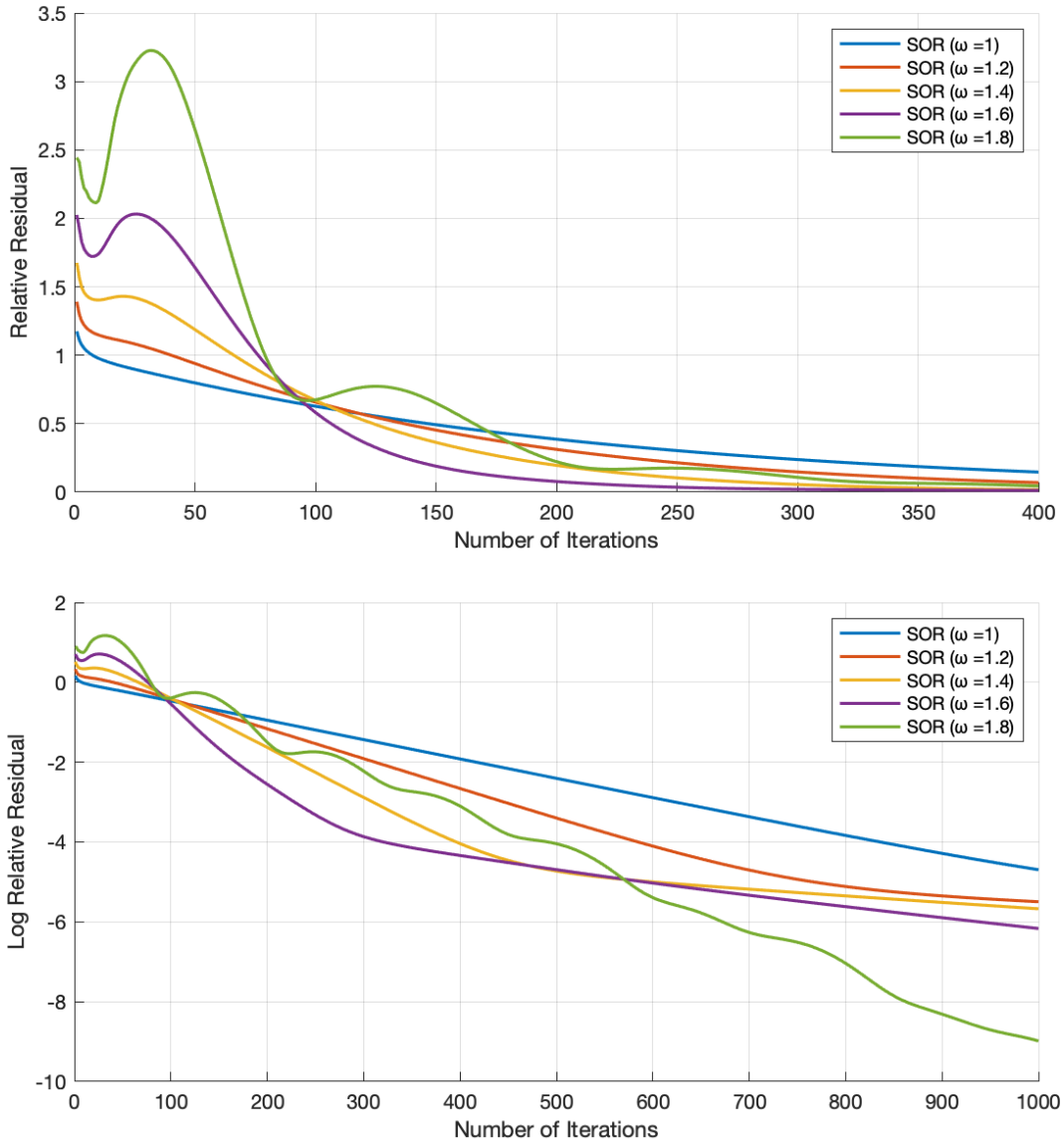


Figure 5: SOR's relative residual (top) and log relative residual (bottom) plotted as a function of number of iterations. Each line represent SOR iteration with different weights

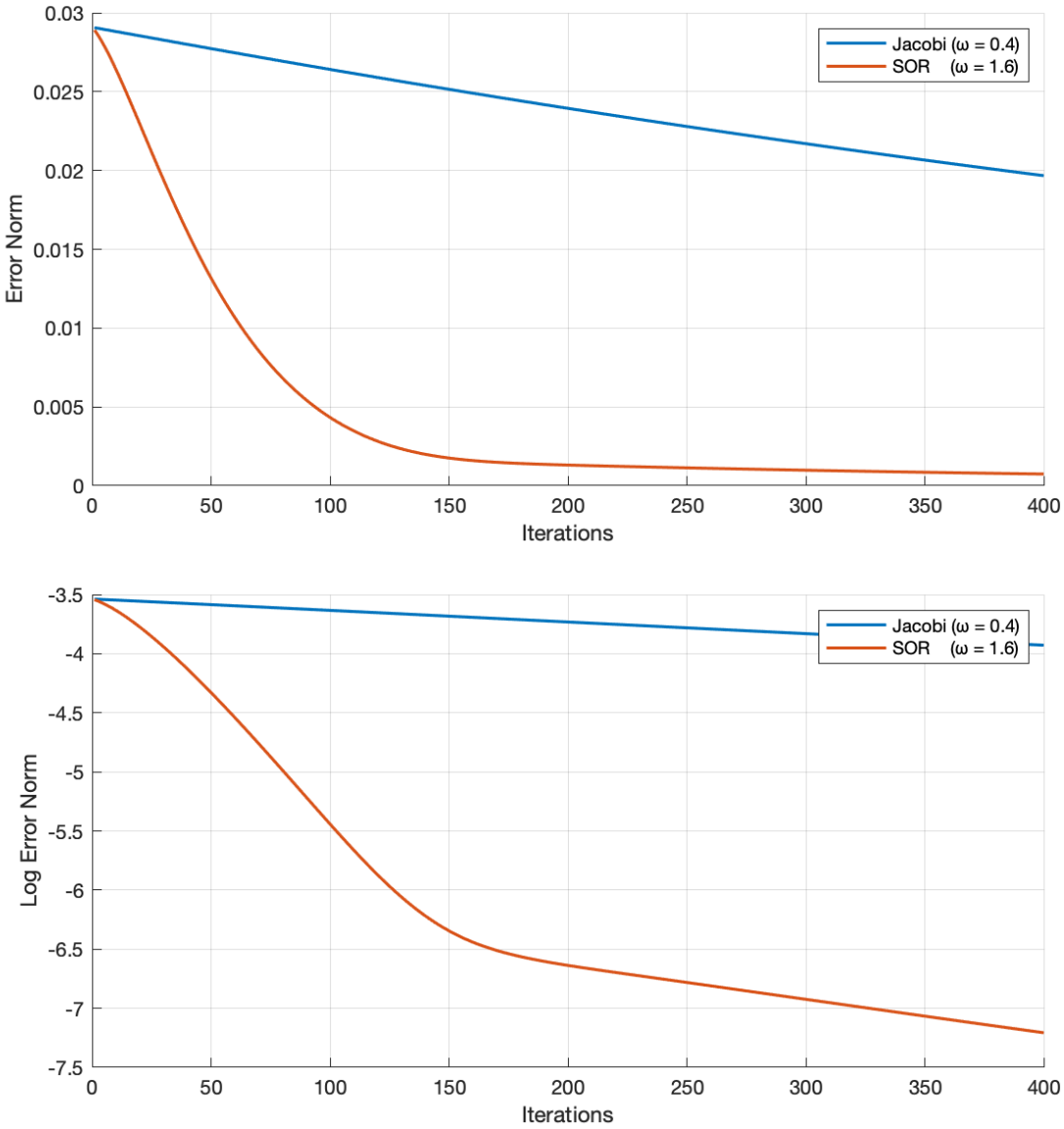


Figure 6: Error (top) and log error (bottom) plotted as a function of number of iterations for different iterative methods

to get a rough idea of the distribution of the effective stress field, minor local variations can be easily dismissed as long as the global trend is captured with the help of the approximate solutions generated by iterative methods. In Figure 7, we plotted the gold standard effective stress field solved using matlab's `mldivide` solver. In Figure 8, we visualized the effective stress field generated using

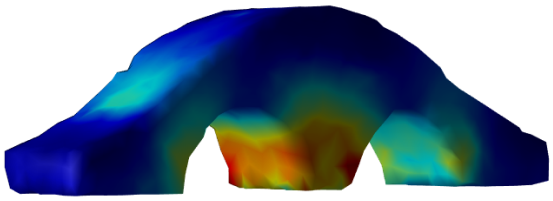


Figure 7: von Minses' effective stress field solved using `mldivide`

ω-Jacobi ($\omega = 0.4$) and SOR ($\omega = 1.6$). Qualitatively, visualization generated using SOR method starting at iteration 200 looks almost indistinguishable from the gold standard., whereas similar observations could be made for ω-Jacobi starting at iteration 500. This suggests that a relative residual of 0.1, resulting in an approximate erorr norm of 0.001, is a *good enough* stopping criterion to guarantee a *sensible* visualization on this particular mesh.

5.5 Visualizing Stress On Hexahedron

Moreover, the most significant benefit obtained from using equal-sized voxels is to construct element stiffness matrix $\mathbf{K}^e$ once and avoid the assembly of global stiffness matrix. To generate comparable visualizations effectively, our strategy is to first conservatively voxelize the original tetdrahedron mesh, where each point (not just vertex) on the tetrahedron mesh is contained inside the voxelization, and then compute and map the nodal displacements on each grid points back to the input mesh in terms of trilinear interpolation. In Figure 9, we experimented with voxelizations with different resolutions and their effect on stress field visualization. We observed that for a moderately sided voxel grid $m = 58$, the visualization is similar to the gold standard. However, we acknowledge that the stress field do look different from those generated using tetrahedron element.

5.6 Supplementary Gallery

Visualization on different meshes is here 10!

6 Discussion & Conclusion

We have demonstrated that even with the simpliest iterative method, we can achieve undistinguishable stress field visualization from direct solvers, in a few number of iterations. This is all achieved without exploring more sophisticated iterative methods like conjugated gradient, multigrid, etc. or hand-tuned preconditioners. Additionally, we have demonstrated how we could save memory and time cost associated with assembly of stiffness matrices $\mathbf{K}$ with the help of regular hexahedron element. To conclude, matrix-free iterative methods over regular hexahedron elements have great potential in real-time linear elasticity simulation, especially for visualization purposes where *good enough* approximate solutions can generate *sensible* visualizations. However, there are complications along the way. For example, we need to experimentally determine the optimal parameter ω for

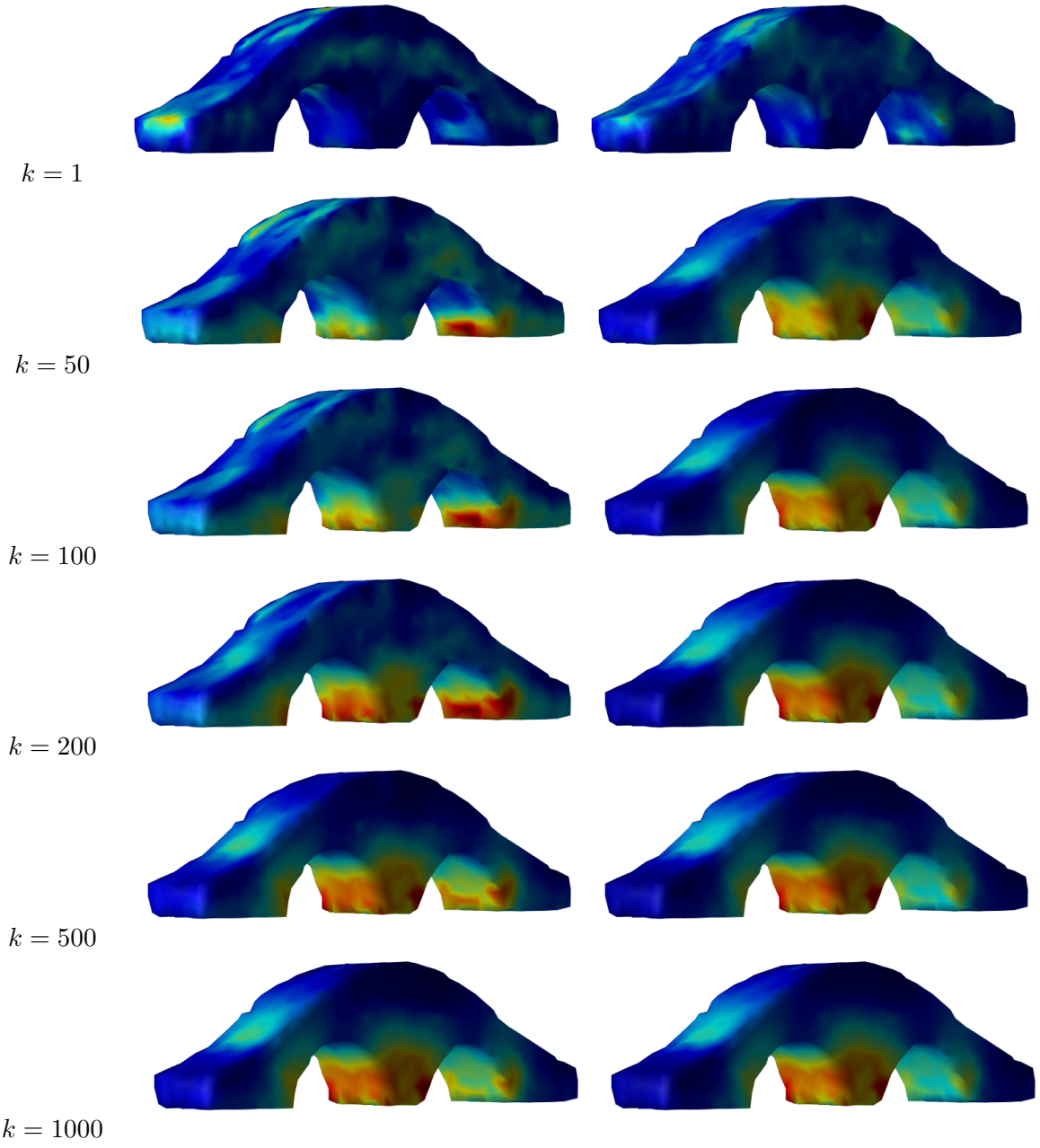
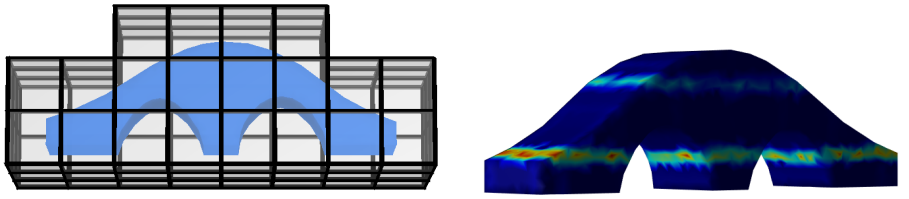
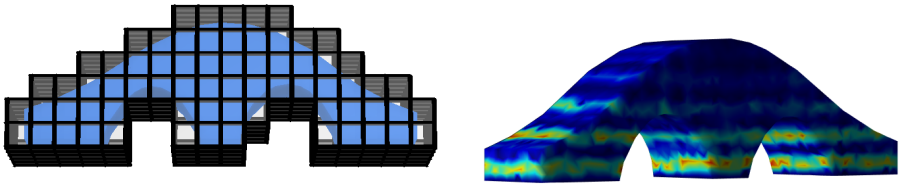


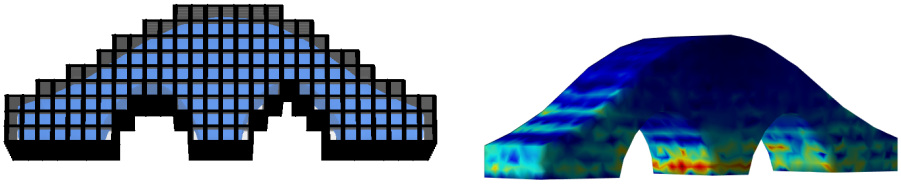
Figure 8: von Mises' effective stress field generated from ω-Jacobi ($\omega = 0.4$) and SOR ($\omega = 1.6$) at iterations $k = 1, 50, 100, 200, 500, 1000$



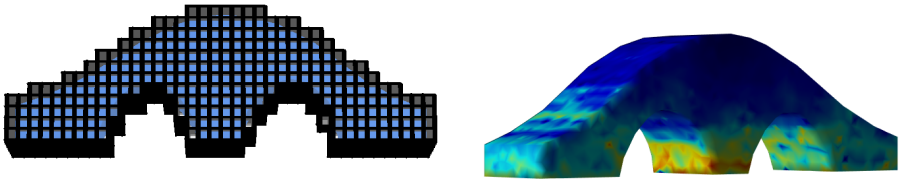
$m = 8$



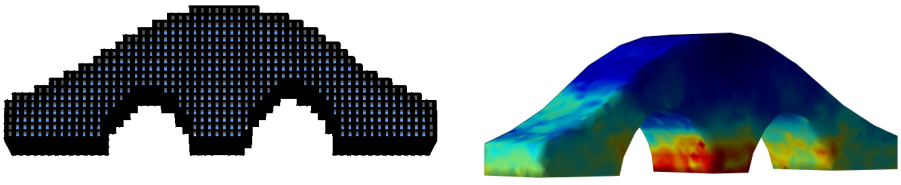
$m = 18$



$m = 28$



$m = 38$



$m = 58$

Figure 9: von Mises' effective stress field generated from interpolated hexahedral nodal displacements with respect to voxel grid resolutions $m = 8, 18, 28, 38, 58$, where m denotes the number of voxels in the bounding box along x direction

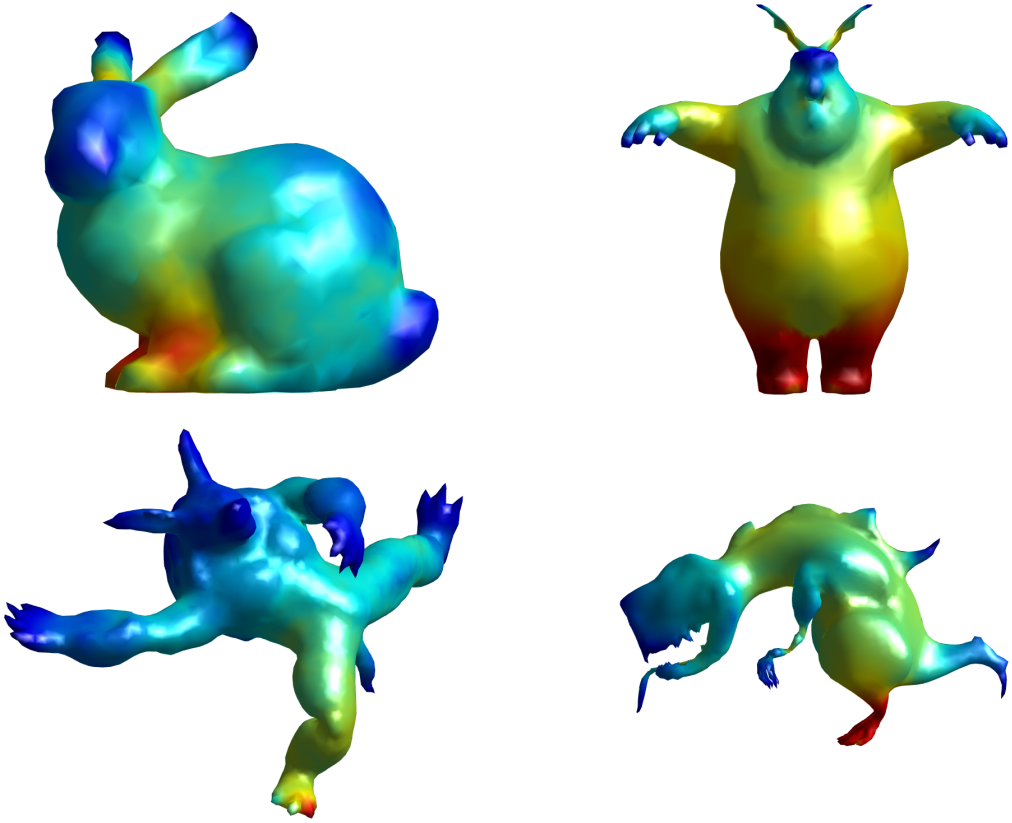


Figure 10: Stress field visualization on different meshes

both iterative method tested. Theories for finding optimal parameters have assumptions that linear leasticity simulation cannot make; So we had to resort to finding a good choice of ω experimentally via a simple grid search. Another complication comes with the use of trilinear interpolation to map stress field back from voxel grids to tetrahedron mesh. We noticed obvious artifacts when the resolution of the voxel grids is coarse.

7 Future Directions

Given the limited amount of time to undertake this project, there is still so much we had not have enough time to look at. A short list of to-dos is as follows,

1. Explore other linear solvers that is more robust and efficient, i.e. conjugate gradient, multigrid
2. Explore preconditioner that helps with solving linear elasticity problem faster
3. Explore different interpolation schemes for mapping stress field back from voxel grids to tetrahedron mesh
4. Implement linear solvers on GPU for real-time simulation

References

[1] Qingnan Zhou, Julian Panetta, and Denis Zorin. "Worst-case structural analysis". In: *ACM Transactions on Graphics* 32.4 (2013), p. 1. DOI: 10.1145/2461912.2461967.

[2] Timothy Langlois et al. "Stochastic structural analysis for context-aware design and fabrication". In: *ACM Transactions on Graphics* 35.6 (2016), pp. 1–13. DOI: 10.1145/2980179.2982436.

[3] Adriana Schulz et al. "Interactive design space exploration and optimization for CAD models". In: *ACM Transactions on Graphics* 36.4 (2017), pp. 1–14. DOI: 10.1145/3072959.3073688.

[4] Christian Dick, Joachim Georgii, and Rüdiger Westermann. "A real-time multigrid finite hexahedra method for elasticity simulation using CUDA". In: *Simulation Modelling Practice and Theory* 19.2 (2011), pp. 801–816. DOI: 10.1016/j.simpat.2010.11.005.

[5] Praveen Yadav and Krishnan Suresh. "Large Scale Finite Element Analysis Via Assembly-Free Deflated Conjugate Gradient". In: *Journal of Computing and Information Science in Engineering* 14.4 (2014), p. 041008. DOI: 10.1115/1.4028591.

[6] Carsten Konke. "Matrix-free voxel-based finite element method for materials with heterogeneous microstructures". In: (2018).

[7] Olivier A Bauchau and James I Craig. "Structural analysis". In: (2009).

[8] Magnus Jorstad. "Multigrid preconditioning of the linear elasticity equation". In: (2016).

[9] A Iserles. "A first course in the numerical analysis of differential equations". In: (2009).

[10] Y Saad. "Iterative methods for sparse linear systems, second edition". In: (2003).