

# A Large-Scale Comparison of Tetrahedral and Hexahedral Elements for Solving Elliptic PDEs with the Finite Element Method

TESEO SCHNEIDER, University of Victoria, Canada

YIXIN HU, New York University, USA

XIFENG GAO, Lightspeed & Quantum Studio, Tencent America, USA

JÉRÉMIE DUMAS, Adobe Research, USA

DENIS ZORIN, New York University, USA

DANIELE PANZZO, New York University, USA

The Finite Element Method (FEM) is widely used to solve discrete Partial Differential Equations (PDEs) in engineering and graphics applications. The popularity of FEM led to the development of a large family of variants, most of which require a tetrahedral or hexahedral mesh to construct the basis. While the theoretical properties of FEM basis (such as convergence rate, stability, etc.) are well understood under specific assumptions on the mesh quality, their practical performance, influenced *both* by the choice of the basis construction and quality of mesh generation, have not been systematically documented for large collections of automatically meshed 3D geometries.

We introduce a set of benchmark problems involving most commonly solved elliptic PDEs, starting from simple cases with an analytical solution, moving to commonly used test problem setups, and using manufactured solutions for thousands of real-world, automatically meshed geometries. For all these cases, we use state-of-the-art meshing tools to create both tetrahedral and hexahedral meshes, and compare the performance of different element types for common elliptic PDEs.

The goal of his benchmark is to enable comparison of complete FEM pipelines, from mesh generation to algebraic solver, and exploration of relative impact of different factors on the overall system performance.

As a specific application of our geometry and benchmark dataset, we explore the question of relative advantages of unstructured (triangular/tetrahedral) and structured (quadrilateral/hexahedral) discretizations. We observe that for Lagrange-type elements, while linear tetrahedral elements perform poorly, quadratic tetrahedral elements perform equally well or outperform hexahedral elements for our set of problems and currently available mesh generation algorithms. This observation suggests that for common problems in structural analysis, thermal analysis, and low Reynolds number flows, high-quality results can be obtained with unstructured tetrahedral meshes, which can be created robustly and automatically.

We release the description of the benchmark problems, meshes, and reference implementation of our testing infrastructure to enable statistically significant comparisons between different FE methods, which we hope will be helpful in the development of new meshing and FEA techniques.

**CCS Concepts:** • Computing methodologies → Mesh models; Mesh geometry models; Volumetric models; Simulation evaluation.

---

Authors' addresses: Teseo Schneider, teseo@uvic.ca, University of Victoria, Canada; Yixin Hu, New York University, USA; Xifeng Gao, Lightspeed & Quantum Studio, Tencent America, USA; Jérémie Dumas, Adobe Research, USA; Denis Zorin, New York University, USA; Daniele Panzzo, New York University, USA.

---

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2022 Association for Computing Machinery.

0730-0301/2022/1-ART1 \$15.00

<https://doi.org/10.1145/3508372>

Additional Key Words and Phrases: Tetrahedral Mesh, Hexahedral Mesh, Finite Element, Comparison

## ACM Reference Format:

Teseo Schneider, Yixin Hu, Xifeng Gao, Jérémie Dumas, Denis Zorin, and Daniele Panzzo. 2022. A Large-Scale Comparison of Tetrahedral and Hexahedral Elements for Solving Elliptic PDEs with the Finite Element Method. *ACM Trans. Graph.* 1, 1, Article 1 (January 2022), 14 pages. <https://doi.org/10.1145/3508372>

## 1 INTRODUCTION

The finite element method (FEM) is commonly used to discretize partial differential equations (PDEs), due to its generality, rich selection of elements adapted to specific problem types, and wide availability of commercial implementations. At a high level, a FE analysis code takes as input the domain boundary, the boundary conditions, and the governing equations of the phenomena of interest, and computes the solution everywhere in the domain.

As an initial step in this procedure, the domain typically has to be discretized in a finite collection of elements. Many choices are possible, ranging from unstructured grids of tetrahedra to perfectly regular grids of cubes. Despite the large amount of research on mesh generation, we were unable to find a systematic study answering a basic question: “What are the practical pros and cons of using unstructured (triangular/tetrahedral) or structured (quadrilateral/hexahedral/grids) discretizations for commonly used elliptic PDEs?”.

This question is critical to inform the development of meshing algorithms: while tetrahedral meshes are easier to generate automatically, hexahedral meshes (i.e., meshes that are composed of only deformed cubes) are much more difficult to adapt to objects with complex geometries, while maintaining high mesh quality. One of the arguments motivating development of these more complex algorithms is a common belief that hexahedral elements yield better accuracy for a given computational cost (see the introduction of, e.g., [Bernard et al. 2016; Guo et al. 2020; Lyon et al. 2016]).

The overall aim of our work is to provide an extensive benchmark for comparing the performance of FE pipelines, including automatic meshing, FE basis construction, and algebraic system solvers, on a set of most common elliptic PDEs and a set of realistic geometries. As an immediate application, we explore the performance of widely used families of elements, coupled with standard solvers, on a large set of meshes generated using currently available meshing algorithms.

More specifically, we compare the *efficiency* of different elements, that is, how much time is typically required to obtain a solution with a given accuracy for different element types on automatically generated unstructured meshes, on manually and automatically generated semi-structured meshes, and on regular lattices.

We consider standard Lagrangian bases [Ciarlet 2002a; Szabó and Babuška 1991] of varying degrees, as well as serendipity [Zienkiewicz et al. 2005] elements (for hexahedra only), which are by far the most popular brick element. Finally, we perform several comparisons using spline-based elements [Hughes et al. 2005], which have recently gained popularity in the IsoGeometric Analysis (IGA) community. While this clearly does not reflect the broad range of existing element types and PDEs in the literature, it includes the most popular general-purpose elements currently used in commercial and open-source FE systems. For solving the resulting linear systems, we consider both state of the art direct [De Coninck et al. 2016] and iterative solvers [Falgout and Yang 2002].

We collected a set of test problems of varying complexity for elliptic PDEs (including Poisson, linear elasticity, Neo-Hookean elasticity, incompressible elasticity, and incompressible Stokes equations). Our set includes common simple test problems (where most of the hex-meshes are grids): beam bending, beam twisting, driven cavity flow, planar domain with a hole, elasticity problems with singular solutions, as well as a large-scale benchmark of manufactured solutions [Salari and Knupp 2000] on 3 200 automatically meshed, real-world, complex 3D models. Our model collection includes both CAD models and scanned geometries, providing a realistic sampling of analysis scenarios. We use TetWild [Hu et al. 2020, 2018] and MeshGems [Spatial 2018] to generate the tetrahedral and hexahedral meshes respectively (we also included the state-of-the-art meshes from Hexalab [Bracci et al. 2019]).

This combination of test models, 3D meshes for these models, elements and solvers is representative of many common FE application scenarios.

We quantify (to the best of our knowledge, for the first time) the overall performance differences between these two families of elements. Our main conclusion is that, while linear elements on triangular/tetrahedral meshes exhibit well-known problems, quadratic tetrahedral elements perform similarly or better (i.e., require similar or less time to compute a solution with a given accuracy) than Lagrangian elements on semi-structured hexahedral meshes, and are somewhat inferior (but still competitive, especially considering tetrahedral meshing is much faster and more robust) to the performance of spline elements on regular lattices when a direct solver is used. Combined with available state-of-the-art robust meshing techniques, quadratic tetrahedral elements are a good choice to realize a fully automatic pipeline, e.g., for SciML applications, or shape optimization, without sacrificing performance compared to hexahedral elements, which require far more complex and less robust mesh generation. More detailed conclusions are presented in Section 6.

We emphasize that our study is limited to a specific set of PDEs, commonly used geometry-agnostic linear solvers, and state-of-the-art meshing algorithms; we leave adding dynamic scenarios, multi-physics, different linear solvers, and other extensions as future work – the provided framework can be readily extended to these cases. We

also note that adaptive refinement is simpler for hexahedral meshes and, as a consequence, adaptive geometric multigrid solvers are more readily available [Alzetta et al. 2018], although it is possible to develop similar solvers for tetrahedral meshes [Kohl et al. 2019]. While the outcome of our study should not be interpreted as a reason to favor tetrahedral discretizations in all situations (and there are applications of hexahedral meshes outside the scope of FEM discretizations, such as lattice structure design), it does point to the need for direct experimental evaluation of meshing strategies, in the context of specific target applications.

We provide the complete source code<sup>1</sup> for the integrated analysis pipelines we tested, the dataset we used<sup>2</sup>, the benchmark solutions, and the scripts to reproduce all results<sup>3</sup>, to enable researchers and practitioners to easily expand this study to additional mesh types (such as polyhedral meshes) and bases.

This study is divided into five sections: we first introduce the closest related works on meshing and analysis (Section 2). We then overview the background on mesh types, basis, and the model PDE that we consider in this study (Section 3). We divide the experimental evaluation into a set of individual experiments, targeting a set of common test problems (including problems with singularities) in Section 4, and then perform a large scale analysis on thousands of automatically generated meshes in Section 5. We finally draw conclusions and identify open challenges in Section 6.

## 2 RELATED WORK

We first review existing comparisons of different types of finite elements (Section 2.1), then briefly discuss commonly used finite element software (Section 2.2) and the state-of-the-art meshing algorithms (Section 2.3).

### 2.1 FEA on Unstructured and Structured Meshes

To the best of our knowledge, our study is the first large-scale comparison between different commonly used types of elements in FEM. However, there are multiple existing comparisons focused on specific models and physics.

In [Cifuentes and Kalbag 1992], the authors conclude that quadratic tetrahedral meshes lead to roughly the same accuracy and time as linear hexahedral meshes, by comparing solutions for several simple structural problems. By evaluating the eigenvalues of the stiffness matrices of various nonlinear and elastoplastic problems, [Benzley et al. 1995] reports that, in their study, linear hexahedral meshes are superior to linear tetrahedral meshes. The authors also show that linear hexahedral meshes are slightly superior to quadratic tetrahedral meshes in the nonlinear elastoplastic analysis experiment.

A more recent work, [Tadepalli et al. 2010, 2011], focuses on modeling footwear with a nonlinear incompressible material model under shear force loading conditions. The conclusion of these works is that trilinear hexahedral meshes are superior to linear tetrahedral meshes, and that quadratic tetrahedral elements are computationally more expensive compared to trilinear hexahedral elements, but

<sup>1</sup><https://github.com/polyfem/polyfem/>

<sup>2</sup><https://archive.nyu.edu/handle/2451/44221>

<sup>3</sup><https://github.com/polyfem/tet-vs-hex>

have higher accuracy. [Wang et al. 2004] compares tetrahedral and hexahedral meshes on linear static problems, modal and nonlinear analysis. The study concludes that quadratic tetrahedral and hexahedral elements have similar performance, but quadratic hexahedra are computationally more expensive. The same study also confirms that linear tetrahedra are too stiff for large deformations, and linear hexahedra with large corner angles should be avoided in regions of stress concentration. The study is restricted to a small set of geometries and focuses on manual hexahedral mesh generation. Our study instead focuses on automatic meshing algorithms for both tetrahedral and hexahedral meshes, and we provide experimental results on thousand of complex geometric models and a wide array of elliptic PDEs.

In medical applications, results for femur models [Ramos and Simões 2006] show that linear tetrahedral meshes of the simplified femur model lead to a closer agreement with the theoretical ones, while quadratic hexahedral meshes are more stable and the result is less affected by mesh refinement. On a kidney model, [Bourdin et al. 2007] observes that both linear and quadratic tetrahedral meshes are slightly stiffer than hexahedral meshes, but are more stable when high impact energies are present in the simulation. For heart mechanics and electrophysiology, [Oliveira and Sundnes 2016] notes that quadratic hexahedra are slightly better than quadratic tetrahedra in the mechanics regime, while linear tetrahedral meshes are the best choice for the electrophysiology problem.

## 2.2 Finite Element Analysis Software

There exists a large number of libraries and software for finite-element analysis, both open-source and commercial. An exhaustive comparison of all existing packages<sup>4</sup> is beyond the scope of this paper, therefore we discuss only several representative packages. We point out an interesting project [Ladutenko 2018] attempting to maintain a complete list of FEA packages with a list of characteristics.

Our goal is to investigate and compare the performance of FEM on meshes with tetrahedral and hexahedral elements, using the standard Lagrangian basis functions and serendipity elements commonly used in engineering applications, as well as spline elements used in IGA.

Open-source packages such as FEniCS [Alnæs et al. 2015], GetFEM++ [Renard and Pommier 2018], libMesh [Kirk et al. 2006], and MFEM [MFEM 2020] support both tetrahedral and hexahedral meshes, although very few (e.g., libMesh) implement both the 20-(serendipity) and 27-nodes variant for quadratic hexahedral elements. Deal.II [Alzetta et al. 2018] is another popular open-source FEA library, however it only supports quadrilateral and hexahedral elements. Commercial packages such as ANSYS [ANSYS Inc.

<sup>4</sup>A non-exhaustive list of open-source FEA packages known to the authors include, in alphabetical order, code\_aster [EDP 2018], Deal.II [Alzetta et al. 2018], DOLFIN (FEniCS) [Alnæs et al. 2015], ElmerFEM [Elmer 2018], FEATool Multiphysics (MATLAB) [Ltd. 2019], Feel++ [Prud'homme et al. 2012], FEI (Trilinos) [Heroux et al. 2005], Firedrake [McRae et al. 2016], FreeFEM++ [Hecht 2012], GetDP [Geuzaine 2008], GetFEM++ [Renard and Pommier 2018], libMESH [Kirk et al. 2006], MFEM [MFEM 2020], Nektar++ [Cantwell et al. 2015], NGSolve [Schöberl 2014], OOFEM [Patzák 2012], PolyFEM [Schneider et al. 2019b], Range [Soltys 2019], SOFA [Faure et al. 2012], and VegaFEM [Barbić et al. 2012].

2019], Abaqus [ABAQUS Inc. 2019], COMSOL Multiphysics [COMSOL Inc. 2018] support Lagrangian tetrahedral elements, but surprisingly often implement only serendipity elements for hexahedra [Zienkiewicz et al. 2005, Chapter 6]. Given their popularity, we included serendipity elements in our study in addition to traditional Lagrangian elements.

Another increasingly popular choice of bases for hexahedral meshes are B-splines and NURBS, most commonly used in the context of isogeometric analysis (IGA) [Hughes et al. 2005]. The popularity of spline bases stems from the fact that they have only one dof per element independently of the degree (however, the support of each basis function grows accordingly, and, as a consequence the stiffness matrices become less sparse). Defining this type of element on fully general hexahedral domains is an open problem [Aigner et al. 2009; Li et al. 2013; Martin and Cohen 2010]. Due to their rising popularity, we deem important to include experiments with these elements in our study, but restrict them to cases where a regular lattice mesh is used.

Since none of these libraries implements both Lagrangian (tetrahedral and hexahedral), serendipity, and spline basis functions (hexahedral only) in the same framework, we added all the elements and basis used in this study to our own open-source FEA library [Schneider et al. 2019b] to ensure a fair comparison. PolyFEM [Schneider et al. 2019b] supports all these element types and interfaces with Hypre [Falgout and Yang 2002] and PARDISO [De Coninck et al. 2016; Kourounis et al. 2018; Verbosio et al. 2017] for the solver and Eigen [Guennebaud et al. 2010] for linear algebra.

## 2.3 Meshing

Three-dimensional mesh generation has been thoroughly studied in multiple communities [Carey 1997; Owen 1998; Shewchuk 2012; Tautges 2001]. For the sake of brevity, we restrict our review to the techniques generating pure tetrahedral or pure hexahedral meshes, which are the focus of our study, with an emphasis on methods implemented in readily available open-source or commercial libraries.

*Tetrahedral Meshing.* The most efficient, popular, and well-studied family of algorithms tackles the generation of meshes satisfying the Delaunay condition [Alliez et al. 2005; Boissonnat and Oudot 2005; Chen and Xu 2004; Cheng et al. 2008; Chew 1987, 1993; Cohen-Steiner et al. 2002; Dey and Levine 2008; Du and Wang 2003; Jamin et al. 2015; Murphy et al. 2001; Remacle 2017; Ruppert 1995; Sheehy 2012; Shewchuk 2012, 1996, 1998, 2002; Si 2015; Si and Gärtner 2005; Si and Shewchuk 2014; Tournois et al. 2009]. These methods are robust if the input is a point cloud, but might fail if the boundary of a shape has to be preserved exactly [Hu et al. 2020, 2018].

To overcome these robustness limitations, alternative approaches are based on a background grid [Bridson and Doran 2014; Bronson et al. 2013; Doran et al. 2013; Labelle and Shewchuk 2007; Molino et al. 2003]. The idea is to fill the bounding box of the 3D input surface with either a uniform grid or an adaptive octree, whose convex cells are trivial to tetrahedralize. These methods achieve high quality in the interior of the mesh (where the grid is regular), but introduce badly shaped elements near the boundary, which is often the region of interest in many practical simulations. On the other hand, front-advancing methods [Alauzet and Marcum

2014; Cuillière et al. 2013; Haimes 2014] start by marching from the boundary to the interior, adding one element at a time, pushing the problematic elements into the interior where the advancing fronts meet.

All these methods are unable to handle commonly occurring input surfaces which contain degenerated faces, gaps, and self-intersections. These types of defects are, unfortunately, common in CAD models, due to the NURBS representation (with a fixed degree) not being closed under boolean operations. To the best of our knowledge, the only method that was demonstrated to be capable of handling these cases *robustly* is TetWild [Hu et al. 2018]. It is based on a hybrid numerical representation to ensure correctness, and it allows a small, controlled deviation from the input surface to achieve a good element quality. We used this technique to generate all unstructured tetrahedral meshes in this study.

*Hexahedral Meshing*, aims at filling the volume enclosed by an input surface with hexahedra. Hexahedra also need to have a good shape to ensure good solution approximation. The natural tensor-product structure of a hexahedron enables to define tensor-product bases, and, e.g., use spline-based elements, but dramatically increases the complexity of meshing algorithms. Semi-manual or interactive approaches are usually employed, such as sweeping and advancing front methods [Gao et al. 2016; Livesu et al. 2016; Shepherd and Johnson 2008], which are used in commercial software such as [ANSYS Inc. 2019; Coreform 2020].

By allowing lower element quality, one can design automatic approaches based on regular lattices [Schneiders 1996; Schneiders and Bünten 1995; Su et al. 2004; Zhang et al. 2007; Zhang and Bajaj 2006] or on octrees [Ebeida et al. 2011; Elsheikh and Elsheikh 2014; Ito et al. 2009; Maréchal 2009; Owen et al. 2017; Qian and Zhang 2010; Schneiders et al. 1996; Spatial 2018; Zhang and Bajaj 2006; Zhang et al. 2013].

Polycube methods [Fang et al. 2016; Fu et al. 2016; Gregson et al. 2011; Huang et al. 2014; Li et al. 2013; Livesu et al. 2013; Zhao et al. 2018] and field-aligned parameterization-based methods [Huang et al. 2011; Jiang et al. 2014; Li et al. 2012; Liu et al. 2018; Nieser et al. 2011; Solomon et al. 2017] aim at producing hexahedral meshes with as few irregular edges and vertices as possible, but designing robust algorithms of this type is still an open problem. Sample results from some of the previous methods have been recently collected into a single repository [Bracci et al. 2019], which we use in our study. We also generate a new dataset composed of 3200 hexahedral meshes using the commercial MeshGems-Hexa software [Spatial 2018].

### 3 BACKGROUND

#### 3.1 FEM bases

There is a multitude of different definitions of bases for both tetrahedral (or triangular) and hexahedral (or quadrilateral) element shapes, with different elements tailored to specific types of problems (e.g., axisymmetric elements, shell elements, plasticity elements, etc.). In our comparison, we target the most common choices: we use the standard linear and quadratic Lagrange bases for tetrahedra, which we denote  $P_1$  and  $P_2$  respectively, and hexahedra, with  $Q_1$  denoting linear tensor-product basis and  $Q_2$  quadratic tensor-product basis [Ciarlet 2002b; Szabó and Babuška 1991]. We also use the serendipity

basis [Zienkiewicz et al. 2005], commonly used in commercial software, and spline basis [Hughes et al. 2005] for hexahedral elements. We use the standard Galerkin formulation [Ciarlet 2002b; Szabó and Babuška 1991] with Gaussian quadrature for all our experiments, avoiding non-standard quadrature.

#### 3.2 Mesh and solution characterization

We use the number of vertices as a measure of the resolution of tetrahedral and hexahedral meshes, as the number of vertices is often used by the meshing algorithms as the “budget” that the meshing algorithms can use to create the best possible mesh, and the number of vertices is equal to the number of degrees of freedom in the case of linear (or tri-linear) elements.

In addition to this particular choice, we also investigate other metrics for a specific example (Table 1), and provide an interactive plot that allows one to compare our results using 24 different measures: solution error measured using  $H^1$ ,  $H^1$  semi-norm,  $L_2$ ,  $L^\infty$ ,  $L^\infty$  of gradient, and  $L^8$  norms; mesh average edge length, minimum edge length and number of vertices; the system matrix size and the number of non zero entries, the numbers of basis functions, dofs, elements, and pressure basis functions; timings for loading mesh data, building basis functions, computing the right-hand side, assembling the system matrix, solving the system, computing the errors, total time and time without right-hand side assembly.

#### 3.3 Model PDEs

We selected the following set of representative **elliptic** problems: (1) Poisson; (2) incompressible stationary Stokes fluid flow equations; (3) elasticity with linear Hooke’s law as the constitutive equation; (4) Neo-Hookean elasticity (5) incompressible linear elasticity. We list the corresponding PDEs for completeness.

Let  $\Omega \subset \mathbb{R}^d$ ,  $d \in \{2, 3\}$  be the domain with boundary  $\partial\Omega$ . We aim to solve

$$\begin{aligned}\mathcal{F}(x, u, \nabla u, D^2 u) &= b, \text{ subject to} \\ u &= d \text{ on } \partial\Omega_D \quad \text{and} \quad \nabla u \cdot n = f \text{ on } \partial\Omega_N\end{aligned}$$

for the function

$$u: \Omega \rightarrow \mathbb{R}^n,$$

where  $D^2$  is the matrix of second derivatives,  $b$  is the right-hand side,  $\partial\Omega_D \subset \partial\Omega$  is the part of the boundary with Dirichlet boundary conditions, and  $\partial\Omega_N \subset \partial\Omega$  is the part of the boundary with Neumann boundary conditions. Since we consider second-order PDEs only,  $\partial\Omega_D \cap \partial\Omega_N = \emptyset$ . The form of  $\mathcal{F}$  and the role of  $u$  depends on the specific PDE.

We consider polygonal and polyhedral domains  $\partial\Omega$  (possibly non-convex). The right-hand side  $b$  in our test examples is analytic, the boundary  $d$  is continuous and piecewise-smooth, and  $f$  is piecewise smooth (but possibly with finite-jump discontinuities); under these assumptions, the weak solutions of the equations we consider are (at least) continuous, but the solution derivatives may be singular. We primarily focus on the error in the solution itself, rather than the derivative error, although consider the stress for some elasticity examples. We state the model problems in the strong form, but only the weak solutions exist for many of the test cases.

*Poisson Equation.* This problem is given by

$$\begin{cases} -\Delta u = b & \text{on } \Omega \\ u = d & \text{on } \partial\Omega_D \\ \nabla u \cdot n = f & \text{on } \partial\Omega_N. \end{cases} \quad (1)$$

*Incompressible Steady Stokes Equations.* The Stokes equations provide the relationship between the velocity  $u$  and the pressure  $p$  for an incompressible fluid with viscosity  $\mu$ .

$$\begin{cases} -\mu\Delta u + \nabla p = b & \text{on } \Omega \\ -\nabla \cdot u = 0 & \text{on } \Omega \\ u = d & \text{on } \partial\Omega_D \\ \mu(\nabla u + \nabla^T u) \cdot n - pn = f & \text{on } \partial\Omega_N \end{cases} \quad (2)$$

*Elasticity.* Elasticity PDEs are formulated in terms of the stress tensor  $\sigma[u]$  (which depends on the displacement  $u$ ) as

$$\begin{cases} -\nabla \cdot \sigma[u] = b & \text{on } \Omega \\ u = d & \text{on } \partial\Omega_D \\ \sigma[u] n = f & \text{on } \partial\Omega_N. \end{cases} \quad (3)$$

In this case the right-hand side  $b$  plays the role of a body force, the Dirichlet boundary conditions are fixed displacement, and the Neumann ones are surface tractions.

Material models define how the stress  $\sigma$  is related to the displacement field  $u$ . For the linear Hookean model,

$$\sigma^L[u] = 2\mu\epsilon[u] + \lambda \operatorname{tr} \epsilon[u]I \quad \epsilon[u] = \frac{1}{2}(\nabla u^T + \nabla u), \quad (4)$$

where  $\epsilon[u]$  is the strain tensor,  $\lambda$  is the first Lamé parameter, and  $\mu$  is the shear modulus. There are two common assumptions reducing the elasticity problem to a 2D problem, plane stress and plane strain; in our experiments we are using plane stress. In this case, the elasticity equation has the same form but with different constants [Hughes 2012]:

$$\mu = \frac{E}{2(1+\nu)}, \quad \lambda_{3D} = \frac{Ev}{(1+\nu)(1-2\nu)}, \quad \text{and} \quad \lambda_{2D} = \frac{\nu E}{1-\nu^2}.$$

Incompressible materials form a separate class: in 3D, an isotropic material has Poisson ratio equal to 0.5, and the previous equation is not well-defined, as  $\lambda$  becomes infinite. While isotropic materials in plane stress state cannot have this problem, as the isotropic Poisson ratio cannot exceed 0.5, anisotropic materials can have Poisson ratio 1 for in-plane deformations, and thus can be 2D-incompressible, which geometrically corresponds to the area of the cross-section of a material element preserved under deformations [Lee and Lakes 1997]. As a consequence, equations for 2D-incompressible materials in plane stress state are also of interest. Additionally, when  $\lambda$  grows, the linear system arising from the discretization of the PDE becomes unstable. A common way to avoid such problem is to introduce a Lagrange-multiplier-like function in the form of the pressure  $p$ . This leads to a mixed formulation of elasticity similar to Stokes equations which is stable for large  $\lambda$ s, and reduces to incompressible elasticity

for  $\lambda^{-1} \rightarrow 0$ .

$$\begin{cases} -\nabla \cdot (2\mu\epsilon[u] + pI) = b & \text{on } \Omega \\ \nabla \cdot u - \lambda^{-1}p = 0 & \text{on } \Omega \\ u = d & \text{on } \partial\Omega_D \\ \sigma^N[u] \cdot n = f & \text{on } \partial\Omega_N \end{cases} \quad (5)$$

Finally, in the Neo-Hookean material model the stress is a non-linear function of strain.

$$\sigma[u] = \mu(F[u] - F[u]^{-T}) + \lambda \ln(\det F[u])F[u]^{-T} \quad F[u] = \nabla u + I, \quad (6)$$

where  $F[u]$  is the deformation gradient.

For elasticity problems, we often use the von Mises stresses

$$\begin{aligned} S_{2D}^2 &= \sigma_{0,0}^2 - \sigma_{0,0}\sigma_{1,1} + \sigma_{1,1}^2 + 3\sigma_{0,1}\sigma_{1,0} \\ S_{3D}^2 &= \frac{(\sigma_{0,0} - \sigma_{1,1})^2 + (\sigma_{2,2} - \sigma_{1,1})^2 + (\sigma_{2,2} - \sigma_{0,0})^2}{2} + \\ &\quad 3(\sigma_{0,1}\sigma_{1,0} + \sigma_{2,1}\sigma_{1,2} + \sigma_{2,0}\sigma_{0,2}). \end{aligned} \quad (7)$$

Note that the stresses are discontinuous since they depend on the gradient of the displacement which is only  $C^0$  for our discretizations. To mitigate visual artefacts we average the stresses around vertices in our plots.

### 3.4 Linear Solvers

All FEM problems we consider require to solve a linear system, which, as the mesh size grows, dominates the running time. A vast amount of research has been invested in developing efficient and robust linear solvers. In our study we use two state-of-the-art solvers: Pardiso [De Coninck et al. 2016] a direct solver using the Cholesky factorization, which we use for smaller problems, and Hypre [Falgout and Yang 2002] an algebraic multigrid solver, which we use for larger problems. Direct solvers work particularly well in 2D, but scale poorly for 3D problems. We leave as future work a more detailed study on the effect of the linear solver on the solution time. The conclusions of this study hold for both types of solvers for our experimental setup and test problems.

## 4 COMMON TEST PROBLEMS

We collected a number of standard test cases to cover different physical phenomena and different scenarios: fluid simulation (Section 4.1), linear elastic time dependent (Section 4.2), linear elastic bars (Section 4.3), linear orthotropic material models (Section 4.4), meshes with high aspect-ratio for linear elastic bars (Section 4.5), classical plane with hole with symmetric boundary conditions for compressible and nearly incompressible material (Section 4.6), nearly incompressible linear material (Section 4.7), nonlinear Neo-Hookean material (Section 4.8), and nonlinear Neo-Hookean material with high stresses (Section 4.9).

Most of the solution domains are chosen to simplify manual creation of hexahedral meshes: the simulations will be performed on an unstructured tetrahedral mesh and a nearly regular lattice with the same number of vertices. Experiments in Sections 4.2 to 4.7 are run on a MacBook Pro 3.1GHz Intel Core i7, 16GB of RAM, and 8 threads. Experiments in Sections 4.8 and 4.9 are run on a cluster node with 2 Xeon E5-2690v4 2.6GHz CPUs and 250GB memory, each with max 128GB of reserved memory and 8 threads. For all

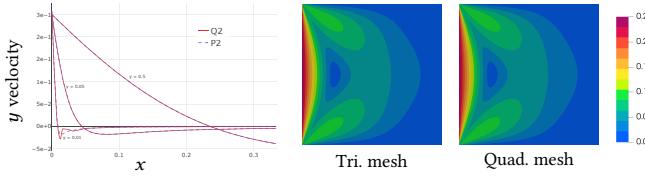


Fig. 1. The velocity magnitude for a Stokes problem discretized with mixed elements. The plot shows the velocity in  $y$ -direction along horizontal lines  $y = 0.01, 0.05, \text{ and } 0.5$  parametrized by  $x$ .

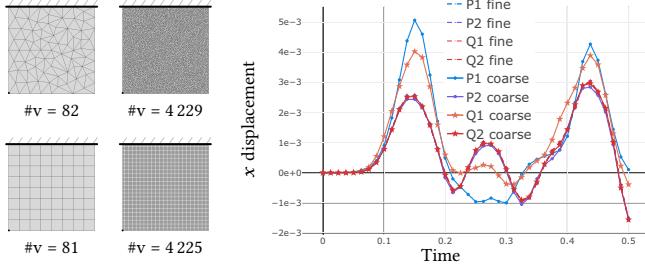


Fig. 2.  $x$  displacement of the bottom-left corner (black dot) of a unit square.

experiments, we use the PolyFEM library [Schneider et al. 2019b], which uses the Pardiso [De Coninck et al. 2016; Kourounis et al. 2018; Verbosio et al. 2017] direct solver, and Newton iterations for the nonlinear problems.

Note that, for completeness, we also validated PolyFEM on the example in Figure 3 for linear and quadratic tetrahedra and serendipity hexahedra on Hooke material against Abaqus. The results are identical up to numerical precision.

#### 4.1 Incompressible Stokes

We use a planar square domain mesh with 4 229 vertices for the triangle mesh and 4 225 vertices for the regular grid. We simulate the Stokesian fluid (2) with viscosity  $\mu = 1$  in the standard “driven cavity” example: the fluid has zero boundary conditions on 3 of the 4 sides and a tangential velocity of 0.25 on the left side. Figure 1 shows the results for mixed linear (for the pressure) and quadratic (for the velocity) elements: the results are indistinguishable between hexahedral and tetrahedral elements.

#### 4.2 Time-Dependent Linear Elasticity

We consider the dynamics of a suspended object under gravity: we fix the top part of a unit square with material parameters  $E = 200$  and  $\nu = 0.35$  and apply a constant body force of 20 in the  $y$  direction. We integrate the dynamic simulation for  $t$  from 0 to 0.5 with 40 time steps integrated with Newmark [Newmark 1959]. We mesh the domain at a coarse and fine resolution, both for triangles and for quads. Figure 2 shows the displacement in the  $x$  direction of the bottom left corner for the 4 discretizations, using linear and quadratic elements.

	$t_b$	$t_a$	$t_s$	$t$	$e_f$	
$P_1$	8.07e-3	1.88e-2	5.60e-2	8.29e-2	6.14e-3	Tet. mesh
$P_2$	2.30e-2	1.80e-1	3.43e-1	5.47e-1	9.19e-5	
$Q_1$	<b>5.96e-3</b>	3.36e-2	6.39e-2	1.03e-1	1.27e-3	
$Q_2$	1.46e-2	4.61e-1	4.34e-1	9.10e-1	<b>4.66e-5</b>	Hex. mesh

Fig. 3. Displacement error in the  $y$  displacement of the moving endpoint compared with a dense solution for a unit force applied at the endpoint of a beam with a square cross-section. The times are averaged over 10 runs per force sample.

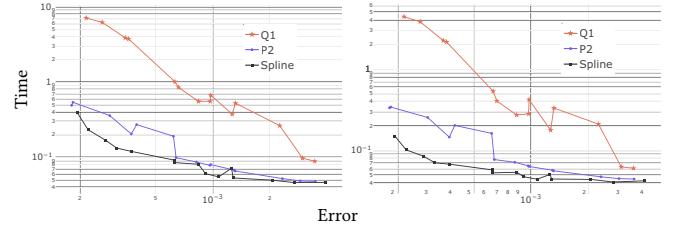


Fig. 4. Time vs. error (with respect to a dense solution) (total time on the left, and solve time only on the right) for  $P_2$ ,  $Q_1$ , and quadratic spline elements.

#### 4.3 Transversally Loaded Beam

In this experiment, we consider beams with different cross-sections (square, circular, and I-like) in the  $xy$ -plane of length  $L$ . The beam is fixed (i.e., zero Dirichlet conditions are applied) at the end ( $z = L$ ), and different tangential forces  $f = [0, -f_y, 0]^T$ ,  $f_y \in [-0.1, -2]$ , are applied at  $z = 0$ , opposite to the fixed side. The rest of the boundary is left free and we do not apply any body force. For these experiments we use linear isotropic material model (4) with Young’s modulus  $E = 210\,000$  and Poisson’s ratio  $\nu = 0.3$ . We study the displacement at the bottom corner of the moving end ( $z = 0$ ) in the  $y$  direction and compare it with a dense solution to compute the error  $e$  (note that the solution is singular only at  $z = L$ , far from the evaluation points). We report as  $e_f$  the slope of the linear fit of the error as a function of the force magnitude. We also report the basis construction time  $t_b$ , assembly time  $t_a$ , solve time  $t_s$ , and total time  $t$ . Note that all the timings reported are averaged over 10 different runs per force sample.

*Square Cross-section.* For running the simulation, we use a square cross-section of side  $s = 20$ , length  $L = 100$  and mesh it with a tetrahedral mesh with 739 vertices and a hexahedral mesh (regular grid) with 750 vertices. Figure 3 shows the errors compared with the dense solution, where trilinear hexahedral elements outperform linear tetrahedral elements but the quadratic counterparts are indistinguishable. Timing-wise, the quadratic tetrahedra are slightly better.

We created a sequence of hexahedral and tetrahedral meshes with similar errors for a force  $f = [0, -2, 0]^T$ . Figure 4 shows that for a given error,  $P_2$  discretization is around four times faster than  $Q_1$ , and SPLINE<sub>2</sub> have a slight advantage over  $P_2$ . Note that both  $Q_1$  and SPLINE<sub>2</sub> are constructed over a perfectly regular grid, while the  $P_2$  elements are defined over an unstructured tetrahedral mesh.

Finally, we created a sequence of hexahedral meshes that matches total time, total memory, total number of degrees of freedom, and

	time (s)	memory (MB)	DOF	error
$P_1/Q_1$	time	1.01 / 0.98	125 / 132	8,258 / 6,413 1.98e-03 / 6.93e-04
	memory	1.01 / 0.91	125 / 125	8,258 / 6,050 1.98e-03 / 7.49e-04
	DOF	1.01 / 1.07	125 / 149	8,258 / 7,139 1.98e-03 / 6.06e-04
	error	1.01 / 0.12	125 / 18	8,258 / 1,224 1.98e-03 / 1.87e-03
$P_2/Q_2$	time	16.86 / 17.56	2,236 / 2,033	59,885 / 44,541 1.85e-05 / 1.24e-05
	memory	16.86 / 18.77	2,236 / 2,241	59,885 / 48,951 1.85e-05 / 8.40e-06
	DOF	16.86 / 24.44	2,236 / 2,988	59,885 / 59,777 1.85e-05 / 5.43e-06
	error	16.86 / 11.22	2,236 / 1,451	59,885 / 35,017 1.85e-05 / 1.70e-05
$P_2/Q_1$	time	16.86 / 16.74	2,236 / 2,630	59,885 / 58,719 1.85e-05 / 1.58e-04
	memory	16.86 / 14.26	2,236 / 2,226	59,885 / 52,272 1.85e-05 / 1.70e-04
	DOF	16.86 / 17.52	2,236 / 2,669	59,885 / 59,777 1.85e-05 / 1.54e-04
	error	16.86 / 170.29	2,236 / 11,805	59,885 / 180,774 1.85e-05 / 6.36e-05

Table 1. Comparison of performance of tetrahedral and hexahedral elements on several measures: time, memory, DOF and error, with one of the measures matched (marked in gray): for the first row of each comparison, we match time, second memory, etc. The best-performing (according to each measure) element is shown in green. For instance, by comparing  $P_2$  with  $Q_1$  for the same error (last section of the table),  $P_2$  is faster (first column), it uses less memory (second column), and it has less DOFs (third column).

	$t_b$	$t_a$	$t_s$	$t$	$e_f$	Tet. mesh	Hex. mesh
$P_1$	3.52e-2	7.52e-2	1.21e-1	2.31e-1	3.50e-3		
$P_2$	9.88e-2	8.58e-1	1.79	2.75	5.21e-5		
$Q_1$	2.22e-2	1.03e-1	1.76e-1	3.02e-1	9.82e-4		
$Q_2$	5.78e-2	1.71	2.77	4.54	8.38e-5		

Fig. 5. Displacement error with respect to a dense solution per force unit at the endpoint of a beam with a circular cross-section. The times are averaged over 10 runs per force sample.

	$t_b$	$t_a$	$t_s$	$t$	$e_f$	Tet. mesh	Hex. mesh
$P_1$	9.29e-2	1.99e-1	2.67e-1	5.58e-1	1.85e-3		
$P_2$	2.68e-1	1.94	4.59	6.80	7.71e-5		
$Q_1$	6.16e-2	3.12e-1	5.28e-1	9.01e-1	6.77e-4		
$Q_2$	1.54e-1	4.58	9.63	1.44e1	1.05e-4		

Fig. 6. Displacement errors for a unit force applied at the endpoint of an I-beam. The times are averaged over 10 runs per force sample.

error of the tetrahedral mesh in Figure 3 for both linear and quadratic elements. Table 1 summarizes our findings:  $P_1$  is significantly worse than  $Q_1$  but the two quadratic discretizations produce similar results.  $P_2$  overall performs better than  $Q_1$ .

*Circular Cross-section.* We consider a beam of length  $L = 100$  with a circular cross-section of diameter  $d = 20$ . We created a tetrahedral mesh with 2 252 vertices and a hexahedral mesh with 2 288 vertices (note that in this case the mesh is not a regular grid anymore), by extruding a quad mesh generated with [Jakob et al. 2015]. Figure 5 shows similar  $y$ -displacement errors as for the square cross-section,  $P_1$  produces low-quality results, while  $P_2$  and  $Q_2$  are similar.

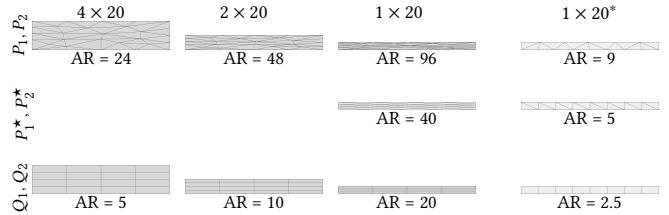
*I-beam Cross-section.* We use an I-beam (the bounding box of the cross-section is  $125 \times 154$ ) of length  $L = 473.11$ . The tetrahedral mesh has 6 102 while the hexahedral mesh, generated by extruding a quad mesh, has 6 080 vertices, results are shown in Figure 6.

#### 4.4 Orthotropic Material

We repeated the previous experiment using linear orthotropic material parameters (carbon fiber). The material parameters are obtained from [Pardini and Gregori 2010]. Three Young moduli are 167, 33,

	$t_b$	$t_a$	$t_s$	$t$	$e_f$	Tet. mesh	Hex. mesh
$P_1$	8.00e-3	2.87e-1	2.11e-2	3.16e-1	2.47		
$P_2$	2.48e-2	1.71	3.37e-1	2.08	4.81e-2		
$Q_1$	6.41e-3	9.65e-1	3.40e-2	1.01	1.35		
$Q_2$	1.66e-2	9.80	4.73e-1	1.03e1	2.17e-2		

Fig. 7. Displacement error (compared with  $P_4$ ) for a force of magnitude  $1e-5$  applied at the endpoint of a beam with orthotropic material and a square cross-section. The times are averaged over 10 runs per force sample.



	$t_b$	$t_a$	$t_s$	$t$	$e_f$	
$4 \times 20$	$P_1$	8.78e-3	1.96e-2	5.72e-2	8.56e-2	9.91e-1
	$P_2$	2.33e-2	1.76e-1	3.42e-1	5.41e-1	5.78e-3
	$Q_1$	5.70e-3	3.17e-2	6.34e-2	1.01e-1	2.87e-1
	$Q_2$	1.51e-2	4.88e-1	4.50e-1	9.53e-1	1.60e-3
$2 \times 20$	$P_1$	8.25e-3	1.91e-2	5.74e-2	8.47e-2	4.58
	$P_2$	2.30e-2	1.71e-1	3.27e-1	5.21e-1	8.56e-2
	$Q_1$	5.69e-3	3.10e-2	6.29e-2	9.95e-2	2.54
	$Q_2$	1.48e-2	4.66e-1	4.33e-1	9.14e-1	8.40e-3
$1 \times 20$	$P_1$	8.94e-3	1.93e-2	5.87e-2	8.69e-2	1.86e1
	$P_2$	2.25e-2	1.72e-1	3.31e-1	5.25e-1	1.65
	$Q_1$	5.85e-3	3.26e-2	6.53e-2	1.04e-1	1.53e1
	$Q_2$	1.47e-2	5.03e-1	4.67e-1	9.85e-1	5.58e-2
$1 \times 20^*$	$P_1^*$	6.31e-3	1.50e-2	5.54e-2	7.67e-2	1.69e1
	$P_2^*$	1.85e-2	1.31e-1	3.74e-1	5.24e-1	1.03
	$P_1$	5.15e-3	1.40e-2	5.39e-2	7.30e-2	1.70e1
	$P_2$	1.55e-2	1.18e-1	2.28e-1	3.62e-1	6.54e-2
$1 \times 20^*$	$Q_1$	3.90e-3	2.47e-2	6.56e-2	9.42e-2	1.33e1
	$Q_2$	9.50e-3	3.43e-1	3.10e-1	6.62e-1	2.96e-2
	$P_2^*$	4.68e-3	1.24e-2	6.29e-2	8.00e-2	1.67e1
	$P_2^*$	1.42e-2	1.09e-1	3.14e-1	4.37e-1	1.17e-1

Fig. 8. Displacement errors with respect to a  $P_4$  solution for a unit force applied at the endpoint for different aspect ratios. The aspect ratio  $1 \times 20^*$  is the same domain as  $1 \times 20$  remeshed with optimized element aspect ratio. The results  $P_1^*$  and  $P_2^*$  are obtained by splitting the hexahedra into 5 tetrahedra.

and 33, The Poisson ratios are 0.18, 0.25, and 0.18, shear moduli are 13, 21, and 21. Figure 7 shows that the  $y$ -displacement error with respect to different discretizations has the same behavior as for isotropic materials (Section 4.3).

#### 4.5 High Aspect-Ratio

To analyze the effect of using elements with high aspect ratio, we repeated the previous experiment for 3 different domains by shrinking the height of the square cross-section from 20, to 4, 2, and 1, while keeping the connectivity identical. This procedure introduces artificial high aspect-ratio elements. To obtain the same anisotropy measure for hexahedral and tetrahedral meshes, we define the aspect ratio between the largest and smallest eigenvalue of the covariance matrix of its vertices. Figure 8 shows the error in the  $y$ -direction displacement per unit of force for the hexahedral and tetrahedral meshes. The tetrahedral mesh suffers from the low element quality much more than the hexahedral mesh. However, if we regenerate the meshes for the thin domain with the same number of degrees

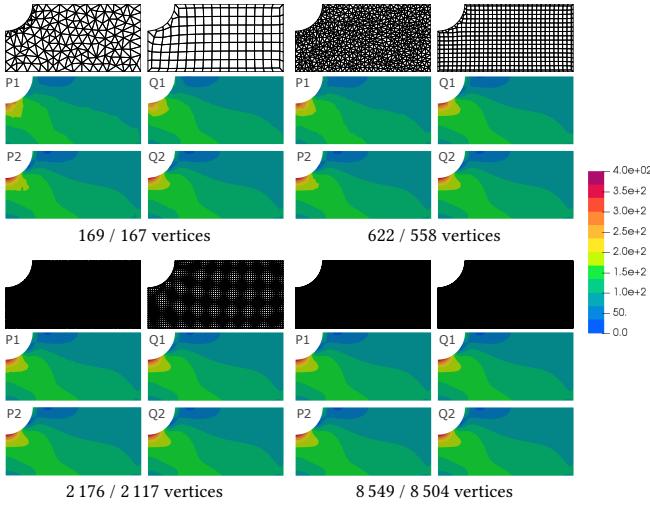


Fig. 9. Visualization of the von Mises stresses for four different mesh resolutions. Each figure shows  $P_1$  (top left),  $P_2$  (bottom left),  $Q_1$  (top right), and  $Q_2$  (bottom right). The numbers below the figures represent the number of vertices of the tri / quad-mesh.

of freedom, but with element quality optimization (last row in Figure 8), the high error of  $P_2$  disappears and the errors are similar as for  $Q_2$ , as shown in last four rows in Figure 8. For comparison, we also generate a tetrahedral mesh by simply splitting the hexahedra into six tetrahedra. Note that this kind of aspect ratios are extreme and do not appear in any *automatically* meshed model in our data set (Figure 14).

#### 4.6 2D Domain with a Hole

Another commonly used test problem is a 2D domain with a hole in the middle. For our experiments we use a square domain of size  $200 \times 100$  with a hole in the center of radius 20, the same material model (linear elasticity (4)) and same material parameters  $E = 210\,000$  and  $\nu = 0.3$ . The experiment consists of applying an opposite in-plane force on the left and right boundary of 100, that is, stretching the plane horizontally. This problem is obviously ill-posed because of the lack of Dirichlet boundary conditions. We use a standard approach to eliminate the null-space of solutions by exploiting symmetry, and simulating on a quarter of the domain. This leads to a domain with a “corner” cut with two symmetric boundary Dirichlet conditions (displacement is constrained only in the orthogonal direction), a zero Neumann condition, and a Neumann condition corresponding to the original force. We solve this particular benchmark problem on four meshes with different resolutions. Figure 9 shows the von Mises stresses (7) on the top for a triangle mesh and bottom for a quadrilateral mesh, left linear and right quadratic elements. As expected, for a sufficiently dense mesh, all methods converge to similar results. The interesting result is that  $Q_2$  elements produce visually better results even at really low resolution (first image and second image). In contrast, for linear triangular elements, we need to increase the mesh resolution up to 8 500 vertices (last image) for the artifacts to disappear.



Fig. 10. Displacement norm for a nearly incompressible 2D domain with a hole for a mesh with 8 549 / 8 504 vertices.

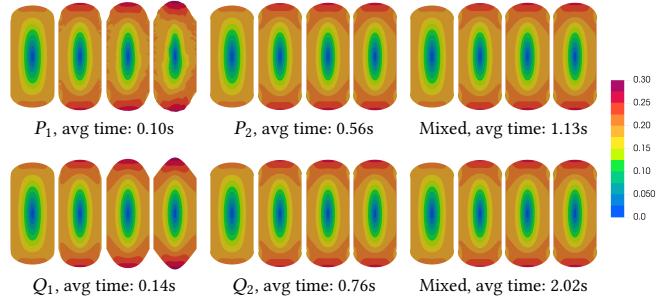


Fig. 11. Visualization of the norm of the displacement for a compressed square, with  $v = 0.9, 0.99, 0.999, 0.9999$ , left to right, for different elements and discretizations.

This particular problem is also a standard benchmark for incompressible material simulation. We performed the same experiment for a nearly incompressible material:  $E = 0.1$  and  $\nu = 0.9999$ . Figure 10 shows the norm of the displacement: as for the compressible case,  $P_2$  and  $Q_2$  have a similar behavior. Interestingly, for this case,  $Q_1$  produces a very different solution.

#### 4.7 Nearly Incompressible Material

For the last linear benchmark, we compared the performance of the four discretizations with the material approaching incompressibility. We apply a boundary displacement  $[0.2, 0]$  on the left and  $[-0.2, 0]$  on the right of a unit square. We perform a series of experiments in which we keep the Young’s modulus fixed at 0.1 while changing the Poisson’s ratio from 0.9 to 0.9999 (1 being the limit of incompressibility in 2D, i.e., area preservation). We compare the standard formulation (4) with a mixed formulation (5) that does not become unstable as  $\nu \rightarrow 1$ . Note that since mixed formulations require different basis degrees for the displacement and the pressure. We performed our experiments using linear pressure bases and quadratic bases for the displacements. We mesh the square with a quad mesh with 4 225 vertices and a tri mesh with 4 229 vertices.

Figure 11 shows the norm of the displacement for this series of experiments. For the nearly incompressible regime (i.e.,  $\nu = 0.9999$ ) it is remarkable that the quadrilateral element discretization leads to a symmetric and smooth (but incorrect) result for the linear case, while the triangular elements producing an unstable output. The two quadratic discretizations produce visually similar results, close to those obtained with the stable mixed method. The only quantitative difference is that the residual error for the direct solver drops from  $1e-15$  (numerical zero) to  $1e-12$ , indicating that the system is close to singular.

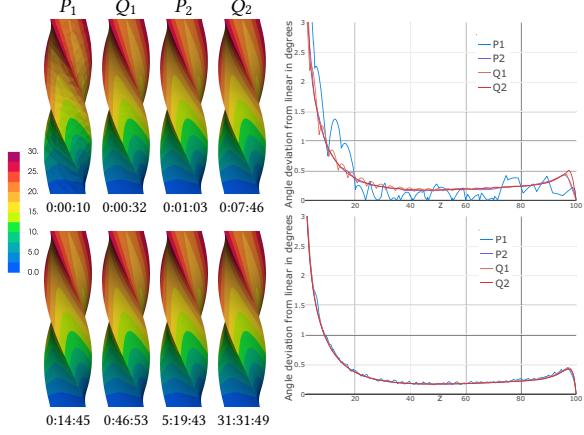


Fig. 12. Nonlinear elastic deformation. Left: the deformed mesh color coded by the norm of the displacement and the running time. Right: the angle of rotation of the cross-section deviation from linearly interpolated along the depth of the bar. We show the deviation from linear interpolation to make the differences between different elements more visible. Note that the linear interpolation is not the exact solution so we do not expect the line to go to zero.

#### 4.8 Beam with Torsional Loads

We now compare the solutions for the Neo-Hookean (6) material model for our discretizations. We take a beam with a cross-section  $[-10, 10]^2$  and length 100,  $E = 200$   $\text{nu} = 0.35$ , fix the bottom part and apply a rotation of 90 degrees to the top. The rest of the surface is left free. To avoid ambiguities in the rotation we use five steps of incremental loading in the Newton solver. We run the experiment on two sets of tetrahedral and hexahedral meshes. Coarse meshes have 739 and 750 while the dense have 50 000 and 58 719 vertices respectively. The first three images of Figure 12 show that the results are indistinguishable, except for small numerical fluctuations. Similar results are for the plot (Figure 12 last plot) of the rotation angle along a line starting at the point  $[9.5, 9.5, 0]$  parallel to the beam axis. Note that the dense  $Q_2$  solution required more than 44GB of RAM for the solver, while  $P_2$  required around 21GB.

The reason for the high memory consumption is the size of the element matrix, which has  $(27 \times 3)^2 = 6561$  entries (compared to 144 for  $P_1$ , 576 for  $Q_1$ , and 900 for  $P_2$ ). Note that the difference in running time does not come from the number of iterations of the Newton solver: for  $P_1, Q_1, P_2, Q_2$  we obtained 16, 17, 20, 17 iterations respectively for the coarse mesh, and 18, 16, 17, 18 for the dense mesh.

We have repeated this experiment using quadratic B-spline bases on the coarse mesh. The result is similar to  $P_2$  and  $Q_2$ , see the inset figure. For this particular example, we measure the solve time of the three discretizations: the spline solve is 3 times faster than  $P_2$  (0.51s versus 1.50s) and 9 times faster than  $Q_2$  (0.51s vs 4.62s) while having roughly the same number of iterations: 16. Note that the assembly time (using full integration which could be improved using [Schillinger et al. 2014]) for spline is similar to  $Q_2$  and is 12 times slower than  $P_2$  (20.54s versus 1.70s). While using splines on regular

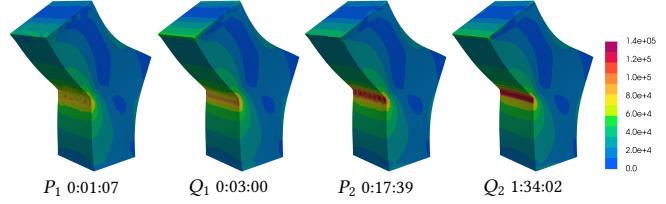


Fig. 13. Von Mises stress and singular solution timings for the four discretizations for the Neo-Hookean material model. The stresses are averaged around vertices (including the vertices where the solution is singular).

grids is natural, the extension to irregular meshes requires the use of T- or U- splines [Beer et al. 2015; Wei et al. 2018], increasing the implementation complexity.

#### 4.9 High Stress

As a final experiment, we run a simulation for an L-shaped domain with the Neo-Hookean material and  $E = 210000$   $\nu = 0.3$ . Our goal is to study the differences in the stresses for singular solutions: the concave corner of L will have a stress singularity. We mesh our domains with 14 155 vertices for the tetrahedral mesh and 14 161 vertices for the hexahedral mesh. We fixed the bottom part of the domain (zero displacement) and rotate the top part by 120 degrees (Dirichlet constraint on the displacement), the rest of the boundary is let free (zero Neumann condition). Figure 13 shows that linear tetrahedral elements underestimate the stress while linear hexahedral elements are somewhat better. The quadratic discretizations are qualitatively similar: the hexahedral mesh does not have the spurious small stress oscillations of  $P_2$  because the elements are aligned with the mesh, however the price to pay is significant, 17 minutes for  $P_2$  compared to more than 1.5 hours for  $Q_2$ .

### 5 LARGE DATASET

Next, to evaluate the performance of different types of elements on a large diverse set of realistic domains, we compute solutions for the Poisson (1) and linear elasticity (4). We use the *method of manufactured solutions* [Salari and Knupp 2000], that is, for an analytically defined solution  $u$  we compute the corresponding right-hand side  $b$  by plugging it into the PDE. The boundary condition  $d$  is obtained by sampling  $u$  on the boundary. For the Poisson equation, we use the Franke function [Cavoretto et al. 2018; Franke 1979]

$$\begin{aligned} u(x_1, x_2, x_3) = & \\ & 3/4 e^{-(9x_1-2)^2 + (9x_2-2)^2 + (9x_3-2)^2}/4 \\ & + 3/4 e^{-(9x_1+1)^2/49 - (9x_2+1)/10 - (9x_3+1)/10} \\ & + 1/2 e^{-(9x_1-7)^2 + (9x_2-3)^2 + (9x_3-5)^2}/4 \\ & - 1/5 e^{-(9x_1-4)^2 - (9x_2-7)^2 - (9x_3-5)^2}, \end{aligned}$$

while for elasticity

$$u(x_1, x_2, x_3) = \frac{1}{80} \begin{pmatrix} x_1 x_2 + x_1^2 + x_2^2 + 6x_3 \\ x_1 x_3 - x_3^3 + x_1 x_2^2 + 3x_1^4 \\ x_1 x_2 x_3 + x_2^2 x_3^2 - 2x_1, \end{pmatrix}$$

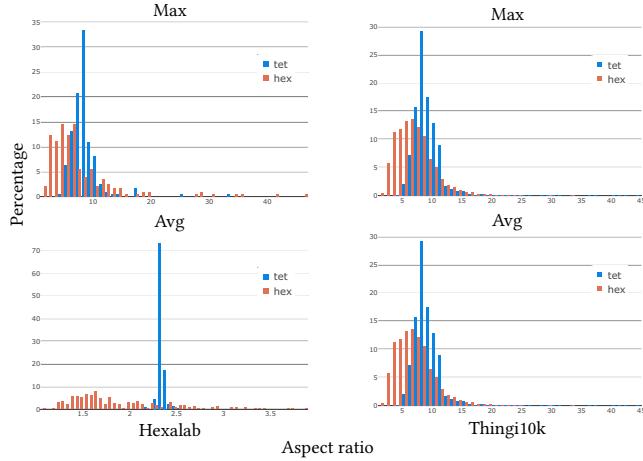


Fig. 14. Histogram of the maximal and mean aspect ratios for the Hexalab (left) and Thingi10k (right) datasets.

with Lamè parameters  $E = 200$  and  $\nu = 0.35$ . In addition to standard tensor product bases for hexahedra, we compare to the popular *serendipity* bases [Zienkiewicz et al. 2005][Chapter 6], which have only 20 nodes per element instead of 27.

We use two sources for our data: (1) the Hexalab dataset containing results of 16 state-of-the-art hexahedral meshing techniques [Bracci et al. 2019], (2) the Thingi10k dataset [Zhou and Jacobson 2016] consisting of triangulated surfaces. For each dataset, we produce a tetrahedral mesh dataset from the surfaces of the hexahedral meshes (generated with MeshGems [Spatial 2018]) using TetWild [Hu et al. 2018] with a matching number of vertices. Note that since matching the number of vertices is a heuristic process, we discard all meshes where the difference in the number of vertices is larger than 5% of the total number of vertices. To ensure that we are solving a similar problem on the two tessellations we remove meshes whose Hausdorff distance between the surfaces of corresponding hexahedral and tetrahedral meshes differs more than  $10^{-3}$  of the diagonal of the bounding box of the hexahedral mesh surfaces. Finally, we discard all meshes whose ratio between boundary and total vertices is more than 30%. Since the Hexalab dataset is small, we opted for doing one step of uniform refinement to increase the number of interior vertices instead of discarding them. In summary, the two datasets are:

- (1) 237 Hexalab hexahedral meshes and 237 tetrahedral meshes generated with Tetwild.
- (2) 3 200 hexahedral meshes generated with MeshGems [Spatial 2018] and 3 200 tetrahedral meshes generated with Tetwild both obtained from the surfaces in the Thingi10k dataset.

For conciseness, we report only the most significant results. Many other metrics (e.g.,  $H^1$ -error, the time required to assemble bases, nonzero entries of the matrix, etc.) can be found in the interactive plot.

We remark that, while Tetwild *guarantees* to produce valid tetrahedral meshes, Meshgems and the methods used in the Hexalab dataset do not provide any guarantee. We observe that out of the

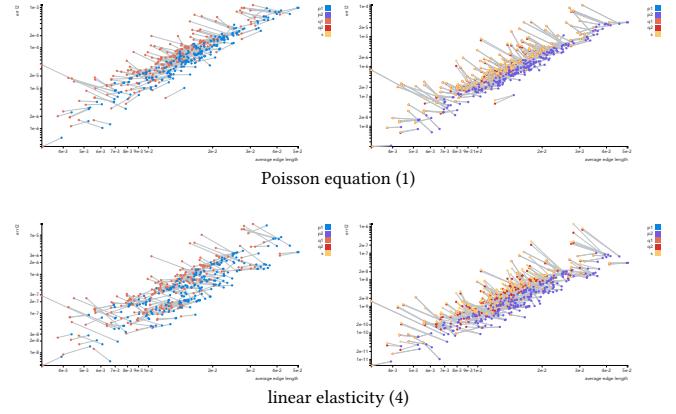


Fig. 15.  $L_2$  error vs. average mesh size for the Hexalab dataset for linear (left) and quadratic (right) elements. The lines connect two points belonging to the same model.

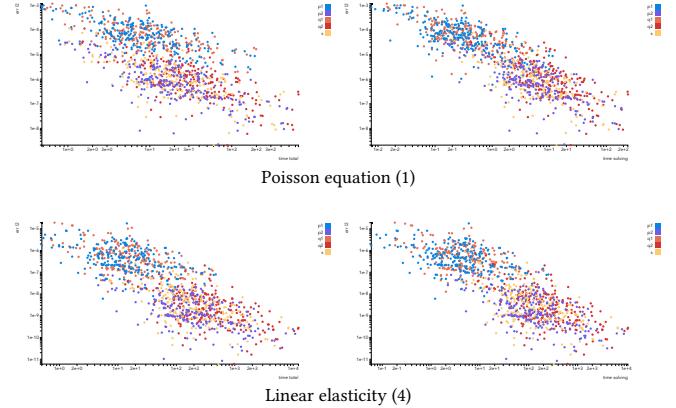


Fig. 16.  $L_2$  error vs. total time (left) and solver time (right) for the Hexalab dataset.

237 Hexalab meshes, 8 (3.4%) contain at least one inverted element (2 from [Livesu et al. 2016] and 6 from [Gao et al. 2016]). For the Thingi10k dataset, Meshgems produces 577 (18.0%) meshes with at least one invalid element. To check if a hexahedron has a negative volume we sample it with  $10^3$  uniformly spaced samples, evaluate the Jacobian at each point, and mark it as flipped if at least one evaluation is negative. Another important quality measure is the aspect ratio of the elements (Section 4.5). Figure 14 shows that both our datasets contain reasonably well-shaped elements.

All experiments are run on a cluster node with 2 Xeon E5-2690v4 2.6GHz CPUs and 250GB memory, each with max 128GB of reserved memory and 8 threads. For all experiments we use the Hypre [Falgout and Yang 2002] algebraic multigrid iterative solver and the PolyFEM library for the finite element assembly.

*Hexalab.* To avoid clutter in the plots we omit the results obtained from meshes with inverted elements leading to plots with 229 points. For the complete statistics see the interactive plot. We first compare the error of the method with respect to the average edge length,

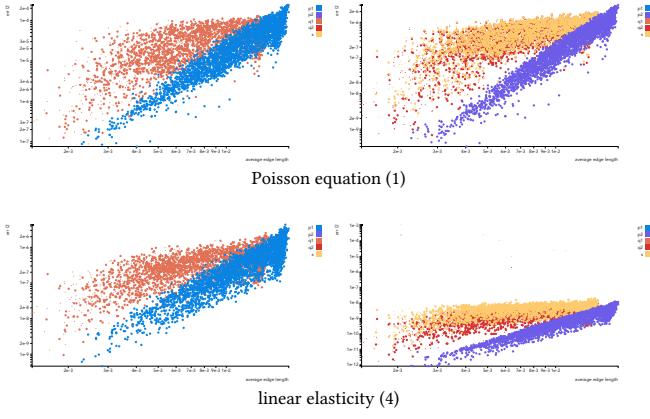


Fig. 17.  $L_2$  error vs. average mesh size for linear (left) and quadratic (right) elements. The smaller dot sizes indicate models with inverted elements.

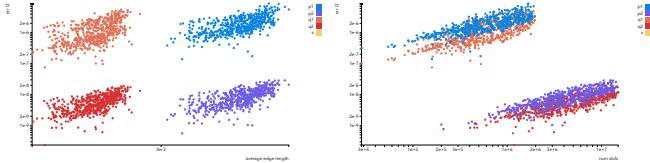


Fig. 18. The  $L_2$  error vs. average edge length (left) and number of degrees of freedom (right) for the Poisson equation (1) on 580 “uniform” hexahedral meshes.

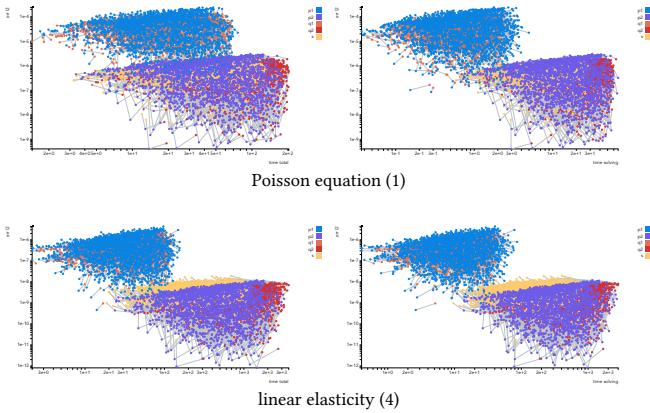


Fig. 19. Total time (left) and solve time (right) vs. the  $L_2$  error.

Figure 15. We confirm the results of Section 4 for the state-of-the-art hexahedral meshing methods; the accuracy of the solution on a hexahedral and tetrahedral mesh is comparable, in our experiments, for both Poisson and linear elasticity. Figure 16 shows the total and solve time required to reach a certain error, where we draw the same conclusions: the results of the four discretizations are similar. The plots show, as expected, that for a given mesh serendipity elements are faster but less accurate than  $Q_2$  elements. However this advantage is not consistent enough to change the conclusion

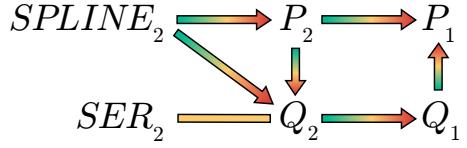


Fig. 20. The arrows indicate which method is inferior (red side); the yellow box indicates that the methods are comparable.

related to quadratic tetrahedral elements. Statistics for the individual hexahedral meshing method are available in the interactive plot.

*Thingi10k*. We repeated the same experiment on 3 200 hexahedral meshes generated with MeshGems. For this large dataset it is interesting to note that qualitative behavior of the edge length vs. error curve (Figure 17) is different between hexahedra and tetrahedra: the curve for tetrahedral elements exhibit the expected convergence, while the curve for hexahedra is more flat. This effect comes from the fact that MeshGems is an octree-based method with a tendency to create highly anisotropic meshes. This effect can be mitigated by limiting the difference between the minimal and maximal refinement levels in the octree used to construct the mesh. This leads to more uniform element size, and the results become similar to the results for tetrahedral meshes and the Hexalab dataset, Figure 18.

We also compared running and solve times (Figure 19) and, as expected, serendipity elements are faster than  $Q_2$  elements but have a larger error. Tetrahedral elements are between the two hexahedral elements: their accuracy is similar to  $Q_2$  and a running/solving time similar to serendipity.

## 6 DISCUSSION AND CONCLUSIONS

We presented a large-scale, quantitative study of several common types of finite elements applied to five elliptic PDEs. Our results are consistent on all elliptic PDEs we tried.

We summarize our findings in Figure 20, which allows us to draw the following conclusions for the five elliptic PDEs we considered in our study:

- (1) Consistently with well-known observations,  $P_1$  elements are less efficient (more time spent to obtain a solution with given accuracy) than all other options in all our experiments (Sections 4 and 5).
- (2)  $Q_2$  elements are slightly more accurate than quadratic serendipity elements  $SER_2$  but are slightly more expensive for a fixed mesh (Section 5).
- (3)  $P_2$  elements are generally more efficient than  $P_1$ ,  $Q_1$ ,  $Q_2$ ,  $SER_2$ , that is, we can obtain a given target error in less time, if we can chose the mesh resolution optimal for the desired error level. We were not able to identify any disadvantages for these elements for the range of problems and geometries we have considered (Sections 4 and 5).
- (4) Quadratic spline elements  $SPLINE_2$  (on a regular lattice) are more efficient than  $Q_2$  elements (Section 4.8).  $SPLINE_2$  are also more efficient compared to  $P_2$  (3x faster solving time for the same accuracy) but with a much longer assembly time (12 times slower, which could be reduced with more advanced

integration techniques [Schillinger et al. 2014]). Their use, however, is restricted by the current meshing technology, as they require meshes with regular grid structure almost everywhere for optimal performance. When these elements are mixed with standard  $Q_2$  elements to handle general hexahedral meshes with singular vertices and edges [Schneider et al. 2019a; Wei et al. 2018], their performance advantage is considerably reduced.

For the five elliptic PDEs we considered, unstructured tetrahedral meshes with quadratic Lagrangian basis are a good choice for a “black-box” analysis pipeline: robust tetrahedral meshing algorithms that can process thousands of real-world models exist [Hu et al. 2018], and  $p$ -refinement can be used to compensate for the rare badly shaped triangles introduced by the meshing algorithms [Schneider et al. 2018].

We leave the extension of this study to non-elliptic PDEs, multi-physics, and collision response as future work. Another important potential extension is the study of bases with orders higher than 2, as is typically the case in IGA setting, or an extension to spectral elements. Another venue for future work is to analyze the impact of the existing different per-element optimizations (e.g., reduced quadrature, hourglass control, special quadrature rules that exploit the tensor-product structure of  $Q$  elements, etc.). However, we note that these different optimizations will mostly impact the performance of the assembly and will have little influence on the solve time, which dominates the runtime for sufficiently large problems.

Finally, we acknowledge that the choice of elements is only one of the sources of error in numerical simulations; in realistic scenarios, material models, boundary conditions, or domain shape also play role in the accuracy of a simulation. Extending our study to account for these sources of error is an interesting venue for future work.

## ACKNOWLEDGMENTS

We thank the NYU IT High Performance Computing for resources, services, and staff expertise. This work was partially supported by the NSF CAREER award under Grant No. 1652515, the NSF grants OAC-1835712, OIA-1937043, CHS-1908767, CHS-1901091, DMS-1436591, NSERC DGECR-2021-00461 and RGPIN-2021-03707, the SNSF grant P2TIP2\_175859, a Sloan Fellowship, a gift from Adobe Research, a gift from nTopology, and a gift from Advanced Micro Devices, Inc.

## REFERENCES

- ABAQUS Inc. 2019. Abaqus FEA. <http://www.simulia.com>.
- M. Aigner, C. Heinrich, B. Jüttler, E. Pilgerstorfer, B. Simeon, and A. V. Vuong. 2009. Swept Volume Parameterization for Isogeometric Analysis. In *Mathematics of Surfaces XIII*. Springer Berlin Heidelberg, 19–44.
- F. Alauzet and D. Marcum. 2014. A Closed Advancing-Layer Method With Changing Topology Mesh Movement for Viscous Mesh Generation. In *Proceedings of the 22nd International Meshing Roundtable*. Springer International Publishing, 241–261.
- Pierre Alliez, David Cohen-Steiner, Mariette Yvinec, and Mathieu Desbrun. 2005. Variational Tetrahedral Meshing. *ACM Transactions on Graphics* 24, 3 (07 2005), 617.
- Martin S. Alnæs, Jar Blechta, Johan Hake, August Johansson, Benjamin Kehlet, Anders Logg, Chris Richardson, Johannes Ring, Marie E. Rognes, and Garth N. Wells. 2015. The FEniCS Project Version 1.5. *Archive of Numerical Software* 3, 100 (2015).
- G. Alzetta, D. Arndt, W. Bangerth, V. Boddu, B. Brands, D. Davydov, R. Gassmoeller, T. Heister, L. Heltai, K. Kormann, M. Kronbichler, M. Maier, J.-P. Pelteret, B. Turcksin, and D. Wells. 2018. The deal.II Library, Version 9.0. *Journal of Numerical Mathematics* 26, 4 (2018), 173–183.
- ANSYS Inc. 2019. ANSYS®. <https://www.ansys.com/>.
- Jernej Barbic, Fun Shing Sin, and Daniel Schroeder. 2012. Vega FEM Library. <http://www.jernejbarbic.com/vega>.
- Gernot Beer, Stéphane Bordas, et al. 2015. *Isogeometric Methods for Numerical Simulation*. Vol. 240. Springer.
- Steven E Benzley, Ernest Perry, Karl Merkley, Brett Clark, and Greg Sjaardama. 1995. A Comparison of All Hexagonal and All Tetrahedral Finite Element Meshes for Elastic and Elasto-Plastic Analysis. In *Proceedings, 4th international meshing roundtable*, Vol. 17. Sandia National Laboratories, 179–191.
- P.-E. Bernard, J.-F. Remacle, N. Kowalski, and C. Geuzaine. 2016. Frame field smoothness-based approach for hex-dominant meshing. *Computer-Aided Design* 72 (2016), 78–86. <https://doi.org/10.1016/j.cad.2015.10.003> 23rd International Meshing Roundtable Special Issue: Advances in Mesh Generation.
- Jean-Daniel Boissonnat and Steve Oudot. 2005. Provably Good Sampling and Meshing of Surfaces. *Graphical Models* 67, 5 (09 2005), 405–451.
- Xavier Bourdin, Xavier Trossette, Philippe Petit, and Philippe Beillas. 2007. Comparison of Tetrahedral and Hexahedral Meshes for Organ Finite Element Modeling: An Application to Kidney Impact. In *20th International technical conference on the enhanced safety of vehicle*, Lyon.
- Matteo Bracci, Marco Tarini, Nico Pietroni, Marco Livesu, and Paolo Cignoni. 2019. HexaLab.net: An Online Viewer for Hexahedral Meshes. *Computer-Aided Design* 110 (05 2019), 24–36.
- Robert Bridson and Crawford Doran. 2014. Quartet: A tetrahedral mesh generator that does isosurface stuffing with an acute tetrahedral tile. <https://github.com/crawforddoran/quartet>.
- Jonathan R. Bronson, Joshua A. Levine, and Ross T. Whitaker. 2013. Lattice Cleaving: Conforming Tetrahedral Meshes of Multimaterial Domains With Bounded Quality. In *Proceedings of the 21st International Meshing Roundtable*. Springer Berlin Heidelberg, 191–209.
- C.D. Cantwell, D. Moxey, A. Comerford, A. Bolis, G. Rocco, G. Mengaldo, D. De Grazia, S. Yakovlev, J.-E. Lombard, D. Ekelschot, B. Jordi, H. Xu, Y. Mohamied, C. Eskilsson, B. Nelson, P. Vos, C. Biotto, R.M. Kirby, and S.J. Sherwin. 2015. Nektar++: An open-source spectral/hplement framework. *Computer Physics Communications* 192 (07 2015), 205–219.
- Graham F Carey. 1997. *Computational Grids: Generations, Adaptation & Solution Strategies*. CRC Press.
- Roberto Cavoretto, Teseo Schneider, and Patrick Zulian. 2018. OpenCL based parallel algorithm for RBF-PUM interpolation. *Journal of Scientific Computing* 74, 1 (Jan. 2018), 267–289.
- Long Chen and Jin-chao Xu. 2004. Optimal Delaunay Triangulations. *Journal of Computational Mathematics* (2004), 299–308.
- Siu-Wing Cheng, Tamal K Dey, and Joshua A Levine. 2008. A Practical Delaunay Meshing Algorithm for a Large Class of Domains. In *Proceedings of the 16th International Meshing Roundtable*. Springer, 477–494.
- L. P. Chew. 1987. Constrained Delaunay Triangulations. In *Proceedings of the third annual symposium on Computational geometry - SCG '87*. ACM Press.
- L. Paul Chew. 1993. Guaranteed-Quality Mesh Generation for Curved Surfaces. In *Proceedings of the ninth annual symposium on Computational geometry - SCG '93*. ACM Press.
- Philippe G Ciarlet. 2002a. *The finite element method for elliptic problems*. Vol. 40. Siam.
- Philippe G. Ciarlet. 2002b. *The Finite Element Method for Elliptic Problems*. Society for Industrial and Applied Mathematics.
- A. O. Cifuentes and A. Kalbag. 1992. A Performance Study of Tetrahedral and Hexahedral Elements in 3-D Finite Element Structural Analysis. *Finite Elements in Analysis and Design* 12, 3-4 (12 1992), 313–318.
- David Cohen-Steiner, Éric Colin de Verdière, and Mariette Yvinec. 2002. Conforming Delaunay Triangulations in 3D. In *Proceedings of the eighteenth annual symposium on Computational geometry - SCG '02*. ACM Press.
- COMSOL Inc. 2018. COMSOL Multiphysics. <https://www.comsol.com/>.
- Coreform. 2020. Cubit. <https://coreform.com/products/coreform-cubit/>.
- Jean-Christophe Cuillière, Vincent François, and Jean-Marc Drouet. 2013. Automatic 3D Mesh Generation of Multiple Domains for Topology Optimization Methods. In *Proceedings of the 21st International Meshing Roundtable*. Springer Berlin Heidelberg, 243–259.
- Arne De Coninck, Bernard De Baets, Drosos Kourounis, Fabio Verbosio, Olaf Schenk, Steven Maenhout, and Jan Fostier. 2016. Needles: Toward Large-Scale Genomic Prediction with Marker-by-Environment Interaction. 203, 1 (2016), 543–555.
- Tamal K. Dey and Joshua A. Levine. 2008. Delpsc: A Delaunay Mesher for Piecewise Smooth Complexes. In *Proceedings of the twenty-fourth annual symposium on Computational geometry - SCG '08*. ACM Press.
- Crawford Doran, Athena Chang, and Robert Bridson. 2013. Isosurface Stuffing Improved: Acute Lattices and Feature Matching. In *ACM SIGGRAPH 2013 Talks*. ACM Press.
- Qiang Du and Desheng Wang. 2003. Tetrahedral Mesh Generation and Optimization Based on Centroidal Voronoi Tessellations. *International journal for numerical methods in engineering* 56, 9 (2003), 1355–1373.

- Mohamed S. Ebeida, Anjul Patney, John D. Owens, and Eric Mestreau. 2011. Isotropic Conforming Refinement of Quadrilateral and Hexahedral Meshes Using Two-Refinement Templates. *Internat. J. Numer. Methods Engrg.* 88, 10 (05 2011), 974–985. EDF. 2018. Code\_Aster. <https://www.code-aster.org/>.
- Elmer. 2018. Elmer FEM. <http://www.elmefem.org/>.
- Ahmed H. Elsheikh and Mustafa Elsheikh. 2014. A Consistent Octree Hanging Node Elimination Algorithm for Hexahedral Mesh Generation. *Advances in Engineering Software* 75 (09 2014), 86–100.
- Robert D. Falgout and Ulrike Meier Yang. 2002. hypre: A Library of High Performance Preconditioners. In *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 632–641.
- Xianzhong Fang, Weiwei Xu, Hujun Bao, and Jin Huang. 2016. All-Hex Meshing Using Closed-Form Induced Polycube. *ACM Transactions on Graphics* 35, 4 (07 2016), 1–9.
- François Faure, Christian Duriez, Hervé Delingette, Jérémie Allard, Benjamin Gilles, Stéphanie Marchesseau, Hugo Talbot, Hadrien Courtecuisse, Guillaume Bouquet, Igor Peterlik, and Stéphane Cotin. 2012. SOFA: A Multi-Model Framework for Interactive Physical Simulation. In *Studies in Mechanobiology, Tissue Engineering and Biomaterials*. Springer Berlin Heidelberg, 283–321.
- Richard Franke. 1979. *A Critical Comparison of Some Methods for Interpolation of Scattered Data*. Technical Report.
- Xiao-Ming Fu, Chong-Yang Bai, and Yang Liu. 2016. Efficient Volumetric PolyCube-Map Construction. *Computer Graphics Forum* 35, 7 (10 2016), 97–106.
- Xifeng Gao, Tobias Martin, Sai Deng, Elaine Cohen, Zhigang Deng, and Guoning Chen. 2016. Structured Volume Decomposition via Generalized Sweeping. *IEEE Transactions on Visualization and Computer Graphics* 22, 7 (07 2016), 1899–1911.
- C. Geuzaine. 2008. GetDP: a general finite-element solver for the de Rham complex. In *PAMM Volume 7 Issue 1. Special Issue: Sixth International Congress on Industrial Applied Mathematics (ICIAM07) and GAMM Annual Meeting*, Zürich 2007. Wiley.
- James Gregson, Alla Sheffer, and Eugene Zhang. 2011. All-Hex Mesh Generation via Volumetric PolyCube Deformation. *Computer Graphics Forum* 30, 5 (08 2011), 1407–1416.
- Gaël Guennebaud, Benoît Jacob, et al. 2010. Eigen v3. <http://eigen.tuxfamily.org>.
- Hao-Xiang Guo, Xiaohan Liu, Dong-Ming Yan, and Yang Liu. 2020. Cut-Enhanced PolyCube-Maps for Feature-Aware All-Hex Meshing. *ACM Trans. Graph.* 39, 4, Article 106 (July 2020).
- Robert Haines. 2014. MOSS: Multiple Orthogonal Strand System. In *Proceedings of the 22nd International Meshing Roundtable*. Springer International Publishing, 75–91.
- F. Hecht. 2012. New development in FreeFem++. *J. Numer. Math.* 20, 3–4 (2012), 251–265.
- Michael A. Heroux, Roscoe A. Bartlett, Vicki E. Howle, Robert J. Hoekstra, Jonathan J. Hu, Tamara G. Kolda, Richard B. Lehoucq, Kevin R. Long, Roger P. Pawlowski, Eric T. Phipps, Andrew G. Salinger, Heidi K. Thornquist, Ray S. Tuminaro, James M. Willenbring, Alan Williams, and Kendall S. Stanley. 2005. An overview of the Trilinos project. *ACM Trans. Math. Softw.* 31, 3 (2005), 397–423.
- Yixin Hu, Teseo Schneider, Bolun Wang, Denis Zorin, and Daniele Panozzo. 2020. Fast Tetrahedral Meshing in the Wild. *ACM Trans. Graph.* 39, 4 (July 2020).
- Yixin Hu, Qingnan Zhou, Xifeng Gao, Alec Jacobson, Denis Zorin, and Daniele Panozzo. 2018. Tetrahedral Meshing in the Wild. *ACM Transactions on Graphics* 37, 4 (07 2018), 1–14.
- Jin Huang, Tengfei Jiang, Zeyun Shi, Yiyi Tong, Hujun Bao, and Mathieu Desbrun. 2014.  $\ell_1$ -Based Construction of Polycube Maps From Complex Shapes. *ACM Transactions on Graphics* 33, 3 (06 2014), 1–11.
- Jin Huang, Yiyi Tong, Hongyu Wei, and Hujun Bao. 2011. Boundary Aligned Smooth 3D Cross-Frame Field. *ACM Transactions on Graphics* 30, 6 (12 2011), 1.
- T.J.R. Hughes. 2012. *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*. Dover Publications.
- T. J. R. Hughes, J. A. Cottrell, and Y. Bazilevs. 2005. Isogeometric Analysis: CAD, Finite Elements, NURBS, Exact Geometry and Mesh Refinement. *Computer Methods in Applied Mechanics and Engineering* 194, 39–41 (10 2005), 4135–4195.
- Yasushi Ito, Alan M. Shih, and Bharat K. Soni. 2009. Octree-Based Reasonable-Quality Hexahedral Mesh Generation Using a New Set of Refinement Templates. *Internat. J. Numer. Methods Engrg.* 77, 13 (03 2009), 1809–1833.
- Wenzel Jakob, Marco Tarini, Daniele Panozzo, and Olga Sorkine-Hornung. 2015. Instant Field-Aligned Meshes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH ASIA)* 34, 6 (Nov 2015). <https://doi.org/10.1145/2816795.2818078>
- Clément Jamin, Pierre Alliez, Mariette Yvinec, and Jean-Daniel Boissonnat. 2015. CGALmesh: A Generic Framework for Delaunay Mesh Generation. *ACM Trans. Math. Software* 41, 4 (10 2015), 1–24.
- Tengfei Jiang, Jin Huang, Yuanzhen Wang, Yiyi Tong, and Hujun Bao. 2014. Frame Field Singularity Correction for Automatic Hexahedralization. *IEEE Transactions on Visualization and Computer Graphics* 20, 8 (08 2014), 1189–1199.
- B. S. Kirk, J. W. Peterson, R. H. Stogner, and G. F. Carey. 2006. libMesh: A C++ Library for Parallel Adaptive Mesh Refinement/Coarsening Simulations. *Engineering with Computers* 22, 3–4 (2006), 237–254.
- Nils Kohl, Dominik Thönnes, Daniel Drzisga, Dominik Bartuschat, and Ulrich Rüde. 2019. The HyTeG finite-element software framework for scalable multigrid solvers. *International Journal of Parallel, Emergent and Distributed Systems* 34, 5 (2019), 477–496. <https://doi.org/10.1080/17445760.2018.1506453>
- D. Kourounis, A. Fuchs, and O. Schenk. 2018. Towards the Next Generation of Multitiered Optimal Power Flow Solvers. *IEEE Transactions on Power Systems* PP, 99 (2018), 1–10.
- François Labelle and Jonathan Richard Shewchuk. 2007. Isosurface Stuffing: Fast Tetrahedral Meshes With Good Dihedral Angles. In *ACM SIGGRAPH 2007 papers on - SIGGRAPH '07*. ACM Press.
- Konstantin Ladutenko. 2018. FEA-Compare. <https://github.com/kostyfisik/FEA-compare>.
- Taeyong Lee and R. S. Lakes. 1997. Anisotropic polyurethane foam with Poisson's ratio greater than 1. *Journal of Materials Science* 32, 9 (01 May 1997), 2397–2401.
- Bo Li, Xin Li, Kexiang Wang, and Hong Qin. 2013. Surface Mesh to Volumetric Spline Conversion With Generalized Polycurves. *IEEE Transactions on Visualization and Computer Graphics* 19, 9 (09 2013), 1539–1551.
- Yufei Li, Yang Liu, Weiwei Xu, Wenping Wang, and Baining Guo. 2012. All-Hex Meshing Using Singularity-Restricted Field. *ACM Transactions on Graphics* 31, 6 (11 2012), 1.
- Heng Liu, Paul Zhang, Edward Chien, Justin Solomon, and David Bommes. 2018. Singularity-Constrained Octahedral Fields for Hexahedral Meshing. *ACM Transactions on Graphics* (07 2018).
- Marco Livesu, Alessandro Muntoni, Enrico Puppo, and Riccardo Scateni. 2016. Skeleton-Driven Adaptive Hexahedral Meshing of Tubular Shapes. *Computer Graphics Forum* 35, 7 (10 2016), 237–246.
- Marco Livesu, Nicholas Vining, Alla Sheffer, James Gregson, and Riccardo Scateni. 2013. PolyCut: Monotone Graph-Cuts for PolyCube Base-Complex Construction. *ACM Transactions on Graphics* 32, 6 (11 2013), 1–12.
- Precise Simulation Ltd. 2019. FEATool Multiphysics v1.10, User's Guide. <https://www.featuol.com>.
- Max Lyon, David Bommes, and Leif Kobbelt. 2016. HexEx: Robust Hexahedral Mesh Extraction. *ACM Trans. Graph.* 35, 4, Article 123 (July 2016), 11 pages. <https://doi.org/10.1145/2897824.2925976>
- Tobias Martin and Elaine Cohen. 2010. Volumetric Parameterization of Complex Objects by Respecting Multiple Materials. *Computers & Graphics* 34, 3 (06 2010), 187–197.
- Loïc Maréchal. 2009. Advances in Octree-Based All-Hexahedral Mesh Generation: Handling Sharp Features. In *18th international meshing roundtable*. Springer.
- A. T. T. McRae, G.-T. Bercea, L. Mitchell, D. A. Ham, and C. J. Cotter. 2016. Automated Generation and Symbolic Manipulation of Tensor Product Finite Elements. *SIAM Journal on Scientific Computing* 38, 5 (01 2016), S25–S47.
- MFEM. 2020. MFEM: Modular Finite Element Methods Library. <https://mfem.org>.
- Neil Molino, Robert Bridson, and Ronald Fedkiw. 2003. Tetrahedral Mesh Generation for Deformable Bodies. In *Proc. Symposium on Computer Animation*.
- Michael Murphy, David M. Mount, and Carl W. Gable. 2001. A Point-Placement Strategy for Conforming Delaunay Tetrahedralization. *International Journal of Computational Geometry & Applications* 11, 06 (12 2001), 669–682.
- N. M. Newmark. 1959. *A method of computation for structural dynamics*. American Society of Civil Engineers.
- M. Nieser, U. Reitebuch, and K. Polthier. 2011. CubeCover- Parameterization of 3D Volumes. *Computer Graphics Forum* 30, 5 (08 2011), 1397–1406.
- Bernardo Lino De Oliveira and Joakim Sundnes. 2016. Comparison of Tetrahedral and Hexahedral Meshes for Finite Element Simulation of Cardiac Electro-Mechanics. In *Proceedings of the VII European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS Congress 2016)*. Institute of Structural Analysis and Antiseismic Research School of Civil Engineering National Technical University of Athens (NTUA) Greece.
- Steven J Owen. 1998. A Survey of Unstructured Mesh Generation Technology. In *IMR*, 239–267.
- Steven J. Owen, Ryan M. Shih, and Corey D. Ernst. 2017. A Template-Based Approach for Parallel Hexahedral Two-Refinement. *Computer-Aided Design* 85 (04 2017), 34–52.
- Luiz Claudio Pardini and Maria Luisa Gregori. 2010. Modeling elastic and thermal properties of 2.5D carbon fiber and carbon/SiC hybrid matrix composites by homogenization method. *Journal of Aerospace Technology and Management* 2 (08 2010), 183 – 194.
- B. Patzák. 2012. OOFEM - an object-oriented simulation tool for advanced modeling of materials and structures. *Acta Polytechnica* 52, 6 (2012), 59–66.
- Christophe Prud'homme, Vincent Chabannes, Vincent Doyeux, Mourad Ismail, Abdoulaye Samake, and Goncalo Pena. 2012. Feel++ : A computational framework for Galerkin Methods and Advanced Numerical Methods. *ESAIM: Proceedings* 38 (12 2012), 429–455.
- Jin Qian and Yongjie Zhang. 2010. Sharp Feature Preservation in Octree-Based Hexahedral Mesh Generation for CAD Assembly Models. In *Proceedings of the 19th International Meshing Roundtable*. Springer Berlin Heidelberg, 243–262.
- A. Ramos and J. A. Simões. 2006. Tetrahedral Versus Hexahedral Finite Elements in Numerical Modelling of the Proximal Femur. *Medical Engineering & Physics* 28, 9 (11 2006), 916–924.
- Jean-François Remacle. 2017. A Two-Level Multithreaded Delaunay Kernel. *Computer-Aided Design* 85 (04 2017), 2–9.

- Yves Renard and Julien Pommier. 2018. Getfem++, an open source generic C++ library for finite element methods. <http://getfem.org/>.
- J. Ruppert. 1995. A Delaunay Refinement Algorithm for Quality 2-Dimensional Mesh Generation. *Journal of Algorithms* 18, 3 (05 1995), 548–585.
- Kambiz Salarai and Patrick Knupp. 2000. *Code Verification by the Method of Manufactured Solutions*. Technical Report.
- Dominik Schillinger, Shaikh J. Hossain, and Thomas J.R. Hughes. 2014. Reduced Bézier element quadrature rules for quadratic and cubic splines in isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering* 277 (aug 2014), 1–45.
- Teseo Schneider, Jérémie Dumas, Xifeng Gao, Daniele Panozzo, and Denis Zorin. 2019a. Poly-Spline Finite Element Method. *ACM Transactions on Graphics* 38, 3 (03 2019), 1–14.
- Teseo Schneider, Jérémie Dumas, Xifeng Gao, Denis Zorin, and Daniele Panozzo. 2019b. PolyFEM. <https://polyfem.github.io/>.
- Teseo Schneider, Yixin Hu, Jérémie Dumas, Xifeng Gao, Daniele Panozzo, and Denis Zorin. 2018. Decoupling simulation accuracy from mesh quality. *ACM Transactions on Graphics* 37, 6 (dec 2018).
- R. Schneiders. 1996. A Grid-Based Algorithm for the Generation of Hexahedral Element Meshes. *Engineering with Computers* 12, 3–4 (09 1996), 168–177.
- Robert Schneiders and Rolf Bünten. 1995. Automatic Generation of Hexahedral Finite Element Meshes. *Computer Aided Geometric Design* 12, 7 (11 1995), 693–707.
- Robert Schneiders, Roland Schindler, and Frank Weiler. 1996. Octree-Based Generation of Hexahedral Element Meshes. In *In Proceedings Of The 5th International Meshing Roundtable*. Sandia National Laboratories.
- Joachim Schöberl. 2014. C++11 implementation of finite elements in NGSolve. *Institute for Analysis and Scientific Computing, Vienna University of Technology* (2014).
- Donald R. Sheehy. 2012. New Bounds on the Size of Optimal Meshes. *Computer Graphics Forum* 31, 5 (08 2012), 1627–1635.
- Jason F. Shepherd and Chris R. Johnson. 2008. Hexahedral Mesh Generation Constraints. *Engineering with Computers* 24, 3 (03 2008), 195–213.
- Jonathan Shewchuk. 2012. *Delaunay Mesh Generation*. Chapman and Hall/CRC.
- Jonathan Richard Shewchuk. 1996. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In *Selected Papers from the Workshop on Applied Computational Geometry, Towards Geometric Engineering (FCRC '96/WACG '96)*. Springer-Verlag, 203–222.
- Jonathan Richard Shewchuk. 1998. Tetrahedral Mesh Generation by Delaunay Refinement. In *Proceedings of the fourteenth annual symposium on Computational geometry - SCG '98*. ACM Press.
- Jonathan Richard Shewchuk. 2002. Constrained Delaunay Tetrahedralizations and Provably Good Boundary Recovery. In *11th International Meshing Roundtable*. Sandia National Laboratories.
- Hang Si. 2015. TetGen, a Delaunay-Based Quality Tetrahedral Mesh Generator. *ACM Trans. Math. Softw.* 41, 2, Article 11 (Feb. 2015), 36 pages.
- Hang Si and Klaus Gärtner. 2005. Meshing Piecewise Linear Complexes by Constrained Delaunay Tetrahedralizations. In *Proceedings of the 14th international meshing roundtable*. Springer, 147–163.
- Hang Si and Jonathan Richard Shewchuk. 2014. Incrementally Constructing and Updating Constrained Delaunay Tetrahedralizations With Finite-Precision Coordinates. *Engineering with Computers* 30, 2 (04 2014), 253–269.
- Justin Solomon, Amir Vaxman, and David Bommes. 2017. Boundary Element Octahedral Fields in Volumes. *ACM Transactions on Graphics* 36, 3 (05 2017), 1–16.
- Tomás Šoltyš. 2019. Range Software. <http://www.range-software.com/>.
- Spatial. 2018. MeshGems. <http://meshgems.com/volume-meshing-meshgems-hex.html>. Accessed: 2018-06-05.
- Y. Su, K. H. Lee, and A. Senthil Kumar. 2004. Automatic Hexahedral Mesh Generation for Multi-Domain Composite Models Using a Hybrid Projective Grid-Based Method. *Computer-Aided Design* 36, 3 (03 2004), 203–215.
- Barna Szabó and Ivo Babuška. 1991. *Finite element analysis*. John Wiley & Sons.
- Srinivas C. Tadepalli, Ahmet Erdemir, and Peter R. Cavanagh. 2010. A Comparison of the Performance of Hexahedral and Tetrahedral Elements in Finite Element Models of the Foot. In *ASME 2010 Summer Bioengineering Conference, Parts A and B*. ASME.
- Srinivas C. Tadepalli, Ahmet Erdemir, and Peter R. Cavanagh. 2011. Comparison of Hexahedral and Tetrahedral Elements in Finite Element Analysis of the Foot and Footwear. *Journal of Biomechanics* 44, 12 (08 2011), 2337–2343.
- Timothy J. Tautges. 2001. The Generation of Hexahedral Meshes for Assembly Geometry: Survey and Progress. *Internat. J. Numer. Methods Engrg.* 50, 12 (2001).
- Jane Tournois, Camille Wormser, Pierre Alliez, and Mathieu Desbrun. 2009. Interleaving Delaunay Refinement and Optimization for Practical Isotropic Tetrahedron Mesh Generation. *ACM Transactions on Graphics* 28, 3 (07 2009), 1.
- Fabio Verbosio, Arne De Coninck, Drosos Kourounis, and Olaf Schenk. 2017. Enhancing the scalability of selected inversion factorization algorithms in genomic prediction. *Journal of Computational Science* 22, Supplement C (2017), 99 – 108.
- Erke Wang, Thomas Nelson, and Rainer Rauch. 2004. Back to Elements-Tetrahedra vs. Hexahedra. In *Proceedings of the 2004 international ANSYS conference*. ANSYS Pennsylvania.
- Xiaodong Wei, Yongjie Jessica Zhang, Deepesh Toshniwal, Hendrik Speleers, Xin Li, Carla Manni, John A. Evans, and Thomas J. R. Hughes. 2018. Blended B-Spline Construction on Unstructured Quadrilateral and Hexahedral Meshes With Optimal Convergence Rates in Isogeometric Analysis. *Computer Methods in Applied Mechanics and Engineering* 341 (11 2018), 609–639.
- Hongmei Zhang, Guoqun Zhao, and Xinwu Ma. 2007. Adaptive Generation of Hexahedral Element Mesh Using an Improved Grid-Based Method. *Computer-Aided Design* 39, 10 (10 2007), 914–928.
- Yongjie Zhang and Chandrajit Bajaj. 2006. Adaptive and Quality Quadrilateral/Hexahedral Meshing From Volumetric Data. *Computer Methods in Applied Mechanics and Engineering* 195, 9–12 (02 2006).
- Yongjie Zhang, Xinghua Liang, and Guoliang Xu. 2013. A Robust 2-Refinement Algorithm in Octree and Rhombic Dodecahedral Tree Based All-Hexahedral Mesh Generation. In *Proceedings of the 21st International Meshing Roundtable*. Springer Berlin Heidelberg, 155–172.
- Hui Zhao, Na Lei, Xuan Li, Peng Zeng, Ke Xu, and Xianfeng Gu. 2018. Robust Edge-Preserving Surface Mesh Polycube Deformation. *Computational Visual Media* 4, 1 (01 2018), 33–42.
- Qingnan Zhou and Alec Jacobson. 2016. Thingi10K: A Dataset of 10, 000 3D-Printing Models. *CoRR* abs/1605.04797 (2016). arXiv:1605.04797
- Olek C Zienkiewicz, Robert L Taylor, and Jian Z Zhu. 2005. *The Finite Element Method: Its Basis and Fundamentals*. Elsevier.