A Large Scale Comparison of Tetrahedral and Hexahedral Elements for Finite Element Analysis

Teseo Schneider[a,*], Yixin Hu[a], Xifeng Gao[b], Jérémie Dumas[c], Denis Zorin[a], Daniele Panozzo[a]

[a] *NYU Courant, New York*
[b] *Florida State University, Florida*
[c] *nTopology, New York*

Abstract

The Finite Element Method (FEM) is widely used to solve discrete Partial Differential Equations (PDEs) and it is a staple in most engineering applications. The popularity of this approach led to the development of a large family of variants, and, while their theoretical properties (such as convergence rate, stability, etc.) are well studied, their practical performances have never been systematically studied over a large collection of 3D models.

We introduce a large set of benchmark problems, starting from simple cases with an analytical solution, moving to classical experimental setups, and finally fabricating solutions for thousands of real-world geometries. For all these cases, we use state-of-the-art meshing tools to create both unstructured (tetrahedral) and structured (hexahedral) meshes, and compare the performance of different element types for a wide spectrum of elliptic PDEs ranging from heat convection to fluid propagation.

We observe that, while linear tetrahedral elements perform poorly, often leading to locking artefacts, quadratic tetrahedral elements outperform hexahedral elements in all the settings we tested. This unexpected result suggests that for static structural analysis, thermal analysis, and low reynolds number flows it is unnecessary to target automatic hex mesh generation, since superior results can be obtained with unstructured meshes, which can be created robustly and automatically with existing meshing algorithms.

We release the experimental description, meshes, and reference implementation of our testing infrastructure, providing a standard benchmark. This enables statistically significant comparisons between different FE methods and discretization, which we believe will provide a guide in the development of new meshing and FEA techniques.

*Corresponding author
Email address: teseo.schneider@nyu.edu (Teseo Schneider)

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.

In this work, we introduce a large scale study to systematically compare the performances of several common choices of FEM discretizations in a set of benchmark problems. We consider three types of volumetric discretizations: *unstructured* collection of tetrahedral elements, *semi-structured* collection of hexahedral elements, and regular lattices. The purpose of our study is to compare efficiency of different elements, i.e., how much time is typically required to obtain a solution with given accuracy with different element types. For all cases, we consider standard Lagrangian bases [1] of varying degree, in addition to serendipity [2] elements (for hexes only), which are by far the most popular brick element used in commercial FEM systems. Finally, we show several comparisons using spline-based elements [3], which have recently gained popularity in the IsoGeometric Analysis (IGA) community.

We devised a large set of test problems of varying complexity for elliptic PDEs (including Poisson, linear elasticity, Neo-Hookean elasticity, incompressible elasticity, and Stokes). Our set includes common test problems: 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 [4] on 3 200 real-world, complex 3D models. Our model collection includes both CAD models and scanned geometries, providing a realistic sampling of analysis scenarios.

Our study indicates that in most situations, quadratic tetrahedral elements are more efficient (in terms of time it takes to achieve a given accuracy) than Lagrangian elements on semi-structured hexahedral meshes, and are somewhat inferior to the performance of spline elements on regular lattices. Combined with available state-of-the-art robust meshing techniques, quadratic tetrahedral elements can be used to realize a "black-box" pipeline, without sacrificing performance compared to hexhedral elements, which require far more complex and less robust mesh generation.

We emphasize that our study is limited to elliptic PDEs, and it may be possible that our conclusions will not extend to more challenging scenarios such as fracture, dynamics, shocks, or multi-physics simulations which we leave as future work.

We provide the complete source code[1] for the integrated analysis pipelines we tested, the dataset we used,the benchmark solutions, and the scripts to

[1] https://github.com/polyfem/polyfem/

reproduce all results[2], to enable researchers and practitioners to easily expand this study to additional mesh types (such as polyhedral meshes) and bases.

2. Related Work

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

[5] concludes that quadratic tet-meshes lead to roughly the same accuracy and time as linear hex-meshes, by testing several simple models on different structural problems. By evaluating the eigenvalues of the stiffness matrices of various nonlinear and elasto-plastic problems, [6] reports that linear hex-meshes are superior to linear tet-meshes. The authors also show that linear hex-meshes is slightly better than quadratic tet-meshes in the nonlinear elasto-plastic analysis experiment.

A more recent work, [7, 8], focuses on modeling footwear with a nonlinear incompressible material model under shear force loading conditions. The conclusion of these works is that trilinear hex-meshes are superior to linear tet-meshes, and that quadratic tetrahedral elements are computationally more expensive compared to trilinear hexahedral elements, but have higher accuracy. A study more similar to ours [9] compares tet- and hex-meshes on linear static problems, modal and nonlinear analysis. The study concludes that quadratic tet and hex elements have similar performance, but quadratic hexes are computationally more expensive. The same study also confirms that linear tets are too stiff for large deformations, and linear hexes with large corner angles should be avoided in regions of stress concentration.

In medical applications, results for femur models [10] show that linear tet-meshes of the simplified femur model lead to closer agreement with the theoretical ones, while quadratic hex meshes are more stable and the result is less affected by mesh refinement. On a kidney model, [11] observes that both linear and quadratic tet-meshes are slightly stiffer than hex-meshes, but are more stable when high impact energies are present in the simulation. For heart mechanics and electrophysiology, [12] notes that quadratic hexes are slightly better than quadratic tets in the mechanics regime, while linear tet-meshes are the best choice for the electrophysiology problem.

[2] https://github.com/polyfem/tet-vs-hex

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[3] is beyond the scope of this paper, therefore we discuss only several representative packagages. We point out an interesting project [33] 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 and popular 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 [15], GetFEM++ [23], libMesh [24] and MFEM [25] 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 [14] is another popular open-source FEA library, however it only supports quadrilateral and hexahedral elements. Commercials packages such as ANSYS [34], Abaqus [35], COMSOL Multiphysics [36] support Lagrangian tetrahedral elements, but surprisingly often implement only serendipity elements for hexes [2][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) [3]. 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). Defining this type of elements on fully general hexahedral domains is an open problem [37, 38, 39]. 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 (tet and hex), serendipity and spline basis functions (hex only) in the same framework, we added all the elements and basis used in this study to our own open-source FEA library [29] to ensure a fair comparison. PolyFEM [29] supports all these element types and interfaces with Hypre [40] and PARDISO [41, 42, 43] for the solver and Eigen [44] for linear algebra.

2.3. Meshing

Three-dimentional mesh generation has been thoroughly studied in multiple communities [45, 46, 47, 48]. For the sake of brevity, we restrict our review to the techniques generating pure tetrahedral or pure hexahedral meshes, which

[3] A non-exhaustive list of open-source FEA packages known to the authors include, in alphabetical order, code_aster [13], Deal.II [14], DOLFIN (FEniCS) [15], ElmerFEM [16], FEATool Multiphysics (MATLAB) [17], Feel++ [18], FEI (Trilinos) [19], Firedrake [20], FreeFEM++ [21], GetDP [22], GetFEM++ [23], libMESH [24], MFEM [25], Nektar++ [26], NGSolve [27], OOFEM [28], PolyFEM [29], Range [30], SOFA [31], and VegaFEM [32].

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 [49, 50, 51, 45, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67]. 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 [68].

To overcome these robustness limitations, alternative approaches are based on a background grid [69, 70, 71, 72, 73]. 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 [74, 75, 76] 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 [68]. 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. Hexes 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 [77, 78, 79], which are used in commercial software such as [34, 80].

By allowing lower element quality, one can design automatic approaches based on regular lattices [81, 82, 83, 84, 85] or on octrees [86, 84, 87, 88, 89, 90, 91, 92, 93, 94].

Polycube methods [95, 96, 97, 98, 99, 39, 100] and field-aligned parameterization based methods [101, 102, 103, 104, 105, 106] aim at producing hex-meshes with as few singular 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 [107], which we use in our study. We also generate a new large scale dataset composed of 3200 hexahedral meshes using the commercial MeshGems-Hexa software [94].

3. Background

3.1. Choice of 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. In our comparison, we decided to target the most common choices: we use the standard linear and quadratic Lagrange bases for both tet and hexes (tensor product for hexes) [1], and we also use serendipity [2] (common in industry) and spline [3] (common in IGA) basis for hexes. We use the standard Galerkin formulation [1] with Guassian quadrature for all our experiments, avoiding non-standard quadrature.

3.2. Comparing Meshes

We measure resolution of all meshes (tet and hex) by their number of vertices. The choice is justified by the fact that the number of vertices of often used by the meshing algorithms as a "budget" that the meshing algorithms can use to create the best possible mesh. Another important implication is that same number of vertices lead to the same number of degree of freedom in case of linear (or tri-linear) elements. Finally, matching the number of vertices involves the initial creation of the mesh and not the actual simulation allowing for more control.

In addition to this particular choice, we also investigate other metrics for a specific example (Table 1), and provide an interactive plot that allows to compare our results using 24 different measures (i.e., error H^1, error H^1 semi-norm, error L_2, error L^∞, error L^∞ of gradient, error L^8, average edge length, minimum edge length number of vertices, matrix size, number of non zero entries, number of bases, number of dofs, number of elements, number of pressure bases, time loading mesh, time building basis, time assigning right-hand side, time assembling stiffness matrix, time solving, time computing errors, time total, time total without right-hand side assembly).

3.3. Model PDEs

We selected the following set of representative elliptic problems: (1) Poisson; (2) Stokes; (3) Elasticity with 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

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

for the function

$$u \colon \Omega \to \mathbb{R}^n,$$

where D^2 is the matrix of second derivatives, b is the righ-hand side, $\partial\Omega_D \subset \partial\Omega$ is the part of the boundary with d as Dirichlet boundary conditions, and $\partial\Omega_N \subset \partial\Omega$

is the part of the boundary with f as the Neumann boundary condition. 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.

Poisson Equation. This equation is given by

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

Incompressible Stokes Equations. The Stokes equations provide the relationship between the velocity u and the pressure p for an incompressible fluid with viscosity μ.

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

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

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

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 stress σ is related to the displacement field u. For the linear Hooke model,

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

where $\epsilon[u]$ is the strain tensor, λ is the first Lamé parameter, and μ is the shear modulus.

When considering incompressible materials (Poisson ratio equal to 0.5), λ goes to infinity and the previous equation is not well defined. Additionally, when λ 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 λs, and reduces to incompressible elasticity for $\lambda^{-1} \to 0$.

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

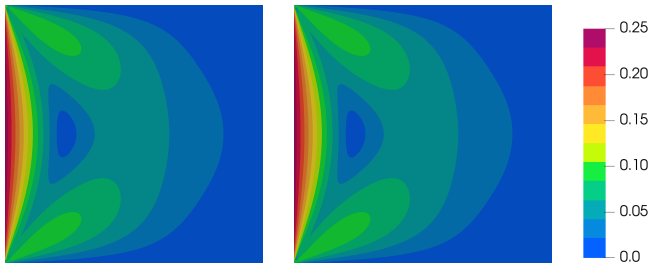


Figure 1: Norm of velocity for a Stokes simulation with mixed linear and quadratic elements.

Finally, the in the Neo-Hooke material model the stress is a nonlinear function of strain.

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

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

For elasticity problems, we often use the von Mises stresses

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

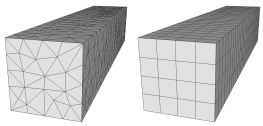
4. Common Test Problems

We collected a number of standard test cases for fluid simulation (Section 4.1), linear Hookean material deformation (Sections 4.2 and 4.3), nearly incompressible linear material (Section 4.4), and nonlinear Neo-Hookean material (Sections 4.5 and 4.6). Most of the domains of interest are chosen to simplify manual creation of hexahedral meshes: the simulations will be performed on a unstructured tetrahedral mesh and a regular lattice with the same number of vertices. Experiments in Sections 4.2, 4.3, and 4.4 are run on a MacBook Pro 3.1GHz Intel Core i7, 16GB of RAM, and 8 threads. Experiments in Sections 4.5 and 4.6 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 PolyFEM library [29], which uses the Pardiso [41, 42, 43] direct solver, and Newton iterations for the nonlinear problems. Note that, for completeness, we also validated PolyFEM on the example in Figure 2 for linear and quadratic tetrahedra and serendipity hexahedra on Hooke material against Abaqus. The results are identical up to numerical precision.

4.1. Stokes

For fluid simulation, we test on planar square domain meshes with 4 229 vertices for the triangle mesh and 4 225 vertices for the regular grid. We simulate

	t_b	t_a	t_s	t	e_f
P_1	9.65e-3	1.80e-2	3.72e-2	6.49e-2	6.14e-3
P_2	2.62e-2	2.01e-1	4.73e-1	7.00e-1	9.19e-5
Q_1	6.65e-3	2.57e-2	4.28e-2	7.52e-2	1.27e-3
Q_2	1.73e-2	4.34e-1	6.36e-1	1.09	4.66e-5



Tetmesh Hexmesh

Figure 2: Displacement error per force unit at the endpoint of a beam with a squared section. The times are averaged over 10 runs per force sample.

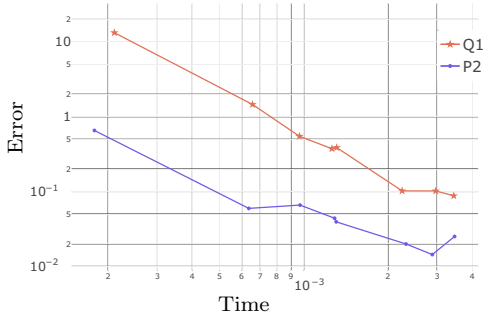


Figure 3: Time versus error comparison plot between P_2 and Q_1.

the Stokesian fluid (2) with viscosity $\mu = 1$ in the standard "driven cavity" example: the fluid has zero boundary conditions on 3 of the four 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 hexes and tets.

4.2. Bended Bar

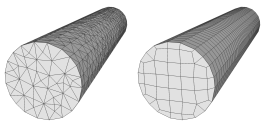
In this experiment, we consider beams with different cross-sections (square, circular, and I-like) in the xy-plane of length L. The beam is clamped (i.e., zero Dirichlet) at the end ($z = L$) and different tangential forces $f = [0, -f_y, 0]^T$, $f_y \in [-0.1, -2]$, are applied at $z = 0$, opposite to the clamped side. The rest of the boundary is left free (i.e., zero Neumann) and we do not apply any body force. For these experiments we use linear Hookean material model (4) with Young's modulus $E = 210\,000$ and Poisson's ratio $\nu = 0.3$. We study the displacement in y-direction of the moving end ($z = 0$) and compare it with a P_4 solution to compute the error e. We report as e_f the linearly fitted trend of the error increase per force unit. We also report the basis construction time t_b, assembly time t_a, solve time t_s, and total time t. Note that, all timinings reported are averages over 10 different runs per force sample.

Square Section. For running the simulation we use a square section of side $s = 20$, length $L = 100$ and mesh it with tetrahedral mesh with 739 vertices and an hexahedral mesh (regular grid) with 750 vertices. Figure 2 shows the errors compared with the dense solution where trilinear hexahedral elements

		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 between tetrahedral and hexahedral meshes with matching dimensions, green shows the "winner." For instance, by comparing P_2 with Q_1 for the same error (last row of the table) P_2 is faster (fist column) uses less memory (second column) and has less DOFs (third column).

	t_b	t_a	t_s	t	e_f
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



Tetmesh Hexmesh

Figure 4: Displacement error per force unit at the endpoint of a beam with a circular section. The times are averaged over 10 runs per force sample.

outperform linear tetrahedral elements but the quadratic counterparts are indistinguishable, whereas timing-wise quadratic tetrahedra are significantly better.

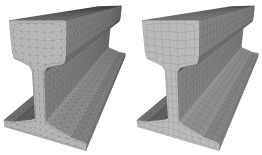
We created a sequence of hexahedral and tetrahedral meshes where the Q_1 error matches the P_2 error for a force $f = [0, -2, 0]^T$. Figure 3 shows that for a given error, P_2 discretization is around 4 times faster than Q_1.

Finally, we created a sequence of hexahedral meshes that matches total time, total memory, total number of degree of freedom, and error of the tetrahedral mesh in Figure 2 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 Section. We consider a bar of length $L = 100$ with a circular 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). Figure 4 shows similar errors as for the square section, P_1 produces low-quality results, while P_2 and Q_2 are similar.

I-beam Section. In our final experiment we use an I-beam (the bounding box of the section is 125×154) of length $L = 473.11$. The tetrahedral mesh has 6 102 while the hexahedral mesh has 6 080 vertices, results are shown in Figure 5.

	t_b	t_a	t_s	t	e_f
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



Tetmesh Hexmesh

Figure 5: Displacement errors per force unit at the endpoint of an I-beam. The times are averaged over 10 runs per force sample.

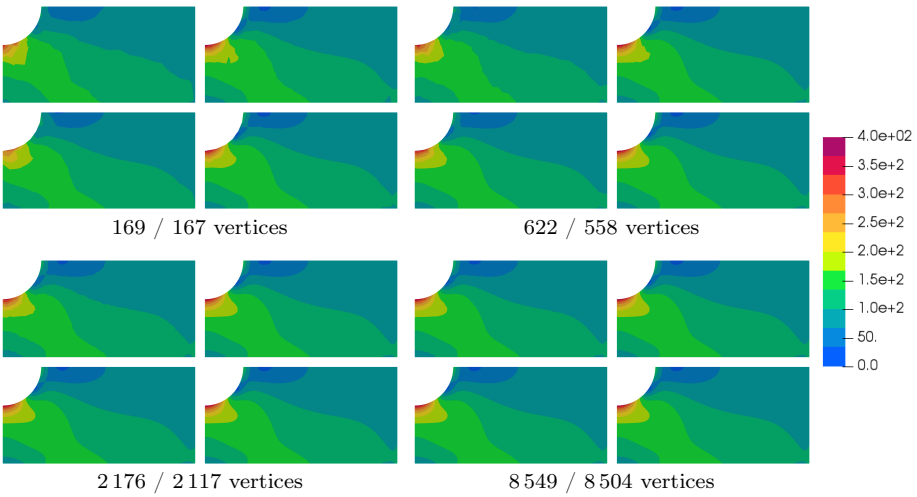


Figure 6: Visualization of the von Mises stresses for four different mesh resolution. Each figure show P_1 in the top left, P_2 in the top right, Q_1 in the bottom left, and Q_2 in the bottom right. The numbers below the figures represent the number of vertices of the tri / quad-mesh.

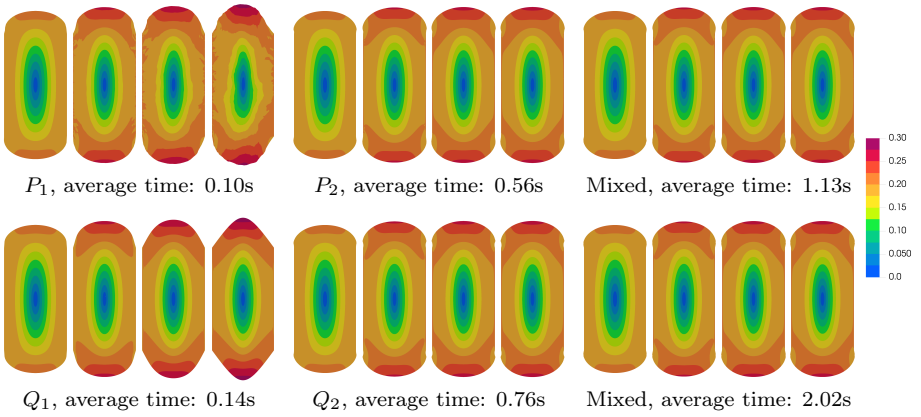


Figure 7: Visualization of the norm of the displacement for a squeezed squared material with $\nu \in \{0.9, 0.99, 0.999, 0.9999\}$ left to right for different elements and discretizations.

4.3. 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 200×100 with an hole in the center of radius 20, the same material model (Hooke elasticity (4)) and same material parameters $E = 210\,000$ and $\nu = 0.3$. Note that the conversion to λ is different because the problem is planar. 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 used 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 orthogonal direction), a zero Neumann, and a Neumann corresponding to the original force. We solve this particular benchmark problem on four meshes with different resolutions. Figure 6 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 produces 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.

4.4. Nearly Incompressible Material

For the last linear benchmark, we compared the performance of the four discretizations in a more and more incompressible regime. We apply Dirichlet boundary condition of $[0.2, 0]$ on the left and $[-0.2, 0]$ on the right of a unit square, thus obtaining an important squeezing effect. 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

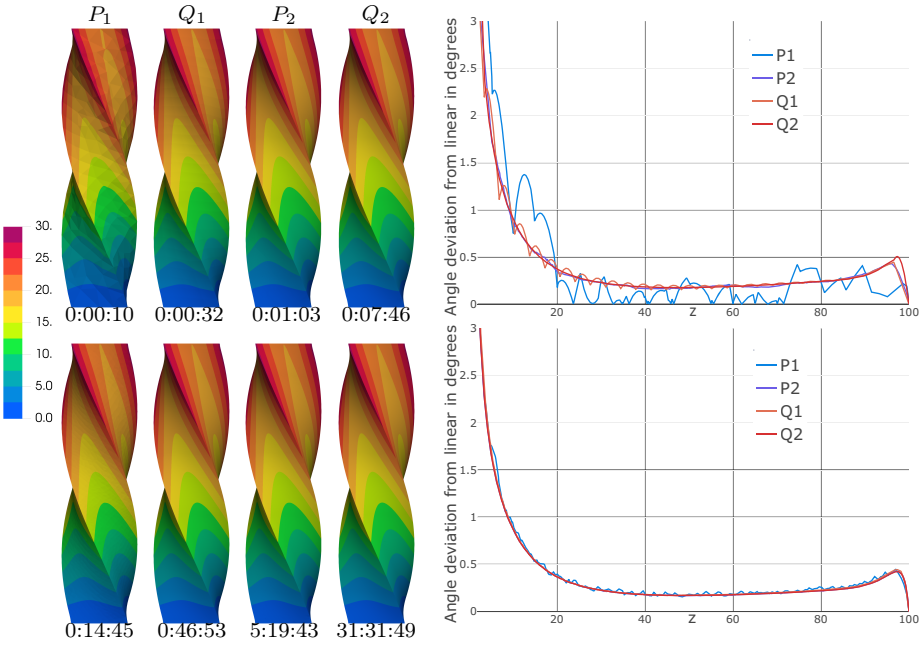


Figure 8: Nonlinear elastic example, where we show the deformed mesh, the running time and the angle displacement deviation with respect to linear along the depth of the bar. We show the deviation from linear to make the differences more visible.

2D). We compare the standard formulation (4) with a mixed formulation (5) that does not become unstable as $\nu \to 1$. Note that, since mixed formulations require two different basis' order for the displacement and the pressure. We performed our experiments using linear pressure bases and quadratic bases for displacement. We mesh the square with quad mesh with 4 225 vertices and a tri mesh with 4 229 vertices.

Figure 7 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 quad-mesh produces a symmetric and "goodlooking" wrong result for the linear case, while the tri-mesh produces an unstable output. The two quadratic discretizations produce a visually similar result as the stable mixed method. The only quantitative difference is that the residual error for our exact solver drops from 1e-15 (numerical zero) to 1e-12, indicating that the system becomes singular.

4.5. Twisted Bar

We now compare the solutions of the Neo-Hooke (6) material model for our discretizations. We take a bar with section $[-10, 10]^2$ and length 100, $E = 200$ $nu = 0.35$, fix the bottom part (zero Dirichelt boundary condition) and apply a rotation of 90 degrees to the top, the rest of the surface is left free. We run the experiment on two sets of tet and hex meshes, where the coarse ones have 739

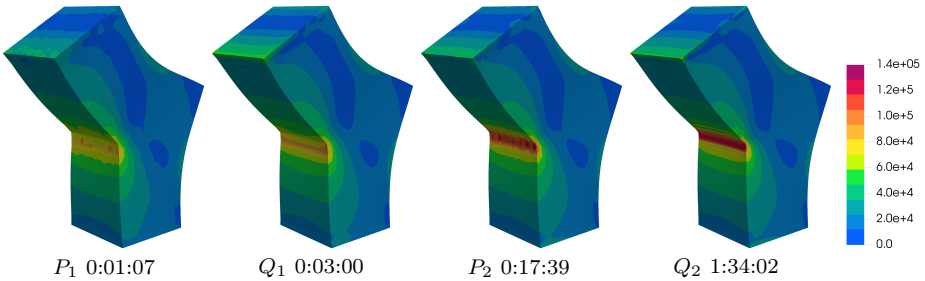


Figure 9: Von Mises stress and time of a singular solution for the four discretizations with nonlinear material model.

and 750 while the dense have 50 000 and 58 719 vertices respectively. Figure 8 first 3 images show that, a part for small numerical fluctuations, the results are indistinguishable. Similar results are for the plot (Figure 8 last plot) of the rotation angle along a line situated at $[9.5, 9.5]$ passing trough the bar. Note that the dense Q_2 solution required more that 44GB of RAM for the solver, while P_2 required around 21. The reason for such long times is the size of the local dense matrix, 27 local DOF times 3 dimensions all squared, which makes for 6 561 entries (versus 144 for P_1, 576 for Q_1, and 900 for P_2).

For this experiment we also use quadratic B-spline bases on the coarse mesh, the result is similar to P_2 and Q_2, see inset. For this particular example we measure the *solve* time of the three discretiaztions: 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). Note that, the assembly time (using full integration, it could be improved using [108]) for spline is similar to Q_2 and is 12 times slower than P_2 (20.54s versus 1.70s).

4.6. High Stress

As a final experiment, we run a simulation for an L-shaped domain with the Neo-Hooke material and $E = 210000$ $\nu = 0.3$. Our goal is to study the differences in the stresses for singular solutions: the concave part of the L will have a discontinuous concentration of stress. We mesh our domains with 14 155 vertices for the tet mesh and 14 161 vertices for the hex 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 9 shows, once again, that linear tet elements underestimate the stress while linear hex-elements are a bit better. The quadratic discretizations are qualitatively similar, the hex-mesh produces a nicer-looking result 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

To confirm the results of the previous section on a large dataset, we compute artificial solutions for the Poisson (1) and Hooke elasticity (4) for linear and quadratic dicretizations and compare the errors on a large dataset. Since general solutions of these PDEs are unknowns we use the *method of manufactured solutions* [4], that is to "invent" a solution u and plug it in the PDE to obtain the right-hand side b and the boundary condition d. For Poisson we use the Franke [109] function

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

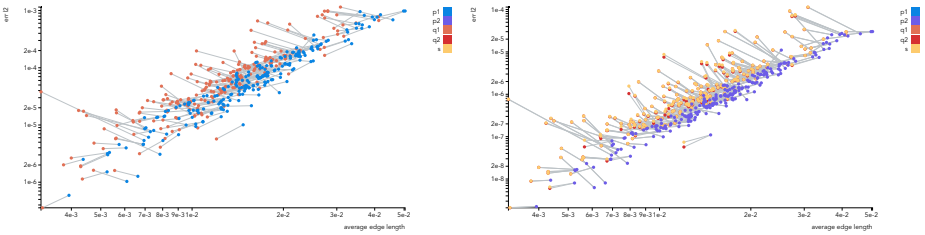
while for elasticity

$$u(x_1, x_2, x_3) = \frac{1}{80} \begin{pmatrix} x_1x_2 + x_1^2 + x_2^3 + 6x_3 \\ x_1x_3 - x_3^3 + x_1x_2^2 + 3x_1^4 \\ x_1x_2x_3 + x_2^2x_3^2 - 2x_1, \end{pmatrix}$$

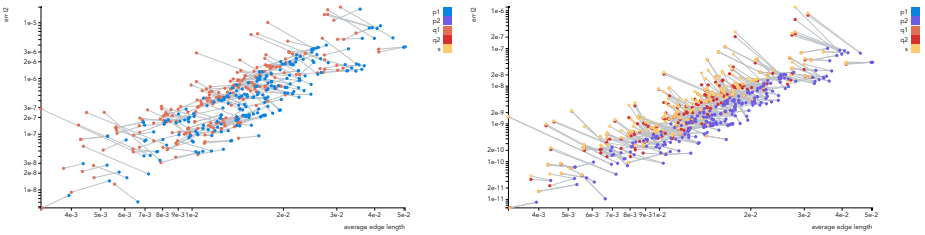
with Lamè parameters $E = 200$ and $\nu = 0.35$. In addition to standard tensor product bases for hexes, we compare to the popular *serendipity* bases [2][Chapter 6], which have only 20 nodes per element instead of 27.

We select 2 sources for our data: (1) the Hexalab dataset containing results of 16 state-of-the-art hex-meshing techniques [107], (2) the Thingi10k dataset [110] consisting of triangulated surfaces. For each dataset, we create a corresponding tet-mesh dataset out of the surface of the hex-mesh using TetWild [68] with matching number of vertices. Note that, since matching the number of vertices is an heuristic process, we discard all meshes where the difference in the number of vertices is larger than 5% 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 the hex- and tet-meshes differs more than 10^{-3} of the diagonal of the bounding box of the hex-mesh surfaces. Finally, to avoid trivial results (i.e., there is nothing to solve but the boundary conditions which are known), we discard all meshes whose ratio between boundary and total vertices is more than 30%. Note that since we do not want to discard any Hexalab meshes, if the 30% ratio is not met, we perform one step of uniform refinement to increase the number of interior vertices. In summary, the two datasets are:

1. 237 Hexalab hex-meshes and 237 tet-meshes generated with Tetwild.
2. 3 200 hex-meshes generated with MeshGems [94] and 3 200 tetmeshes generated with Tetwild both obtained from the surfaces in the Thingi10k dataset.

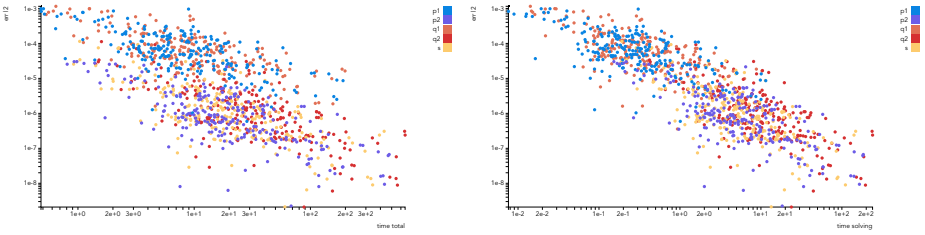


Poisson equation (1)

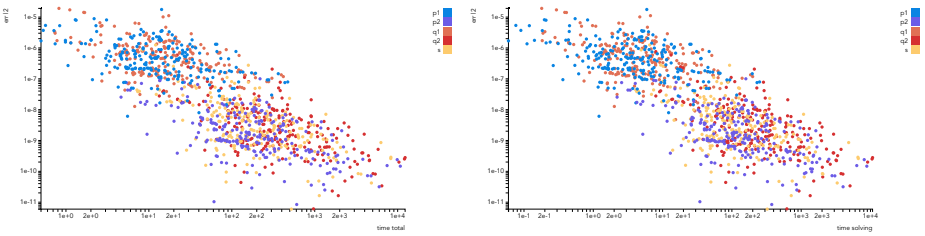


linear elasticity (4)

Figure 10: L_2 error versus average mesh size for linear (left) and quadratic (right) elements. The line connects two points belonging to the same model.

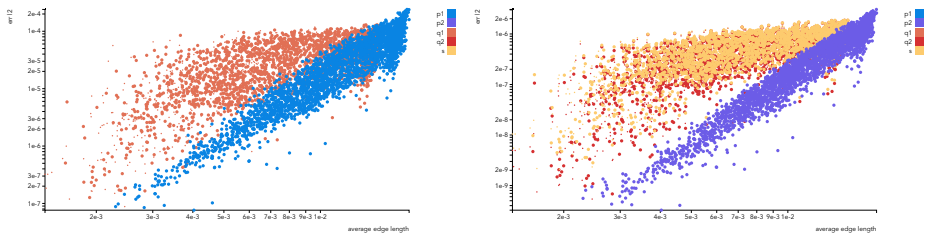


Poisson equation (1)

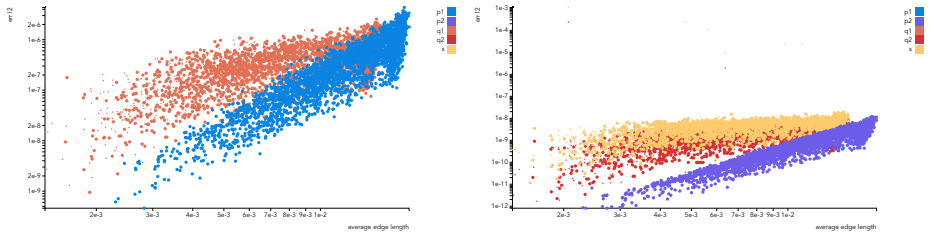


linear elasticity (4)

Figure 11: Total time (left) and solving time (right) versus the L_2 error.



Poisson equation (1)



linear elasticity (4)

Figure 12: L_2 error versus average mesh size for linear (left) and quadratic (right) elements. The smaller dot sizes indicate models with inverted elements.

For the sake of conciseness we report only the most significant results. Many other metrics (e.g., error H^1, 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 tet-meshes, Meshgems and the methods used in the Hexalab dataset do not provide any guarantee. We report that out of the 237 Hexalab meshes, 8 (3.4%) of them contain at least one inverted element (2 from [79] and 6 from [78]). For the Thingi10k dataset, Meshgems produces 577 (18.0%) meshes with at least one invalid element. To check if a hexahedron has 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.

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 [40] 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, Figure 10. We confirm the results of Section 4 for the state-of-the-art hex-meshing methods; the accuracy of the solution on a hex- and tet-mesh is comparable for both Poisson and linear elasticity. Figure 11 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

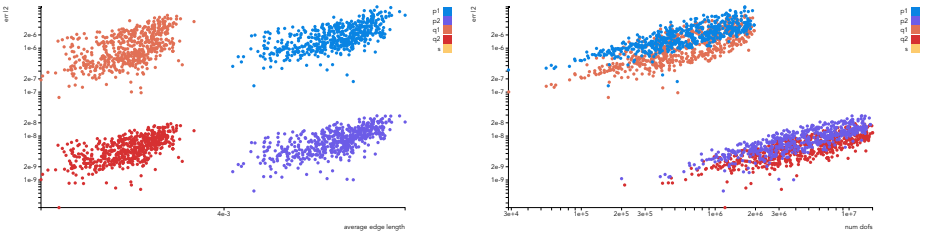
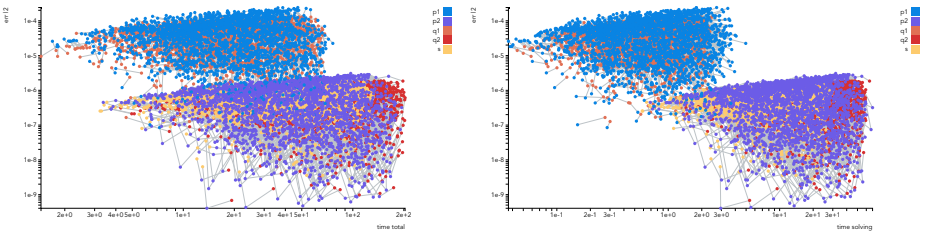
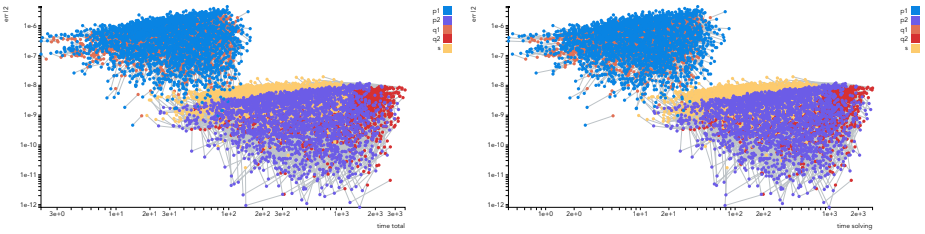


Figure 13: Average edge length (left) and number of degrees of freedom (right) versus the L_2 error for Poisson equation (1) on 580 "uniform" hex-meshes.



Poisson equation (1)



linear elasticity (4)

Figure 14: Total time (left) and solving time (right) versus the L_2 error.

but less accurate than Q_2 elements. However this advantage is not consistent enough to change the conclusion related to tetrahedral elements. Statistics for the individual hex-meshing method are available in the interactive plot.

Thingi10k. We repeated the same experiment on 3 200 hex-meshes generated with MeshGems. For this large dataset it is interesting to note that the edge length versus error trend (Figure 12) is different between hexs and tets: tets exhibit the expected convergence while hexs are more "flat". This effect comes from the fact that MeshGems is an octree based method and has the tendency of creating highly anisotropic meshes. This effect can be mitigated by reducing the range of refinement of the octree; by doing so the results become similar to tet-meshes and the one on the Hexalab dataset, Figure 13.

We also compared running and solve times (Figure 14) and, as expected, serendipity elements are faster than Q_2 elements but have larger error. Tetrahedral elements are between the two hex-elements: their accuracy is similar to Q_2 and a running/solving time similar to serendipity.

6. Conclusion

We presented a large scale, quantitative study of several common types of finite elements applied to five elliptic PDEs. We compared the performance of different unstructured and structured discretizations and finite element bases. Our results are remarkably consistent on all elliptic PDEs we tried.

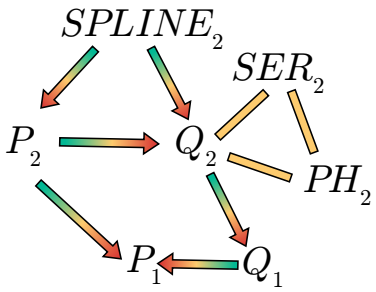


Figure 15: The arrows indicate which method is inferior; the yellow boxes indicate that the methods are comparable.

We summarize our findings in Figure 15, which allows us to draw the following conclusions:

1. Consistently with well-known observations, P_1 elements are less efficient 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. For hex-dominant meshes, a mix of Q_2 elements and quadratic polyhedral elements can be used (we refer to this hybrid element type as PH_2). This is similar to the construction presented in [111] (but without spline elements). This approach performs similar to Q_2: it would be interesting to add to our study a comparison with other polyhedral methods [112], which we leave as future work.

4. P_2 elements are generally more efficient than P_1, Q_1, Q_2, SER_2, PH_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.

5. Quadratic spline elements SPLINE_2 (on a regular lattice) are more efficient than Q_2 elements (Section 4.5). 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 [108]). Their application, however, is severely 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 [111, 113], their performance advantage is considerably reduced. For the typical hexahedral semi-structured meshes obtained with existing technology, the difference between using a mix of Q_2 and spline elements from pure Q_2 elements is not significant; thus we can expect P_2 elements to be more efficient in most cases.

For elliptic PDEs, unstructured tetrahedral meshes with quadratic Lagrangian basis are thus an ideal choice for a black box analysis pipeline: robust tetrahedral meshing algorithms that can process thousands of real-world models exist [68], and p-refinement can be used to compensate for the rare badly shaped triangles introduced by the meshing algorithms [114].

We leave the extension of this study to non-elliptic PDEs, multiphysics and collision response as future work. Another interesting extension is the study of bases with orders higher then 2, as is typically the case in IGA applications.

Acknowledgment

This work was supported in part through the NYU IT High Performance Computing resources, services, and staff expertise. This work was partially supported by the NSF CAREER award with number 1652515, the NSF grant IIS-1320635, the NSF grant DMS-1436591, the NSF grant 1835712, the SNSF grant P2TIP2_175859, a gift from Adobe Research, and a gift from nTopology.

References

References

[1] I. Babuska, M. Suri, Locking effects in the finite element approximation of elasticity problems, Tech. rep. (1990). doi:10.21236/ada239645.

[2] O. C. Zienkiewicz, R. L. Taylor, J. Z. Zhu, The Finite Element Method: Its Basis and Fundamentals, Elsevier, 2005.

[3] T. J. R. Hughes, J. A. Cottrell, Y. Bazilevs, Isogeometric analysis: Cad, finite elements, nurbs, exact geometry and mesh refinement, Computer Methods in Applied Mechanics and Engineering 194 (39-41) (2005) 4135–4195. doi:10.1016/j.cma.2004.10.008.

[4] K. Salari, P. Knupp, Code verification by the method of manufactured solutions, Tech. rep. (06 2000). doi:10.2172/759450.

[5] A. O. Cifuentes, A. Kalbag, A performance study of tetrahedral and hexahedral elements in 3-d finite element structural analysis, Finite Elements in Analysis and Design 12 (3-4) (1992) 313–318. doi:10.1016/0168-874x(92)90040-j.

[6] S. E. Benzley, E. Perry, K. Merkley, B. Clark, G. Sjaardama, 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, 1995, pp. 179–191.

[7] S. C. Tadepalli, A. Erdemir, P. R. Cavanagh, 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, 2010. doi:10.1115/sbc2010-19427.

[8] S. C. Tadepalli, A. Erdemir, P. R. Cavanagh, Comparison of hexahedral and tetrahedral elements in finite element analysis of the foot and footwear, Journal of Biomechanics 44 (12) (2011) 2337–2343. doi:10.1016/j.jbiomech.2011.05.006.

[9] E. Wang, T. Nelson, R. Rauch, Back to elements-tetrahedra vs. hexahedra, in: Proceedings of the 2004 international ANSYS conference, ANSYS Pennsylvania, 2004.

[10] A. Ramos, J. A. Simões, Tetrahedral versus hexahedral finite elements in numerical modelling of the proximal femur, Medical Engineering & Physics 28 (9) (2006) 916–924. doi:10.1016/j.medengphy.2005.12.006.

[11] X. Bourdin, X. Trosseille, P. Petit, P. Beillas, 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, 2007.

[12] B. L. D. Oliveira, J. Sundnes, 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, 2016. doi:10.7712/100016.1801.9193.

[13] EDF, Code_aster, https://www.code-aster.org/ (2018).

[14] 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, D. Wells, The deal.II library,

version 9.0, Journal of Numerical Mathematics 26 (4) (2018) 173–183. doi:10.1515/jnma-2018-0054.

[15] M. S. Alnæs, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. E. Rognes, G. N. Wells, The FEniCS project version 1.5, Archive of Numerical Software 3 (100). doi:10.11588/ans.2015.100.20553.

[16] Elmer FEM, http://www.elmerfem.org/ (2018).

[17] P. S. Ltd., FEATool Multiphysics v1.10, user's guide, https://www.featool.com (2019).

[18] C. Prud'homme, V. Chabannes, V. Doyeux, M. Ismail, A. Samake, G. Pena, Feel++ : A computational framework for galerkin methods and advanced numerical methods, ESAIM: Proceedings 38 (2012) 429–455. doi:10.1051/proc/201238024.

[19] M. A. Heroux, R. A. Bartlett, V. E. Howle, R. J. Hoekstra, J. J. Hu, T. G. Kolda, R. B. Lehoucq, K. R. Long, R. P. Pawlowski, E. T. Phipps, A. G. Salinger, H. K. Thornquist, R. S. Tuminaro, J. M. Willenbring, A. Williams, K. S. Stanley, An overview of the Trilinos project, ACM Trans. Math. Softw. 31 (3) (2005) 397–423. doi:http://doi.acm.org/10.1145/1089014.1089021.

[20] A. T. T. McRae, G.-T. Bercea, L. Mitchell, D. A. Ham, C. J. Cotter, Automated generation and symbolic manipulation of tensor product finite elements, SIAM Journal on Scientific Computing 38 (5) (2016) S25–S47. doi:10.1137/15m1021167.

[21] F. Hecht, New development in FreeFem++, J. Numer. Math. 20 (3-4) (2012) 251–265.

[22] C. Geuzaine, 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, Vol. 7, Wiley, 2008, pp. 1010603–1010604.

[23] Y. Renard, J. Pommier, Getfem++, an open source generic c++ library for finite element methods, http://getfem.org/ (2018).

[24] B. S. Kirk, J. W. Peterson, R. H. Stogner, G. F. Carey, `libMesh`: A C++ Library for Parallel Adaptive Mesh Refinement/Coarsening Simulations, Engineering with Computers 22 (3–4) (2006) 237–254, https://doi.org/10.1007/s00366-006-0049-3.

[25] MFEM: Modular finite element methods library, mfem.org. doi:10.11578/dc.20171025.1248.

[26] C. Cantwell, D. Moxey, A. Comerford, A. Bolis, G. Rocco, G. Mengaldo, D. D. Grazia, S. Yakovlev, J.-E. Lombard, D. Ekelschot, B. Jordi, H. Xu, Y. Mohamied, C. Eskilsson, B. Nelson, P. Vos, C. Biotto, R. Kirby, S. Sherwin, Nektar++: An open-source spectral/hpelement framework, Computer Physics Communications 192 (2015) 205–219. doi:10.1016/j.cpc.2015.02.008.

[27] J. Schöberl, C++11 implementation of finite elements in NGSolve, Institute for Analysis and Scientific Computing, Vienna University of Technology.

[28] B. Patzák, OOFEM - an object-oriented simulation tool for advanced modeling of materials and structures, Acta Polytechnica 52 (6) (2012) 59–66.

[29] T. Schneider, J. Dumas, X. Gao, D. Zorin, D. Panozzo, PolyFEM, https://polyfem.github.io/ (2019).

[30] T. Šoltys, Range software, http://www.range-software.com/ (2019).

[31] F. Faure, C. Duriez, H. Delingette, J. Allard, B. Gilles, S. Marchesseau, H. Talbot, H. Courtecuisse, G. Bousquet, I. Peterlik, S. Cotin, SOFA: A multi-model framework for interactive physical simulation, in: Studies in Mechanobiology, Tissue Engineering and Biomaterials, Springer Berlin Heidelberg, 2012, pp. 283–321. doi:10.1007/8415_2012_125.

[32] J. Barbič, F. S. Sin, D. Schroeder, Vega FEM Library, http://www.jernejbarbic.com/vega (2012).

[33] Fea-compare, https://github.com/kostyfisik/FEA-compare (2018).

[34] ANSYS Inc., ANSYS®, https://www.ansys.com/ (2019).

[35] ABAQUS Inc., Abaqus FEA, http://www.simulia.com (2019).

[36] COMSOL Inc., COMSOL Multiphysics, https://www.comsol.com/ (2018).

[37] M. Aigner, C. Heinrich, B. Jüttler, E. Pilgerstorfer, B. Simeon, A. V. Vuong, Swept volume parameterization for isogeometric analysis, in: Mathematics of Surfaces XIII, Springer Berlin Heidelberg, 2009, pp. 19–44. doi:10.1007/978-3-642-03596-8_2.

[38] T. Martin, E. Cohen, Volumetric parameterization of complex objects by respecting multiple materials, Computers & Graphics 34 (3) (2010) 187–197. doi:10.1016/j.cag.2010.03.011.

[39] B. Li, X. Li, K. Wang, H. Qin, Surface mesh to volumetric spline conversion with generalized polycubes, IEEE Transactions on Visualization and Computer Graphics 19 (9) (2013) 1539–1551. doi:10.1109/tvcg.2012.177.

[40] R. D. Falgout, U. M. Yang, hypre: A library of high performance preconditioners, in: Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2002, pp. 632–641. doi:10.1007/3-540-47789-6_66.

[41] A. De Coninck, B. De Baets, D. Kourounis, F. Verbosio, O. Schenk, S. Maenhout, J. Fostier, Needles: Toward large-scale genomic prediction with marker-by-environment interaction 203 (1) (2016) 543–555. doi:10.1534/genetics.115.179887.

[42] F. Verbosio, A. D. Coninck, D. Kourounis, O. Schenk, Enhancing the scalability of selected inversion factorization algorithms in genomic prediction, Journal of Computational Science 22 (Supplement C) (2017) 99 – 108. doi:10.1016/j.jocs.2017.08.013.

[43] D. Kourounis, A. Fuchs, O. Schenk, Towards the next generation of multiperiod optimal power flow solvers, IEEE Transactions on Power Systems PP (99) (2018) 1–10. doi:10.1109/TPWRS.2017.2789187.

[44] G. Guennebaud, B. Jacob, et al., Eigen v3, http://eigen.tuxfamily.org (2010).

[45] J. Shewchuk, Delaunay Mesh Generation, Chapman and Hall/CRC, 2012. doi:10.1201/b12987.

[46] G. F. Carey, Computational Grids: Generations, Adaptation & Solution Strategies, CRC Press, 1997.

[47] S. J. Owen, A survey of unstructured mesh generation technology, in: IMR, 1998, pp. 239–267.

[48] T. J. Tautges, The generation of hexahedral meshes for assembly geometry: Survey and progress, International Journal for Numerical Methods in Engineering 50 (12) (2001) 2617–2642. doi:10.1002/nme.139.

[49] L. P. Chew, Guaranteed-quality mesh generation for curved surfaces, in: Proceedings of the ninth annual symposium on Computational geometry - SCG '93, ACM Press, 1993. doi:10.1145/160985.161150.

[50] J. R. Shewchuk, Tetrahedral mesh generation by delaunay refinement, in: Proceedings of the fourteenth annual symposium on Computational geometry - SCG '98, ACM Press, 1998. doi:10.1145/276884.276894.

[51] J. Ruppert, A delaunay refinement algorithm for quality 2-dimensional mesh generation, Journal of Algorithms 18 (3) (1995) 548–585. doi:10.1006/jagm.1995.1021.

[52] D. R. Sheehy, New bounds on the size of optimal meshes, Computer Graphics Forum 31 (5) (2012) 1627–1635. doi:10.1111/j.1467-8659.2012.03168.x.

[53] J.-F. Remacle, A two-level multithreaded delaunay kernel, Computer-Aided Design 85 (2017) 2–9. doi:10.1016/j.cad.2016.07.018.

[54] Q. Du, D. Wang, Tetrahedral mesh generation and optimization based on centroidal voronoi tessellations, International journal for numerical methods in engineering 56 (9) (2003) 1355–1373.

[55] P. Alliez, D. Cohen-Steiner, M. Yvinec, M. Desbrun, Variational tetrahedral meshing, ACM Transactions on Graphics 24 (3) (2005) 617. doi:10.1145/1073204.1073238.

[56] J. Tournois, C. Wormser, P. Alliez, M. Desbrun, Interleaving delaunay refinement and optimization for practical isotropic tetrahedron mesh generation, ACM Transactions on Graphics 28 (3) (2009) 1. doi:10.1145/1531326.1531381.

[57] M. Murphy, D. M. Mount, C. W. Gable, A point-placement strategy for conforming delaunay tetrahedralization, International Journal of Computational Geometry & Applications 11 (06) (2001) 669–682. doi:10.1142/s0218195901000699.

[58] D. Cohen-Steiner, E. C. de Verdière, M. Yvinec, Conforming delaunay triangulations in 3D, in: Proceedings of the eighteenth annual symposium on Computational geometry - SCG '02, ACM Press, 2002. doi:10.1145/513400.513425.

[59] L. P. Chew, Constrained delaunay triangulations, in: Proceedings of the third annual symposium on Computational geometry - SCG '87, ACM Press, 1987. doi:10.1145/41958.41981.

[60] H. Si, K. Gärtner, Meshing piecewise linear complexes by constrained delaunay tetrahedralizations, in: Proceedings of the 14th international meshing roundtable, Springer, 2005, pp. 147–163.

[61] J. R. Shewchuk, Constrained delaunay tetrahedralizations and provably good boundary recovery, in: Eleventh International Meshing Roundtable, Sandia National Laboratories, 2002, pp. 193–204.

[62] H. Si, J. R. Shewchuk, Incrementally constructing and updating constrained delaunay tetrahedralizations with finite-precision coordinates, Engineering with Computers 30 (2) (2014) 253–269. doi:10.1007/s00366-013-0331-0.

[63] S.-W. Cheng, T. K. Dey, J. A. Levine, A practical delaunay meshing algorithm for a large class of domains, in: Proceedings of the 16th International Meshing Roundtable, Springer, 2008, pp. 477–494.

[64] J.-D. Boissonnat, S. Oudot, Provably good sampling and meshing of surfaces, Graphical Models 67 (5) (2005) 405–451. doi:10.1016/j.gmod.2005.01.004.

[65] C. Jamin, P. Alliez, M. Yvinec, J.-D. Boissonnat, CGALmesh: A generic framework for delaunay mesh generation, ACM Transactions on Mathematical Software 41 (4) (2015) 1–24. doi:10.1145/2699463.

[66] T. K. Dey, J. A. Levine, Delpsc: A delaunay mesher for piecewise smooth complexes, in: Proceedings of the twenty-fourth annual symposium on Computational geometry - SCG '08, ACM Press, 2008. doi:10.1145/1377676.1377712.

[67] L. Chen, J.-c. Xu, Optimal delaunay triangulations, Journal of Computational Mathematics (2004) 299–308.

[68] Y. Hu, Q. Zhou, X. Gao, A. Jacobson, D. Zorin, D. Panozzo, Tetrahedral meshing in the wild, ACM Transactions on Graphics 37 (4) (2018) 1–14. doi:10.1145/3197517.3201353.

[69] N. Molino, R. Bridson, R. Fedkiw, Tetrahedral mesh generation for deformable bodies, in: Proc. Symposium on Computer Animation, 2003.

[70] J. R. Bronson, J. A. Levine, R. T. Whitaker, Lattice cleaving: Conforming tetrahedral meshes of multimaterial domains with bounded quality, in: Proceedings of the 21st International Meshing Roundtable, Springer Berlin Heidelberg, 2013, pp. 191–209. doi:10.1007/978-3-642-33573-0_12.

[71] F. Labelle, J. R. Shewchuk, Isosurface stuffing: Fast tetrahedral meshes with good dihedral angles, in: ACM SIGGRAPH 2007 papers on - SIGGRAPH '07, ACM Press, 2007. doi:10.1145/1275808.1276448.

[72] C. Doran, A. Chang, R. Bridson, Isosurface stuffing improved: Acute lattices and feature matching, in: ACM SIGGRAPH 2013 Talks on - SIGGRAPH '13, ACM Press, 2013. doi:10.1145/2504459.2504507.

[73] R. Bridson, C. Doran, Quartet: A tetrahedral mesh generator that does isosurface stuffing with an acute tetrahedral tile, https://github.com/crawforddoran/quartet (2014).

[74] J.-C. Cuillière, V. Francois, J.-M. Drouet, Automatic 3D mesh generation of multiple domains for topology optimization methods, in: Proceedings of the 21st International Meshing Roundtable, Springer Berlin Heidelberg, 2013, pp. 243–259. doi:10.1007/978-3-642-33573-0_15.

[75] F. Alauzet, D. Marcum, 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, 2014, pp. 241–261. doi:10.1007/978-3-319-02335-9_14.

[76] R. Haimes, Moss: Multiple orthogonal strand system, in: Proceedings of the 22nd International Meshing Roundtable, Springer International Publishing, 2014, pp. 75–91. doi:10.1007/978-3-319-02335-9_5.

[77] J. F. Shepherd, C. R. Johnson, Hexahedral mesh generation constraints, Engineering with Computers 24 (3) (2008) 195–213. doi:10.1007/s00366-008-0091-4.

[78] X. Gao, T. Martin, S. Deng, E. Cohen, Z. Deng, G. Chen, Structured volume decomposition via generalized sweeping, IEEE Transactions on Visualization and Computer Graphics 22 (7) (2016) 1899–1911. doi:10.1109/tvcg.2015.2473835.

[79] M. Livesu, A. Muntoni, E. Puppo, R. Scateni, Skeleton-driven adaptive hexahedral meshing of tubular shapes, Computer Graphics Forum 35 (7) (2016) 237–246. doi:10.1111/cgf.13021.

[80] Cubit, https://cubit.sandia.gov, accessed: 2019-01-30 (2019).

[81] R. Schneiders, R. Bünten, Automatic generation of hexahedral finite element meshes, Computer Aided Geometric Design 12 (7) (1995) 693–707. doi:10.1016/0167-8396(95)00013-v.

[82] R. Schneiders, A grid-based algorithm for the generation of hexahedral element meshes, Engineering with Computers 12 (3-4) (1996) 168–177. doi:10.1007/bf01198732.

[83] Y. Su, K. H. Lee, A. S. Kumar, Automatic hexahedral mesh generation for multi-domain composite models using a hybrid projective grid-based method, Computer-Aided Design 36 (3) (2004) 203–215. doi:10.1016/s0010-4485(03)00079-4.

[84] Y. Zhang, C. Bajaj, Adaptive and quality quadrilateral/hexahedral meshing from volumetric data, Computer Methods in Applied Mechanics and Engineering 195 (9-12) (2006) 942–960. doi:10.1016/j.cma.2005.02.016.

[85] H. Zhang, G. Zhao, X. Ma, Adaptive generation of hexahedral element mesh using an improved grid-based method, Computer-Aided Design 39 (10) (2007) 914–928. doi:10.1016/j.cad.2007.05.016.

[86] R. Schneiders, R. Schindler, F. Weiler, Octree-based generation of hexahedral element meshes, in: In Proceedings Of The 5th International Meshing Roundtable, Sandia National Laboratories, 1996.

[87] Y. Ito, A. M. Shih, B. K. Soni, Octree-based reasonable-quality hexahedral mesh generation using a new set of refinement templates, International Journal for Numerical Methods in Engineering 77 (13) (2009) 1809–1833. doi:10.1002/nme.2470.

[88] L. Maréchal, Advances in octree-based all-hexahedral mesh generation: Handling sharp features, in: Proceedings of the 18th international meshing roundtable, Springer, 2009, pp. 65–84.

[89] J. Qian, Y. Zhang, Sharp feature preservation in octree-based hexahedral mesh generation for cad assembly models, in: Proceedings of the 19th International Meshing Roundtable, Springer Berlin Heidelberg, 2010, pp. 243–262. doi:10.1007/978-3-642-15414-0_15.

[90] M. S. Ebeida, A. Patney, J. D. Owens, E. Mestreau, Isotropic conforming refinement of quadrilateral and hexahedral meshes using two-refinement templates, International Journal for Numerical Methods in Engineering 88 (10) (2011) 974–985. doi:10.1002/nme.3207.

[91] Y. Zhang, X. Liang, G. Xu, 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, 2013, pp. 155–172. doi:10.1007/978-3-642-33573-0_10.

[92] A. H. Elsheikh, M. Elsheikh, A consistent octree hanging node elimination algorithm for hexahedral mesh generation, Advances in Engineering Software 75 (2014) 86–100. doi:10.1016/j.advengsoft.2014.05.005.

[93] S. J. Owen, R. M. Shih, C. D. Ernst, A template-based approach for parallel hexahedral two-refinement, Computer-Aided Design 85 (2017) 34–52. doi:10.1016/j.cad.2016.09.005.

[94] MeshGems, http://meshgems.com/volume-meshing-meshgems-hexa.html, accessed: 2018-06-05 (2018).

[95] J. Gregson, A. Sheffer, E. Zhang, All-hex mesh generation via volumetric polycube deformation, Computer Graphics Forum 30 (5) (2011) 1407–1416. doi:10.1111/j.1467-8659.2011.02015.x.

[96] M. Livesu, N. Vining, A. Sheffer, J. Gregson, R. Scateni, Polycut: Monotone graph-cuts for polycube base-complex construction, ACM Transactions on Graphics 32 (6) (2013) 1–12. doi:10.1145/2508363.2508388.

[97] J. Huang, T. Jiang, Z. Shi, Y. Tong, H. Bao, M. Desbrun, ℓ1-based construction of polycube maps from complex shapes, ACM Transactions on Graphics 33 (3) (2014) 1–11. doi:10.1145/2602141.

[98] X. Fang, W. Xu, H. Bao, J. Huang, All-hex meshing using closed-form induced polycube, ACM Transactions on Graphics 35 (4) (2016) 1–9. doi:10.1145/2897824.2925957.

[99] X.-M. Fu, C.-Y. Bai, Y. Liu, Efficient volumetric polycube-map construction, Computer Graphics Forum 35 (7) (2016) 97–106. doi:10.1111/cgf.13007.

[100] H. Zhao, N. Lei, X. Li, P. Zeng, K. Xu, X. Gu, Robust edge-preserving surface mesh polycube deformation, Computational Visual Media 4 (1) (2018) 33–42. doi:10.1007/s41095-017-0100-x.

[101] M. Nieser, U. Reitebuch, K. Polthier, Cubecover- parameterization of 3D volumes, Computer Graphics Forum 30 (5) (2011) 1397–1406. doi:10.1111/j.1467-8659.2011.02014.x.

[102] J. Huang, Y. Tong, H. Wei, H. Bao, Boundary aligned smooth 3D cross-frame field, ACM Transactions on Graphics 30 (6) (2011) 1. doi:10.1145/2070781.2024177.

[103] Y. Li, Y. Liu, W. Xu, W. Wang, B. Guo, All-hex meshing using singularity-restricted field, ACM Transactions on Graphics 31 (6) (2012) 1. doi:10.1145/2366145.2366196.

[104] T. Jiang, J. Huang, Y. Wang, Y. Tong, H. Bao, Frame field singularity correctionfor automatic hexahedralization, IEEE Transactions on Visualization and Computer Graphics 20 (8) (2014) 1189–1199. doi:10.1109/tvcg.2013.250.

[105] J. Solomon, A. Vaxman, D. Bommes, Boundary element octahedral fields in volumes, ACM Transactions on Graphics 36 (3) (2017) 1–16. doi:10.1145/3065254.

[106] H. Liu, P. Zhang, E. Chien, J. Solomon, D. Bommes, Singularity-constrained octahedral fields for hexahedral meshing, ACM Transactions on Graphics 37 (4) (2018) 1–17. doi:10.1145/3197517.3201344.

[107] M. Bracci, M. Tarini, N. Pietroni, M. Livesu, P. Cignoni, Hexalab.net: An online viewer for hexahedral meshes, Computer-Aided Design 110 (2019) 24–36. doi:10.1016/j.cad.2018.12.003.

[108] D. Schillinger, S. J. Hossain, T. J. Hughes, Reduced bézier element quadrature rules for quadratic and cubic splines in isogeometric analysis, Computer Methods in Applied Mechanics and Engineering 277 (2014) 1–45. doi:10.1016/j.cma.2014.04.008.
URL https://doi.org/10.1016/j.cma.2014.04.008

[109] R. Franke, A critical comparison of some methods for interpolation of scattered data, Tech. rep. (12 1979). doi:10.21236/ada081688.

[110] Q. Zhou, A. Jacobson, Thingi10K: A dataset of 10, 000 3D-printing models, CoRR abs/1605.04797. arXiv:1605.04797.

[111] T. Schneider, J. Dumas, X. Gao, D. Panozzo, D. Zorin, Poly-spline finite element method, ACM Transactions on Graphics 38 (3) (2019) 1–14.

[112] K. Hormann, N. Sukumar (Eds.), Generalized Barycentric Coordinates in Computer Graphics and Computational Mechanics, Taylor & Francis, CRC Press, Boca Raton, 2017.

[113] X. Wei, Y. J. Zhang, D. Toshniwal, H. Speleers, X. Li, C. Manni, J. A. Evans, T. J. R. Hughes, 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 (2018) 609–639. doi:10.1016/j.cma.2018.07.013.

[114] T. Schneider, Y. Hu, J. Dumas, X. Gao, D. Panozzo, D. Zorin, Decoupling simulation accuracy from mesh quality, ACM Transactions on Graphics 37 (6) (2018) 1–14. doi:10.1145/3272127.3275067.